WOA 2015
XVI WORKSHOP "DAGLI OGGETTI AGLI AGENTI"
NAPOLI
17-19 GIUGNO 2015

PROCEEDINGS OF THE 16TH WORKSHOP "FROM OBJECT TO AGENTS"

WOA 2015

Claudia di Napoli, Silvia Rossi, Mariacarla Staffa (eds.)

NAPLES, ITALY
JUNE 17-19, 2015

DIETI UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

SAPHARI
SAFE AND AUTONOMOUS PHYSICAL HUMAN-AWARE ROBOT INTERACTION

Istituto di Calcolo e Reti ad Alte Prestazioni
CNR

Ceur-workshop proceedings
http:www.cogrobotics.unina.it/woa15

WORKSHOP CHAIR

Silvia Rossi Università degli Studi di Napoli "Federico II"

STEERING COMMITTEE

Giuliano Armano	Università di Cagliari
Matteo Baldoni	Università di Torino
Federico Bergenti	Università di Parma
Giacomo Cabri	Università di Modena-Reggio Emilia
Massimo Cossentino	ICAR-CNR
Giancarlo Fortino	Università della Calabria
Viviana Mascardi	Università di Genova
Andrea Omicini	Università di Bologna
Agostino Poggi	Università di Parma
Alessandro Ricci	Università di Bologna
Corrado Santoro	Università di Catania
Giuseppe Vizzari	Università di Milano-Bicocca

INVITED SPEAKERS

Onofrio Gigliotta	Università degli Studi di Napoli "Federico II"
Kostas Stathis	Royal Holloway University of London

PROGRAM COMMITTEE

Francesco Amigoni	Politecnico di Milano
Giuliano Armano	Università di Cagliari
Matteo Baldoni	Università di Torino
Cristina Baroglio	Università di Torino
Federico Bergenti	Università di Parma
Enrico Blanzieri	Università di Trento
Piero Bonatti	Università di Napoli
Giacomo Cabri	Università di Modena e Reggio Emilia
Cristiano Castelfranchi	ISTC-CNR
Antonio Chella	Università di Palermo
Massimo Cossentino	ICAR-CNR
Claudia Di Napoli	ICAR-CNR
Rino Falcone	ISTC-CNR
Alessandro Farinelli	Università di Verona
Giancarlo Fortino	Università della Calabria
Salvatore Gaglio	ICAR-CNR
Alfredo Garro	Università della Calabria
Paolo Giorgini	Università di Trento
Ignazio Infantino	ICAR-CNR
Letizia Leonardi	Università di Modena e Reggio Emilia
Sara Manzoni	Università Milano Bicocca
Viviana Mascardi	Università di Genova
Emanuela Merelli	Università di Camerino
Aniello Murano	Università di Napoli
Andrea Omicini	Università di Bologna
Paolo Petta	OFAI Vienna
Agostino Poggi	Università di Parma
Alessandro Ricci	Università di Bologna
Domenico Rosaci	Università di Reggio Calabria
Silvia Rossi	Università di Napoli
Corrado Santoro	Università Catania
Valeria Seidita	Università di Palermo
Mariacarla Staffa	Università di Napoli
Salvatore Venticinque	Seconda Università di Napoli
Giuseppe Vizzari	Università Milano Bicocca

ADDITIONAL REVIEWERS

Alba Amato	Seconda Università di Napoli
Jacopo Binchi	Università di Camerino
Massimiliano De Benedetti	Università Catania
Marco Piangerelli	Università di Camerino
Luca Sabatucci	ICAR-CNR
Alessandro Sapienza	ISTC-CNR

TABLE OF CONTENTS

MUSA: a Middleware for User-driven Service Adaptation

M. Cossentino, C. Lodato, S. Lopes, L. Sabatucci
ICAR-CNR, Palermo, Italy
{cossentino, c.lodato, s.lopes, sabatucci}@pa.icar.cnr.it

***Abstract*—One of the current challenges of Service Oriented Engineering is to provide instruments for dealing with dynamic and unpredictable environments and changing user requirements. Traditional approaches based on static workflows provide little support for adapting at run-time the flow of activities.**

MUSA (Middleware for User-driven Service Adaptation) is a holonic multi-agent system for the self-adaptive composition and orchestration of services in a distributed environment.

I. Introduction

In the last decade web-services have gained industry-wide acceptance as the universal standard for enterprise application integration [1]. Their strengths is to be easily combined as building blocks of a large distributed and scalable software application [2]. On the other side, fostering user participation in business process is an enormous opportunity, where the value is direct and proportional to the capability to customize service parameters according to users' needs [3].

To date, BPEL is one of the most used standards for implementing the orchestration of services. Even if workflow-based languages are greatly supported by industry and research, their approach reveals being static and not easy to extend for supporting some advanced features as, for example, run-time modification of the flow of events, dynamic hierarchies of services, integration of user preferences and, moreover, it is not easy to provide a system for run-time execution, rescheduling and monitoring of activities that is also able to deal with unexpected failures and optimization.

Recently, the research community on services has been very active in defining techniques, methods and middleware for supporting dynamic execution model for workflows.

This paper presents MUSA (Middleware for User-driven Service Adaptation) [1], a holonic multi-agent system for the composition and the orchestration of services in a distributed and open environment.

MUSA aims at providing run-time modification of the flow of events, dynamic hierarchies of services and integration of user preferences together with a self-adaptive system for execution activities that is also able to monitor unexpected failures and to reschedule in order to optimize the flow.

Self-adaptation is based on the intuition to break the static constraints of a classic workflow model by decoupling the 'what' (the outcome the workflow requires to be addressed) from the 'how' (the way this result can be obtained) [4].

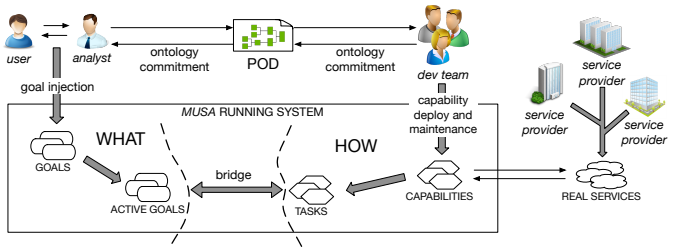


Figure 1. The MUSA vision.

Figure 1 illustrates the underlying idea. Services are provided as usual by world-wide companies, according to their own business processes. MUSA provides a platform in which 1) virtual enterprises can deploy some capabilities that wrap real services, completing them with a semantic layer for their smart use; 2) analysts and/or users can inject their goals for requesting a specific outcome. Under the hypothesis that both goals and capabilities refer to the same semantic layer (described as an ontology), then agents of the system are able to conduct a proactive means-end reasoning for composing available capabilities into task for addressing the user request.

Conceptually, this has required the following ingredients: *goal-orientation* for making user-requirements

[1]Website: http://aose.pa.icar.cnr.it/MUSA/.

explicit in the system thus breaking the strict coupling among activities of the workflow; *holonic system*, for implementing a dynamic and re-configurable architecture of autonomous and proactive agents.; *self-adaptation*, for generating smart and dynamic plans as response to user-requests and to unpredictable events of the environment.

The whole system has been implemented by using JASON [5] and CArtAgO [6], an agent facility based on the AgentSpeak language [7] and the BDI theory [8].

The running example used along the whole paper concerns the domain of *Travel Services*. The system acts as a smart tour operator for composing simple services provided in a local area as, for example, flights, trains, hotels and other tourist attractions. The objective is to provide users with a product, *Travel*, that is the composition and orchestration of atomic services.

The papers is organized as follows: Section II discusses the languages for injecting goals and deploying capabilities into the system. Section III provides details about the dynamic and distributed architecture that emerges for addressing a user-request. Section IV illustrates the core algorithm for allowing the agent to reason on goals and capabilities and for creating plans. Finally, Section V reports some considerations about the approach.

II. Decoupling Goals and Services

MUSA exploits BDI reasoning since it offers the required level of abstraction to build an autonomous and self-aware agent. In particular self-awareness is intended as the ability of agents to recognize their own capabilities (getting knowledge of their preconditions and effects), and to conduct some reasoning over them. The Belief-Desire-Intention (BDI) model was developed at the Stanford Research Institute during the activities of the Rational Agency project [9]. The BDI model assumes software agents had a *mental state* and *a decision making model* representing a promising base for implementing autonomous and self-adaptive systems [10], [11].

In order to make the system able to reason on user-requests and available capabilities a solution is to elect goals and capabilities as first-class entities as it will be described in this section.

A **state of the world** is informally described as a set of non-contradictory first order facts with the assumption that everything that is not explicitly declared is assumed to be false. This is dynamically maintained by the agents of the system as the result of their perceptions and deductions.

A **user-goal** is a desired *change* in the state of the world an actor wants to achieve. In the proposed approach a goal describes the starting state and the final states in terms of states of the world. It is therefore necessary to make a sharp distinction between BDI goals and user-goals. A *user-goal* is injected into the system at run-time (and therefore it not known a-priori by agents) On the other side a *BDI goal* is defined at design-time and the plans for addressing it are hard-coded into the agent. Another difference is that an agent is automatically committed to fulfill all its BDI goals, whereas it owns a higher level of autonomy with respect to user-goals [12]. For example, an agent may check whether it is able to address the goal and then it may decide of committing to it (generally when the agent can get some type of advantage from the situation).

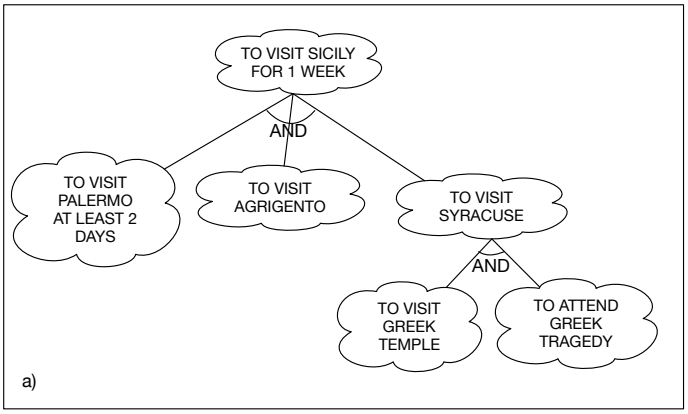
A **goal model** is a directed graph where nodes are goals and edges are AND/OR Refinement or Influence relationships. In a goal model there is exactly one root goal, and there are no refinement cycles. A goal model is an analysts instrument to create dependencies among goals.

A **capability** describes a concrete trajectory in terms of states of the world the system may intentionally use to address a given result. Every agent knows its capabilities together with the way these can be employed. The effect of a capability is an endogenous evolution of the state of the world (a function that takes a state of world and produces a new state of world). The capability can be pursued only if a given pre-condition is true whereas the post-condition must be true after the capability has been successfully executed.

A. GoalSPEC

In order to decouple user-goals from web-services, MUSA provides GoalSPEC [13], a language designed for specifying user-goals and, at the same time, enabling at the same time goal injection and software agent reasoning. It takes inspiration from languages for specifying requirements for adaptation, such as RELAX [14], however GoalSPEC is in line with the definition of goal. The language is based on structured English and it adopts a core grammar with a basic set of keywords that must be extended by plugging-in a domain ontology.

The core element of the metamodel is the *Goal* (desired by some subject). It is composed of a *Trigger Condition* and a *Final State*. The trigger condition is an event that must occur in order to start acting for addressing the goal. The final state is the desired state of the world that must be addressed. The *Subject* describes




```
agent_goal( %TO VISIT SICILY FOR 1 WEEK
    params( [curr_date] , [
       category(curr_date, datetime)]),
    tr_condition( in(curr_date,
       date(2015,7,16),
       date(2015,7,23))),
    final_state( be_visiting(sicily) ),
    herbert
)
...
agent_goal( %TO VISIT SYRACUSE
params( [t] , [
       category(t, integer)]),
    tr_condition( be_at(syracuse) ),
    final_state( and([visited(syracuse,t),
                      t>1] ),
    herbert
)
```

b)

Figure 2. On the left side, an example of goal-model the user can inject into the smart travel system. On the right side, an instance of the same goal-model, but expressed in terms of agent's beliefs.

the name of the involved person, role or group of persons that owns the responsibility to address the goal.

In the domain of the Travel Service, GoalSPEC allows the user to describe the kind of travel she desires. Examples of GoalSPEC productions are listed below:

1) WHEN date(16,2,15) THE user SHALL visited(Palermo) OR visited(Catania)

2) WHEN date(DD,MM,YY) AND ($DD > 15$ AND $DD < 20$) THE user SHALL enjoyed(beach)

3) WHEN date(18,2,15) THE user SHALL visited(Syracuse) AND attended(greek_tragedy)

Figure 2.a shows an example of goal-model where each goal must be further refined with GoalSPEC. When injected into the system, the goal is converted into a set of agent's beliefs (Figure 2.b) in which the Trigger Condition and the Final State are expressed as first-order logical conditions to be tested over the current State of the World.

For a complete specification of the syntax of GoalSPEC see [13], whereas details of the conversion into agent's beliefs are provided in [12].

Table I
ABSTRACT SPECIFICATION OF THE FLIGHT BOOKING CAPABILITY.

Name	FLIGHT_BOOKING
Input	DPTPLACE : AIRPORT, DPTDATE: DATE, ARRPLACE : AIRPORT, PASSNUM : INTEGER
Output	FLIGHID: STRING, DPTSCHEDULE: DATE, ARRSCHEDULE: DATE
Constraints	$DptPlace \neq ArrPlace$ $DptSchedule > DptDate$ $ArrSchedule > DptSchedule$ $PassNumber > 1$
Pre-Condition	$seat\_available(FlighId, DptSchedule, PassNum)$
Post-Condition	$flight\_booked(FlighId, DptSchedule, PassNum)$
Evolution	$evo = \{remove(being\_at(DptPlace)), add(being\_at(ArrPlace))\}$

B. Capability

In AI, the need for representing software agent's actions in order to implement reasoning directed towards action is a long-dated issue [15], [16], [17], [18]. An agent is able to achieve a goal by doing an action if either the agent knows what the action is or it knows that doing the action would result in the goal being satisfied [15].

We use a 'robotic-planning-like' approach to address user-goals, in which the Capability is the internal representation of an atomic unit of work that a software agent may use for addressing changes in the state of the world. A Capability is made of two components: an abstract description (a set of beliefs that makes an agent aware of owning the capability and able to reason on its use), and a concrete implementation (a set of plans for executing the job).

We defined a template for providing the abstract description of a capability as a refinement of that presented in [19] for LARKS (language for advertisement and request for knowledge sharing).

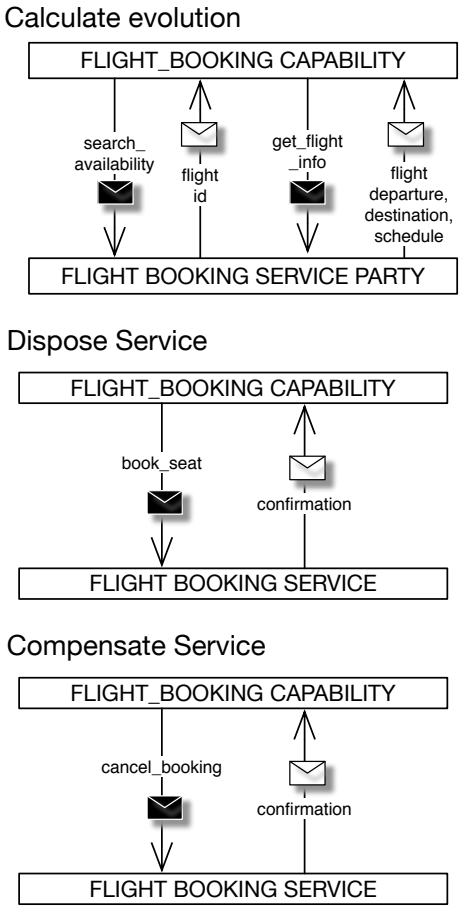


Figure 3. This figure reveals the business logic for the flight booking capability composed of three parts: the calculate evolution, the dispose service and the compensate service.

The template is made of the following elements:

- **Name** is the unique label used to refer to the capability
- **InputParams** is the definition of the input variables necessary for the execution.
- **OutputParams** is the definition of the output variables produced by the execution.
- **Constraints** is an optional (logical or structural) constraints on input/output variables.
- **Pre-Condition** is a condition that must hold in the current state of the world in order to execute the capability.
- **Post-Condition** is a condition that must hold in the final state of the world in order to assert the capability has been profitable.
- **Evolution** is a function that describes how the capability will impact the state of the world as consequence of its execution.

By reasoning on the abstract side (input/output/precondition...) the agent may decide when to use the capability. An example of abstract description is provided in Table I.

On the other side, the concrete implementation encapsulates the code for interacting with the real service by using SOAP and WSDL through the HTTP/HTTPS protocol. The implementation of a capability for dealing web-services is made of three parts: the calculate evolution, the dispose service and the compensate service.

The *calculate evolution* protocol is used when composing the whole plan to address a goal; at this stage the agent has to configure the capability for a specific context. This means to establish input/output parameters to generate a contract with other agents it is collaborating with. More details will be in Section IV-B when illustrating the algorithm for the Goal/Capability Deliberation. For instance, the calculate evolution for the flight_booking searches for flights (Figure 3) that are compatible with a given goal, i.e those flights that match with a given DptPlace, DptDate, ArrPlace and PassNumber.

The dispose service and the compensate service protocols will be used for orchestration and self-adaptation purposes (see Section IV-C).

After that a plan has been established for the execution, the orchestration phase generates, through the *dispose service*, the actual value for the user, i.e the user-goal fulfillment. For instance, the dispose protocol for the flight_booking service actually book the specified flight and produces a ticket for the user.

However a plan may be subject to changes for several purposes. The Self-Adaptation Loop monitors for failures or new goals that may affect the running plan. When a plan changes at run-time, it could be the case that some services that have been disposed are no more useful in the new plan. The proper way to proceed is to use the *compensate service* protocol in order to terminate the contract with a service. For instance, the compensate service for the flight_booking tries to cancel the user booking for a specified flight.

C. Problem Ontology Description

The previous sections have illustrated how goals and capabilities grounds on the *state of the world* and therefore, under the surface, they employ first-order predicates.

In MUSA the Problem Ontology Diagram (POD) [20] is used to provide a denotation to significant states of

the world thus giving a precise semantics to goals and capabilities

An ontology is the specification of a conceptualization made for the purpose of enabling knowledge sharing and reuse [21]. A POD is a conceptual model (and a set of guidelines [22]) used to create an ontological commitment between developers of capabilities and users who inject goals. An ontological commitment is an agreement to use a thesaurus of words in a way that is consistent (even if not complete) with respect to the theory specified by an ontology [23].

This artifact aims at producing a set of concepts, predicates and actions and at creating a semantic network in which these elements are related to one another. The representation is mainly human-oriented but it is particularly suitable for developing cognitive system that are able of storing, manipulating, reasoning on, and transferring knowledge data directly in first-order predicates [22].

Grounding goals and capability abstract description on the same ontology is fundamental to allow the system to adopt a proactive means-end reasoning to compose plans. By committing to the same ontology, capabilities and goals can be implemented and delivered by different development teams and at the same time enabling a semantic compatibility between them.

More details on the POD are in [20], whereas the link between goals and ontology is detailed in [22]. Finally we also provide GIMT (Goal Identification and Modeling Tool) a tool for supporting ontology building and goal modeling [24].

III. Holonic Architecture for Self-Adaptation

Holons provide an elegant and scalable method to guarantee knowledge sharing, distributed coordination and robustness.

Holon is a Greek term for indicating something that is simultaneously a whole and a part [25]. It has been used for introducing a new understanding of ecosystems, and their hierarchical nature. A general definition may be the following:

> A holon is a system (or phenomenon) which is an evolving self-organizing structure, consisting of other holons [26]. A holon has its own individuality, but at the same time, it is embedded in larger wholes (principle of duality or *Janus effect*).

Many concrete things in nature are organized as a holarchy (the recursive structure generated by holons and sub-holons). An example of concrete holon is an *organ* that is a part of an organism, but a whole with regard to the cells of which it is comprised. The human mind uses holarchies for organizing abstract concepts too. An example is a *word* that is part of a sentence, but a whole with regard to the letters that compose it.

A holon has not necessarily the same properties of its parts, as well as if a bird can fly, its cells can not. Holon is therefore a general term for indicating a concrete or abstract entity that has its own individuality, but at the same time, it is embedded in larger wholes.

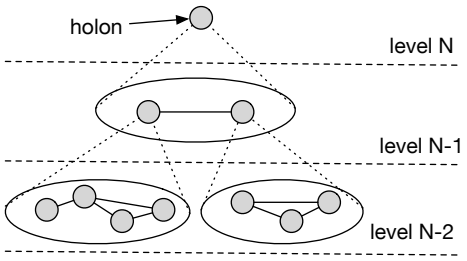


Figure 4. Holarchy layered architecture (elaborated from [27]).

However, each holon is influenced by, and influences its larger wholes. And since a holon also contains parts, it is similarly influenced by, and influences these parts: information flows bidirectionally between smaller and larger systems.

The problem of service composition may be observed as a phenomenon of holon formation [28]. In MUSA services of a choreography maintain their autonomy but they all collaborate for providing an integrated functionality. A composed service is therefore a holon who embeds other component services in a recursive fashion.

A **holonic multi agent system** is a software system made of autonomous holons where the holon is defined recursively (see Figure 4). The holon at generic level N is made of holons a level N-1 kept together by a commitment relationship. The resulting structure is a holarchy, i.e. a hierarchical structure generated by holons and sub-holons where the base case is the agent representing the atomic holon (it is not further decomposable in sub-holons).

In MUSA the commitment function that glues together all the sub-holons to form a super-holon is given by the injected GoalSPEC. The user-goals represent the common objective that the holons have to address.

A. The Basic-Holon Schema

The holon formation is an emergent phenomenon. This section provides details about the static structure of each

holon of the system whereas dynamic aspects of holon formation and execution are given in Section IV.

In MUSA all the agents are (atomic) holons, however aggregations of holons (in super-holons) may emerge at run-time for managing composed services. A holarchy is formed as the recursive replication of the same *basic schema*. This template defines that each holon of the system may contain sub-holons playing one of the three roles: service-broker, state-monitor or goal-handler.

The ***service broker*** is the role in charge of establishing a relationship with one or more end points of a remote service. The candidate service-broker owns the capability for managing the conversation with the party that provides the service (for example the flight_booking shown in Figure 3). The service broker must also be able to catch exceptions and failures and to raise the need for self-adaptation.

The ***state monitor*** is the role responsible for monitoring the user environment (both physical and simulated, including persons acting inside). The state of the world is an abstraction for the operative context in which services are going to be provided. The perception of the state of the world is often necessary for invoking services. This role is in charge of providing the service broker with the configuration of input/output parameters for properly invoking a service. It is also responsible of analyzing inconsistencies in the state of the world, due to unfeasible beliefs that could generate service failures.

The ***goal handler*** is responsible for the interpretation of the GoalSPEC and for the recruitment of the service-brokers and state monitors to form a valid holon. The recruitment is based on a procedure called Means-End Reasoning (detailed in Section IV) in which service-brokers and state monitors are selected on the base of capabilities they offer for addressing a desired state of the world. During service execution, this role is also responsible to check the goal life-cycle (active, addressed, failed).

In terms of governance, the goal handler and each service broker and state monitor are simple workers, but at the moment of the holon formation the head of the holon is elected according to a mechanism of trust and reputation (that is out the scope of this paper). The head has three supplementary responsibilities:

1) it is responsible to represent the whole holon to the outside (other holons), thus if service-brokers and state monitors have been selected for a set of capabilities $\delta_1, \delta_2, \ldots$ then the head offers to the outside the composition of these capabilities (namely a *task*) denoted as $\Delta = \langle \delta_1, \delta_2, \ldots \rangle$);
2) it maintains the current state of the world, obtained through the perception of all the sub-holons and it checks for the integrity of the holon structure (verifying all sub-holons are active);
3) even if each worker maintains its autonomy, the head influences their activity i) by guaranteeing their coordination and synchronization, or ii) by deciding for the re-organization of the holon structure as a consequence of failures or unexpected events.

IV. Proactive Means-End Reasoning

This section illustrates dynamic aspects of the formation of holons according to the structural rules specified in Section III-A. The key for dynamically generating holons is what we call Proactive Means-End Reasoning [29], a distributed procedure that allows agents to autonomously decide how to combine their available capabilities and therefore how to generate holons.

A. Goal Model Decomposition and Holon Formation

Section II-A has introduced the language for goal injection. However a goal is rarely injected into the system as an isolated entity. More frequently the user will use more goals to specify its request: a goal model, i.e. a set of correlated goals to be injected at the same time.

Given a goal model (G, R) where $g_{root} \in G$ is the top goal of the hierarchy, the *Goal Model Decomposition* algorithm explores the hierarchy, starting from g_{root} in a top-down fashion. The objective is to trigger the formation of one or more holons able to address the root goal. The algorithm is recursive and it exploits AND/OR decomposition relationships to deduct a goal addressability by observing its sub-goals.

We used δ_j and $\Delta_k = \langle . \rangle$ to respectively denote a single capability and a task. A task is generally provided by a holon. We also introduce $\{.\}$ to indicate a (complete or partial) solution for addressing a generic goal g_i where the 'dot' is a placeholder for a list of tasks that can be alternatively used for the fulfillment of g_i.

An example of solution for a goal has the following form: $\{\langle \delta_1, \delta_2, \ldots, \delta_n \rangle, \langle \overline{\delta_1}, \overline{\delta_2}, \ldots, \overline{\delta_m} \rangle\}$ or the more compact: $\{\Delta_1, \Delta_2\}$ where $\Delta_1 = \langle \delta_1, \delta_2, \ldots, \delta_n \rangle$ and $\Delta_2 = \langle \overline{\delta_1}, \overline{\delta_2}, \ldots, \overline{\delta_m} \rangle$. To address the goal the system can alternatively execute Δ_1 or Δ_2.

Figure 5 illustrates how the three roles interact. Once a goal g_i is injected, the goal-handler checks whether the goal is a leaf goal or not. If it is not then the goal-handler proceeds with a top-down recursive decomposition. For

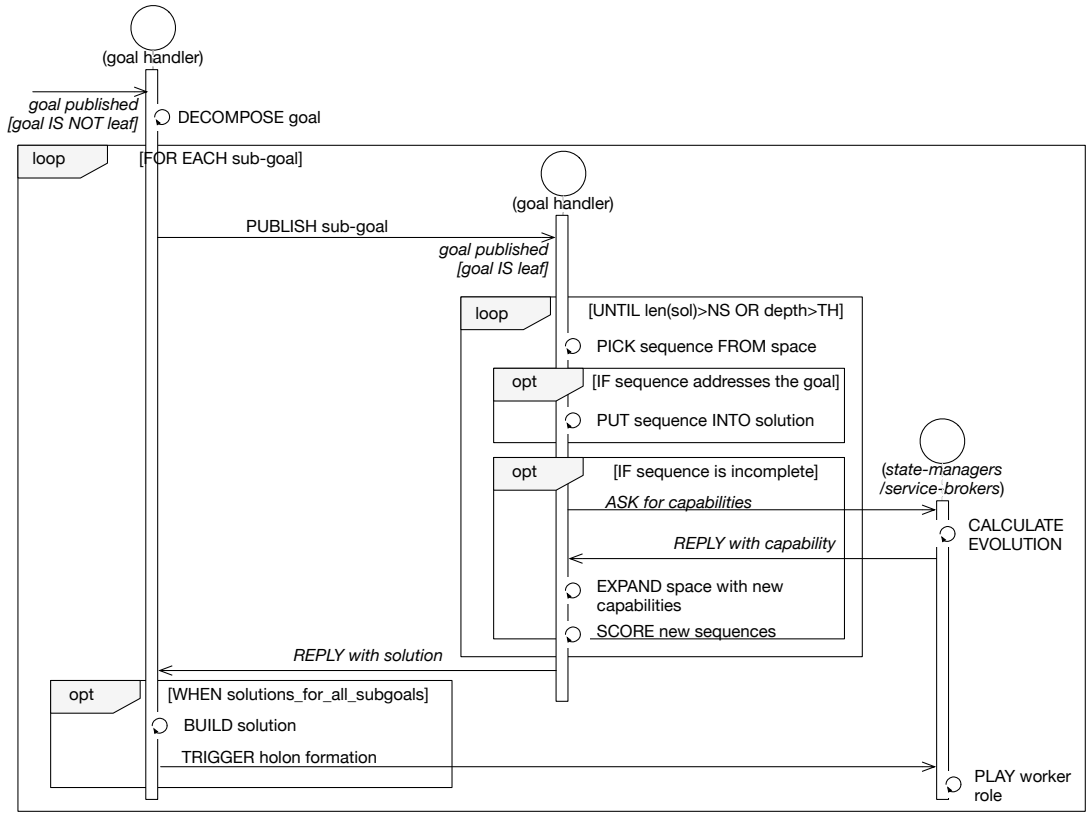


Figure 5. Sequence diagrams for goal model decomposition, goal/capability deliberation and holon formation.

each sub-goal, the goal-handler injects again it, marking the relationship to its parent. After that, the goal-handler waits for solutions for all the sub-goals, and then it organizes the composed solution for g_i by triggering the corresponding holon formation.

- If the relationship is an AND decomposition the result is the permutation of all the solutions found for each child node. If g_A is AND-decomposed in two sub-goals g_B and g_C, where $\{\Delta_1, \Delta_2\}$ is the solution of g_B and $\{\Delta_3\}$ is the solution of g_C, then the solution of g_A is $\{\langle\Delta_1, \Delta_3\rangle, \langle\Delta_2, \Delta_3\rangle\}$.
- If the relationship is an OR decomposition the result is the union of all the solutions found for each child node. If g_A is OR-decomposed in two sub-goals g_B and g_C, where $\{\Delta_1, \Delta_2\}$ is the solution of g_B and $\{\Delta_3\}$ is the solution of g_C, then the solution of g_A is $\{\Delta_1, \Delta_2, \Delta_3\}$.

Otherwise, if the goal is a leaf node of the goal model, then the Goal/Capability Deliberation sub-procedure is called.

B. Goal/Capability Deliberation

The Goal/Capability Deliberation algorithm fronts the problem of combining more available capabilities from state-monitors and service-brokers for addressing a goal: given a initial state of the world, a set of capabilities and a goal, the problem is to discover a set of capabilities which composition may address the goal.

MUSA currently adopts an approach based on a search algorithm that simulates the formation of a holon and therefore the execution of various combinations of agent capabilities (see Figure 6). In this phase the service-broker adopts the *calculate evolution* protocol of the capabilities it owns. This protocol allows the agent to customize the capability in order to establish if it can be used for the specific injected goal. For instance the flight_booking capability is customized by searching a flight from a departure place to a destination in a given date/time (see Section II-B). When a complete solution is discovered the algorithm may still continue to search other solutions. It stops after the number of solutions is greater than NS (a system constant) or when time exceeds a threshold and returns all the discovered solutions.

Exploring all possible solutions for addressing a user-goal is a NP-complete problem. The complexity is reduced by putting some constraints during the exploration of the space of solutions. Colored zones of Figure 6 represent invalid solutions that can be discarded. In addition, considering pre/post conditions, only a limited number of capabilities can be exploited at each step of the algorithm, making the execution more scalable and affordable.

The space exploration algorithm compares partial solutions thus to explore firstly the most promising ones, according to the number of sub-goals that are already fulfilled and to a set of domain metrics indicating the global quality of service. The user can specify which domain metrics to consider, for instance the maximum budget or the kind of transportation to use.

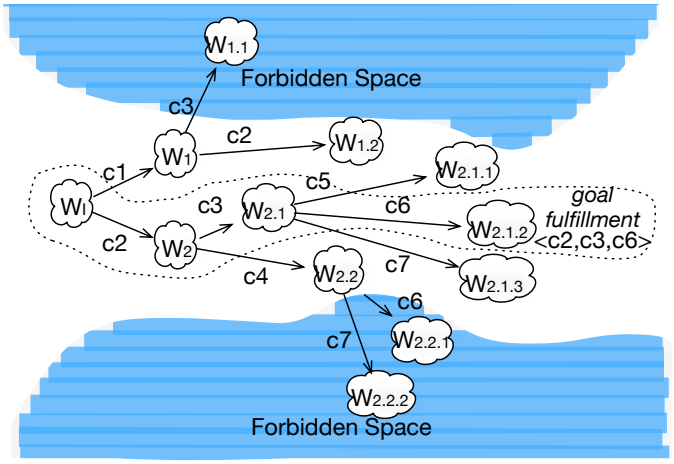


Figure 6. Abstract representation of the strategy used to explore a space of solutions for building a plan.

For a formal description of the Proactive Means-End Reasoning details are provided in [29].

Going back to Figure 5, when one (or more) solutions exist for the root goal of the injected goal model, then one or more holons have been formed for addressing the goal model. One is selected for starting the activities and eventually addressing the goals. The selection of a plan may follow many criteria. For instance in the domain of the travel, the criterion we adopted is the total cost of the travel.

C. The Self-Adaptation Loop

So far the following assumptions holds: i) services are delivered over the internet by service providers; as usual, these are accessible through standards protocols (i.e. WSDL and SOAP); ii) the system is a distributed and decentralized software, made of a number of autonomous agents, each able to perceive the environment and to act as a broker for some web-services (of which it knows description, end points and business logic); and iii) holons are agents or (temporary) group of agents: each with its own individuality.

Self-adaptiveness is the ability of the whole multi-agent system to dynamically adapt its behavior to the execution context. This is done by each individual agent, through the dynamical execution of its own capabilities, according to a shared plan and to contingent perceptions.

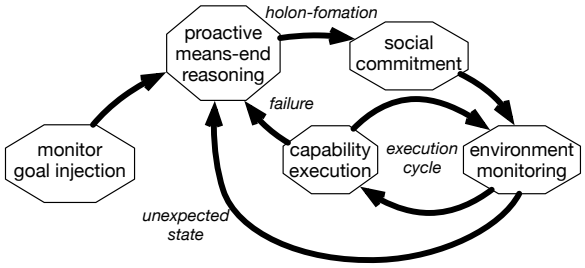


Figure 7. Graphical representation of the Self-Adaptation Cycle.

Orchestration. When a holon is selected for addressing g_{root}, it becomes operative in order to orchestrate all the embedded services and producing the compounded result (see social commitment in Figure 7).

This corresponds to activate its monitoring capabilities and to execute the *dispose service* protocols of service capabilities. State-monitors will be active for the whole real service execution.

In the domain of the *Travel Services*, the service execution has the duration of the user's travel. Service broker will book all necessary flights, hotels, ticket included in the travel plan. The monitor agent, running in the user's mobile device, acquires the user's position, and it may also be used for changing the travel plan at run-time.

Self-Adaptation. Thanks to the work of state-monitors, the head of the holon is continuously updated about the state of the services and it is able to discover when something is going wrong by comparing perceptions with expected states of the world. When a service fails, or when a goal can not be addressed then the head role of the corresponding sub-holon raises an event of failure, that is cause of an adaptation.

In the running example the adaptation is triggered by the user who modified her goals (by informing the system to change the travel plan).

The adaptation is treated by the system as a reorganization of the holonic architecture. The reorganization produces a temporary dis-assembly of the holon and the re-execution of the Proactive Means-End Reasoning procedures but considering the new situation (failures, service availability or new user goals) for generating a new solution.

Before starting the new solution, each holon executes the *compensate* protocols of the capabilities that are no more in the new plan. After that the complete service orchestration activity starts again.

V. Conclusions

Holonic multi-agent systems provide a flexible and reconfigurable architecture to accommodate environment changes and user customization. MUSA is a middleware where the autonomous and proactive collaboration of autonomous agents allows a dynamic (re-)organization of the behavior in order to address user-requests and/or unexpected events. The novelty is that the desired service composition is encapsulated in run-time goal-models that, when injected into the system, trigger a spontaneous emergence of new holarchies devoted to orchestrate the required services.

MUSA have been employed for i) executing dynamic workflows in small/medium enterprises (IDS Project[2]), ii) automatic mash-up of cloud applications (OCCP Project[3], iii) merging protocols for emergency (SIGMA Project[4]).

Authors are already working on 1) a more efficient algorithm for the means-end reasoning, 2) extending the goal language for including uncertainty and norms, 3) an automatic learning approach more robust to ontological discrepancies or language incoherence.

VI. Acknowledgment

The research was funded by the Autonomous Region of Sicily, Project OCCP (Open Cloud Computing Platform), within the Regional Operative Plans (PO-FESR) of the EU Community.

[2]Innovative Document Sharing: research project funded by the Autonomous Region of Sicily within the Regional Operative Plans (PO-FESR) of the EU Community; 2011-2014.

[3]Open Cloud Computing Platform: research project funded by the Autonomous Region of Sicily within the PO-FESR initiative; 2014-2015 - website: http://aose.pa.icar.cnr.it/OCCP.

[4]Research project for Integrated cloud system of sensors for advanced multi-risk management; 2013-2015.

References

[1] M. Pistore, P. Traverso, M. Paolucci, and M. Wagner, "From software services to a future internet of services." in *Future Internet Assembly*, 2009, pp. 183–192.

[2] S. Staab, W. Van der Aalst, V. R. Benjamins, A. Sheth, J. A. Miller, C. Bussler, A. Maedche, D. Fensel, and D. Gannon, "Web services: been there, done that?" *Intelligent Systems, IEEE*, vol. 18, no. 1, pp. 72–85, 2003.

[3] J. L. Zhao, M. Tanniru, and L.-J. Zhang, "Services computing as the foundation of enterprise agility: Overview of recent advances and introduction to the special issue," *Information Systems Frontiers*, vol. 9, no. 1, pp. 1–8, 2007.

[4] L. Sabatucci, C. Lodato, S. Lopes, and M. Cossentino, "Towards self-adaptation and evolution in business process." in *AIBP@ AI* IA*. Citeseer, 2013, pp. 1–10.

[5] R. Bordini, J. Hübner, and M. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*. Wiley-Interscience, 2007, vol. 8.

[6] A. Ricci, M. Piunti, M. Viroli, and A. Omicini, "Environment programming in cartago," in *Multi-Agent Programming:*. Springer, 2009, pp. 259–288.

[7] A. S. Rao, "Agentspeak (l): Bdi agents speak out in a logical computable language," in *Agents Breaking Away*. Springer, 1996, pp. 42–55.

[8] M. E. Bratman, D. J. Israel, and M. E. Pollack, "Plans and resource-bounded practical reasoning," *Computational intelligence*, vol. 4, no. 3, pp. 349–355, 1988.

[9] A. S. Rao, M. P. Georgeff *et al.*, "Bdi agents: From theory to practice." in *ICMAS*, vol. 95, 1995, pp. 312–319.

[10] G. S. Blair, G. Coulson, L. Blair, H. Duran-Limon, P. Grace, R. Moreira, and N. Parlavantzas, "Reflection, self-awareness and self-healing in openorb," in *Proceedings of the first workshop on Self-healing systems*. ACM, 2002, pp. 9–14.

[11] B. Schmerl and D. Garlan, "Exploiting architectural design knowledge to support self-repairing systems," in *Proceedings of the 14th international conference on Software engineering and knowledge engineering*. ACM, 2002, pp. 241–248.

[12] L. Sabatucci, M. Cossentino, C. Lodato, S. Lopes, and V. Seidita, "A possible approach for implementing self-awareness in jason." in *EUMAS*. Citeseer, 2013, pp. 68–81.

[13] L. Sabatucci, P. Ribino, C. Lodato, S. Lopes, and M. Cossentino, "Goalspec: A goal specification language supporting adaptivity and evolution," in *Engineering Multi-Agent Systems*. Springer, 2013, pp. 235–254.

[14] J. Whittle, P. Sawyer, N. Bencomo, B. H. Cheng, and J.-M. Bruel, "Relax: Incorporating uncertainty into the specification of self-adaptive systems," in *Requirements Engineering Conference, 2009. RE'09. 17th IEEE International*. IEEE, 2009, pp. 79–88.

[15] R. C. Moore, "Reasoning about knowledge and action," Ph.D. dissertation, Massachusetts Institute of Technology, 1979.

[16] Y. Lesperance, "A formal account of self-knowledge and action." in *IJCAI*. Citeseer, 1989, pp. 868–874.

[17] M. J. Wooldridge, *Reasoning about rational agents*. MIT press, 2000.

[18] L. Padgham and P. Lambrix, "Formalisations of capabilities for bdi-agents," *Autonomous Agents and Multi-Agent Systems*, vol. 10, no. 3, pp. 249–271, 2005.

[19] K. Sycara, S. Widoff, M. Klusch, and J. Lu, "Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace," *Autonomous agents and multi-agent systems*, vol. 5, no. 2, pp. 173–203, 2002.

[20] M. Cossentino, N. Gaud, V. Hilaire, S. Galland, and A. Koukam, "Aspecs: an agent-oriented software process for engineering complex systems," *Autonomous Agents and Multi-Agent Systems*, vol. 20, no. 2, pp. 260–304, 2010.

[21] M. Saeki, "Semantic requirements engineering," in *Intentional Perspectives on Information Systems Engineering*. Springer, 2010, pp. 67–82.

[22] P. Ribino, M. Cossentino, C. Lodato, S. Lopes, L. Sabatucci, and V. Seidita, "Ontology and goal model in designing bdi multi-agent systems." *WOA@ AI* IA*, vol. 1099, pp. 66–72, 2013.

[23] N. Guarino, M. Carrara, and P. Giaretta, "Formalizing ontological commitment," in *AAAI*, vol. 94, 1994, pp. 560–567.

[24] M. Cossentino, D. Dalle Nogare, R. Giancarlo, C. Lodato, S. Lopes, P. Ribino, L. Sabatucci, and V. Seidita, "Gimt: A tool for ontology and goal modeling in bdi multi-agent design," in *Workshop" Dagli Oggetti agli Agenti"*, 2014.

[25] A. Koestler, "The ghost in the machine. 1967," *London: Hutchinson*, 1967.

[26] J. J. Kay and M. Boyle, *Self-organizing, holarchic, open systems (SOHOs)*. Columbia University Press: New York, NY, USA, 2008.

[27] J. Ferber, *Multi-agent systems: an introduction to distributed artificial intelligence*. Addison-Wesley Reading, 1999, vol. 1.

[28] C. Hahn and K. Fischer, "Service composition in holonic multiagent systems: Model-driven choreography and orchestration," in *Holonic and Multi-Agent Systems for Manufacturing*. Springer, 2007, pp. 47–58.

[29] L. Sabatucci and M. Cossentino, "From Means-End Analysis to Proactive Means-End Reasoning," in *Proceedings of 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, May 18-19 2015, Florence, Italy.

An application of learning agents to smart energy domains

Alba Amato, Marco Scialdone, Salvatore Venticinque
Department of Industrial and Information Engineering
Second University of Naples - Aversa, Italy
Email: {alba.amato,marco.scialdone,salvatore.venticinque}@unina2.it

***Abstract*—The main requirement for building an Internet of Things is the definition of smart objects in which it needs to put intelligence. The pervasive deployment of smart objects will add value to applications by capabilities of communication, negotiation, learning and distributed reasoning. In this paper we investigate how the paradigm shift from objects to agents is the driver for developing these capabilities by a case study in the context of Smart Energy application domain. In fact the paradigm shift we are seeing in these years is to consider the electricity network like an Internet of Energy, where each and every electrical device and generator will be connected in a network and able to communicate data and receive and react in real time to events and stimuli that arrive from other devices or from the grid: a scattered network of sensors, actuators, communication nodes, systems control and monitoring. Here we present the learning-based approach for power management in smart grids providing an agent-oriented modeling of the energy market. The main issue we focus on is a reasonable compromise between the resolution of the consuming profile representation and the performance and real time requirements of the system.**

***Index Terms*—Intelligent Agents; Smart Grid; Learning**

I. INTRODUCTION

The Internet of Things (IoT) aims at controlling the physical world from a distance but it requires integration and collaboration of different technologies in wireless and wired networks, heterogeneous device platforms and application-specific software. The IoT refers to a globally connected, highly dynamic and interactive network of physical and virtual devices [1].

Similarly Object-oriented programming (OOP), is a programming paradigm based on the concept of objects, which abstracts entities in terms of status and provided services by attributes and methods. By a bottom up approach abstract entities (objects) are identified as parts of the system. Their properties and the interrelationships between computer programs are designed by making them out of objects that interact with one another.

Despite of these similarities OOP just breaks software and information into functional units. This programming paradigm does not deal with providing smartness to objects for enabling intelligent interaction models [2]. The building block of the IoT is the smart object and the novelty is the pervasive deployment of such embedded systems connected to the Internet, interacting with one another. In fact smart objects are mostly fully functional on their own, but value is added by capabilities of communication, negotiation, learning and distributed reasoning.

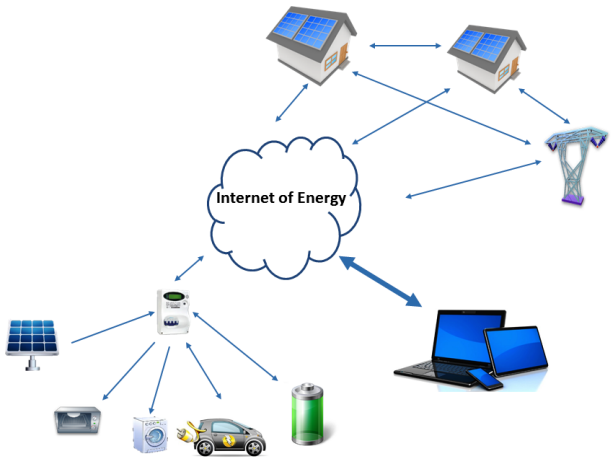


Fig. 1. Internet of Energy Overview

The rapidly growing trend of introducing computing capabilities into everyday objects and places allow us to investigate how the paradigm shifts from objects to agents is the driver for exploiting these capabilities by a case study in the context of Smart Energy application domain.

We are seeing in these years a new generation of electricity network, which behaves like an Internet of Energy, where each and every electrical device and generator will be connected in a network and able to communicate data and receive and react in real time to events and stimuli that arrive from other devices or from the grid: a scattered network of sensors, actuators, communication nodes, systems control and monitoring. In fact the power grid today is no longer designed as a simple network that delivers energy according to one direction: from few large power generation to many small consumption points at the end users.

In particular we address the problem of exploiting the increasing availability of distributed renewable energy sources such as photo-voltaic (PV) panels to improve the energy efficiency in a neighborhood.

In fact, the increasing decentralized production of green energy by photo-voltaic panels has changed the classical model of power supply from the grid to the households.

Nowadays each house becomes a micro-grid connected to the network where energy producers and consumer exchange energy between themselves and with the grid as shown in Figure 1.

As we all know, the electricity demand varies during the day and over the seasons. The electrical current is designed to withstand the maximum level of demand, and then in the absence of constant peaks turns out to be over-sized and underutilized.

The main challenge here is to provide intelligence to consuming and producing devices like photo-voltaic panels, energy storages, appliances, sensors and actuators to let them collaborate in order to agree on the best schedule of energy consumptions according to monitoring information and predictions.

The object oriented paradigm appears to be very useful to represent and manage this type of problem but it is not enough because we need to provide the devices with all advanced capabilities to collaborate as intelligent actors of the IoT.

Intelligent agents will act on behalf of devices learning energy requirements and predicting energy availability. The energy profiles will be then used to negotiate energy exchanges according to which the best global schedule, within a neighborhood, will be built.

In this paper we present the learning-based approach for the distributed energy market. The main issue we focus on is a reasonable compromise between the resolution of the profile representation and the performance and real time requirements of the system. We discuss some experimental results obtained running a prototype implementation.

II. RELATED WORK

Several studies have been conducted regarding the application of multi-agent system in the energy management and negotiation. Paper [3] describes an application of MAS for management of distributed energy resources. Through a software simulation authors demonstrate that is possible to apply a distributed coordination approach to coordinate distributed energy systems. In [4] and [5] a MAS, developed in JADE, is presented for generation scheduling of a micro-grid. The architecture provides several types of agents. For example there is the controller agent that is associated to each device that produces energy such as photo-voltaic panel or wind turbine; load controller agent represents corresponding controllable load in the system. Other several studies have been conducted regarding modeling of the loads. Paper [6] presents a new methodology for modeling common electrical loads. Authors derive their methodology empirically by collecting data from a large variety of loads and showing the significant commonalities between them. A large variety of statistical and artificial intelligence techniques have been developed for short-term load forecasting [7]. Some typical approaches are:

- *Similar-load approach*. This approach is based on searching historical data about loads to identify similar characteristics to predict the next load. The forecast can be a linear combination or regression procedure that can include several similar loads.
- *Regression methods*. Regression is the one of most widely used statistical techniques. For electric load forecasting regression methods are usually used to model the relationship of load consumption and other factors such as weather, day type, and customer class.
- *Time series*. Time series methods are based on the assumption that the data have an internal structure, such as autocorrelation, trend, or seasonal variation. Time series forecasting methods detect and explore such a structure.

In this paper we use a similar-load approach for short-term learning.

Our contribution, and in particular the CoSSMic project [8], is going beyond the state of art by using a distributed negotiation among users' devices on real power grids, that to the authors' knowledge has not been implemented before. The framework will be validated on real infrastructures by trials that involve inhabitants of two different European countries (Germany and Italy). Both software and hardware will be integrated and customized ad hoc to be compliant with existing installations. In [9] we presented a Multi Agent System (MAS) for the deployment of producer and consumer agents that will participate in the energy distribution. We defined a virtual Market that supports the energy negotiation based on XMPP[1] protocol. Agents can make calls for proposals, accept offers and negotiate with other agents. Additional details about how network of agents have been exploited in other applications domains are described in [10].

Load forecasting has always been important for planning and operational decision conducted by utility companies[7], with the deregulation of the energy industries, it will be even more important. However it is critical to predict the evolution of the load demand because it depends on human activity and changes over time with cycles that are daily, weekly, seasonal.

We focus here on a learning approach for short-term load forecasting to estimate load flows and to make decisions about task scheduling. The learning approach means each consuming device behave like an adaptive, intelligent agent, gradually adapting to its environment, and gaining more confidence in its predictions. This has the advantage that great part of the smart-grid configuration is leveraged by agents. The big disadvantage is that the agent has to learn from scratch, which means it might take a some time before the agent can make accurate decisions.

III. THE COSSMIC PROJECT

CoSSMic (Collaborating Smart Solar-powered Micro-grids. FP7-SMART CITIES, 2013) is an ICT European project that aims at fostering a higher rate of self-consumption of decentralised renewable energy production by innovative autonomic systems for the management and control of power micro-grids on users' behalf. This will allow households to optimise consumption and power sales to the network by

[1]Extensible Messaging and Presence Protocol

a collaborative strategy within a neighborhood. In fact the increasing of the decentralized production of green energy is affecting the current energy management scheme both at the grid and at the user level. The amount of electricity used and the energy produced by photovoltaic systems varies over the course of the day, from season to season and depending on the weather conditions. On the other hand the users may not have the opportunity to turn on their appliances when the PV is producing because they do not know when or because they are outside or simply because they do not need those appliances at that time. For this reason there is not an alignment during the time between production and consumption. At the user level this limits both the optimal utilization of green energy and affects the budget spent.

The CoSSMic project proposes an intelligent scheduling of user's consumptions within a micro-grid. A micro-grid is typically confined to a smart home or an office building, and embeds local generation and storage of solar power, and a number of power consuming devices. The main goal is to enable the collaboration among households so that the energy produced by the PVs is consumed by any available consumer in the neighborhood. An autonomic system will able to shift the loads in each neighborhood, according to users' constraints and preferences, in order to find an optimal match between consumption and production during the day, so that the use of renewable energy is maximized. Appliances (refrigerators, washing machine, drier and dishwasher, water heaters, air conditioners) equipped with intelligent controllers are represented by software agents negotiating energy exchanges according to availability, demand, user's preferences and constraints. The energy negotiation between agents will be based on rewards for local producers. To obtain the maximization of self consumption, a predictive approach can be considered, but we need to predict PV production based on the weather forecast and the parameters of the plant. Moreover the power consumed by each device must be known in advance or learned from monitoring information. Also the behaviour of consuming devices can change each day according to a number of parameters (e.g. the external temperature for the refrigerator or the air conditioner, the user's needs for the electric cars or for the oven), which are often unpredictable.

IV. Overview of the CoSSMic solution

Figure 2 shows the main components of our architecture playing the CoSSMic scenario. A detailed description is provided in [9].

The *Graphical User Interface (GUI)* supports interactive control and configuration of the system. It allows to plan the usage of appliances defining constraints and preferences. It also provides to the user real time monitoring information, statistics on historical data and predictions.

Mediator Services are used for storing and management of smart grid information. Mediator services are accessed by device drivers to store measures and by agents to collect information about energy production and consumption of the household. The same services are used to save data about the

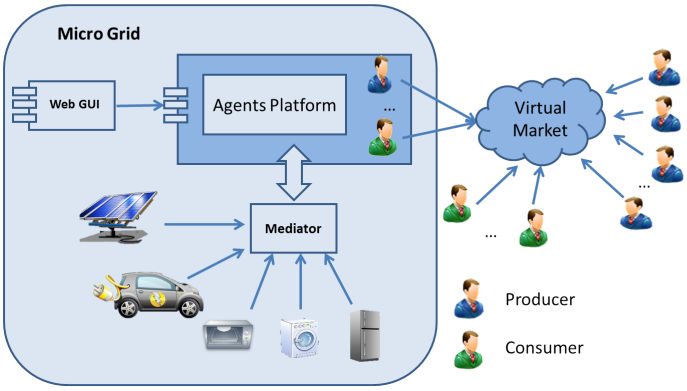


Fig. 2. CoSSMic Platform - Components View

schedule of local devices. In fact the allocation of energy to devices is modeled as task schedule with energy and time constraints.

The algorithm to find the best scheduling of the consuming appliances is designed and implemented as a distributed negotiation among software agents.

Agents are classified according to two categories:

- Consumers: they buy energy for consuming devices. E.g. they will run in houses to manage objects that absorb energy: electric car, computers, ovens, washing machines, etc.
- Producers: they can sell energy. In this category there are, for example, power generators, solar panels, wind turbines.

Those devices, which are able both to produce and consume energy such as energy storages or electric cars, will be represented by a couple of agents belonging to the two different classes defined above.

For the energy negotiation there is not a concrete marketplace, but a virtual market is implemented by a negotiation protocol that uses P2P overlay of agents.

The user will define preferences and constraints about the utilization of his appliances. This policy will be used by the smart devices to find the best plan that maximize the energy self-consumption of the neighborhood. Besides the user's constraint and preferences each agent should know the energy/power profile of its device.

A requirement for producer agents is the knowledge of the energy availability in the future. The prediction about how much energy will be produced by PV panels is computed using weather forecast, properties of the PV plants and historical series.

A requirement for the consumer agents is the knowledge of the consuming profile of the managed device. Of course such a profile will depend on different parameters. In the case of a washing machine the energy consumed could depend on the operation mode, on the amount of clothes, but in the case of an air conditioner or of a freezer the temperature is a relevant feature to take into account. In this case monitoring

information will be used to learn the energy profile while the CoSSMic platform is running.

Devices in the home can send information about electricity consumption through wireless interfaces (for example UHF or Zigbee). Mobile devices (e.g. electric cars), instead, send information through the CoSSMic Cloud. In both cases the information, through the Mediator, reach the agent platform. The mediator integrates also a number of device drivers, which allow to send real time commands to electric devices in the smart house. For example, through device drivers the mediator APIs allow to switch on or switch off devices, when it does not violates any constraints, in order to save energy.

V. Profile Learning

In the following subsection we focus on consuming devices. The prediction of energy production from PV panels will use a different approach and is out of the scope.

A. Energy profiles

In CoSSMic optimization of task scheduling use device profiles. Profiles include some meta-data and time series, i.e. series of time value pairs, which describe the cumulative energy consumed or produced when the device is running. Each device may have more profiles. In fact, as we said before dishwashers and washing machines typically have different operation modes, and in that case there will need one device profile for each mode.

Static or synthetic profile can be used just for the first run. However profiles may change run by run, therefore dynamic profiling is needed. In CoSSMic a learning approach is used to improve dynamically the device profile in real time. For instance, the consumption profile of a heater depends on the ambient temperature. Initial measurements for a certain ambient temperature can be used for the first run. However using monitoring information within the current environment and for the current operation mode will be exploited to update the profile dynamically and to improve the prediction for the next schedule.

B. Learning model

Before the start of the trials and until the software is not available, we used monitoring information to test learning algorithm and to evaluate the proposed approach.

In particular during the trial the user will program the device by the CoSSMic interface and making the system aware about the starting time and the operation mode on the next run. In the following example we used raw energy time-series, which have been collected using a PG&E Landys+Gyr smart meter.

The problem here is that time-series include different working modes which are different for profile and duration and it is necessary to identify the different runs. Another problem is that the samples come at a different rate, according to the energy consumed. To identify different runs of a time series of a washing machine, we split the energy time-series using a supervised approach.

Different runs are recognized by looking at sequence of values whose variation for a certain period (e.g. 10 minutes)

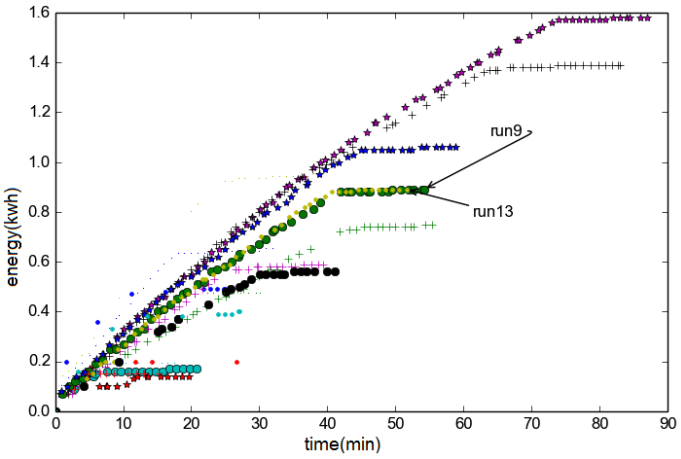


Fig. 3. Cumulative energy of different runs of a washing machine

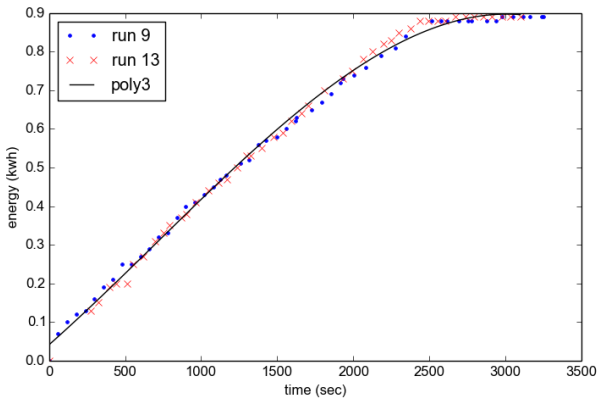


Fig. 4. 3^{th} degree polynomial representation of monitored profile

is below a threshold (e.g. 0.01 kW). In fact in this case we suppose that the washing machine is off and the run is terminated. A new energy increment above the threshold will correspond to the starting time of the next run.

In a second steps we clustered the different runs according to their duration and the amount of consumed energy to identify different operation modes. In Figure 3 the cumulative energy consumption of different runs of a washing machine are shown. The time series run9 and run13 correspond to the same operation mode.

The problem is the best approximation of the profile using two or more time-series by a compact representation that does not affect performance and real time requirements.

In Figure 4 the blue points and the red crosses are the samples from run9 and run13. The black line represent the polynomial 3^{rd} degree polynomial curve that fits the points with a minimal square root error. This representation will introduce of course a residual error, but it is monotone and needs only 5 float parameters to be represented (4 coefficients, duration and residual error), independently from the amount

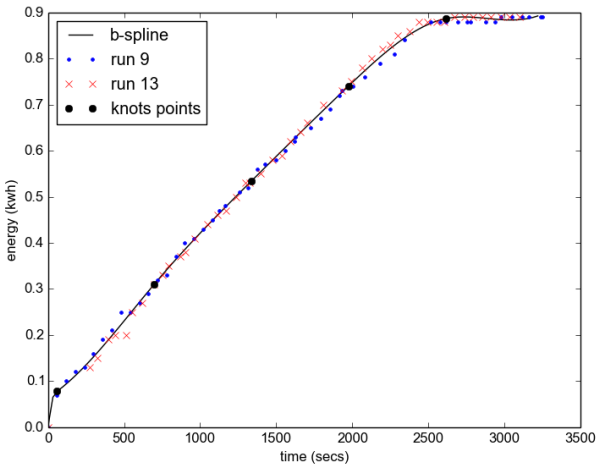


Fig. 5. B-spline representation of monitored profile

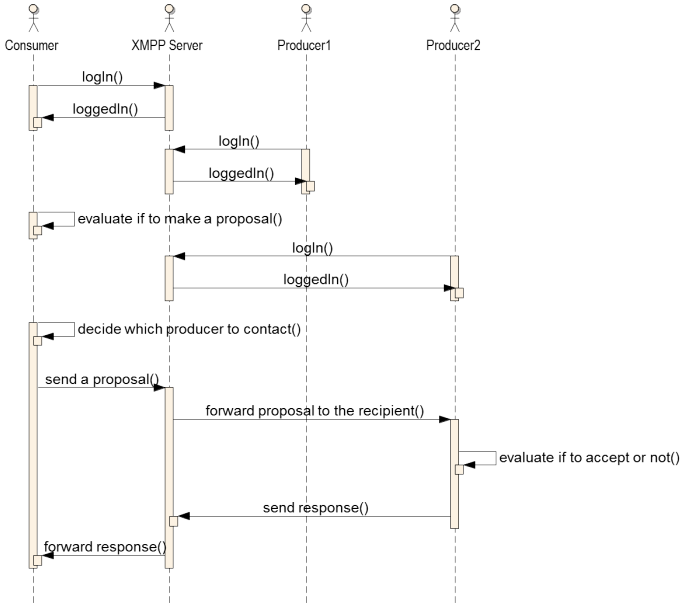


Fig. 6. Negotiation Protocol

raw samples.

In order to improve the resolution of our representation we also considered the utilization of a b-spline. The b-spline is a piecewise polynomial function of degree k. Polynomial curves meet and are continuous in a number of control points. We can control the resolution increasing the number of control points.

In Figure 5 the black line is a b-spline representation of degree k=3 of the best fitting of run9 and run13. In this case 5 control points have been fixed at regular time intervals and the number of float parameters to be transmitted is 27.

VI. SLA BASED ENERGY NEGOTIATION

The main objective of a negotiation is to reach an agreement between a vendor and a customer. A Service Level Agreement (SLA) defines an agreement between a provider and a client in the context of a particular service provision and can be between two (one-to-one) or more (one-to-many or many-to-many) parties.

In our scenario an energy consumer and an energy producer agree to shift the energy workloads of a device to optimise the mapping between production and consumption, but within a time period defined by the earliest start time and the latest start time defined by the user. For this reason the emergent behavior of whole multi agents system will implement a distributed scheduler that allocates consuming and producing tasks for each device within the neighbourhood.

The negotiation protocol will be implemented by an overlay of agents which exchange FIPA[2] messages using XMPP as transport layer. The XMPP server is used for authentication and to support communication across firewalls. Moreover many concepts of the XMPP protocol has been re-used (friendship, presence, multi-user chat, etc..). Alternative server-less solution will be investigated in future works.

[2]Foundation Intelligent Physical Agents

In Figure 6 a sequence diagram about the agents it is shown. A consumer that wants to schedule his consuming tasks will execute the following steps:

- Connects and login to the XMPP server to join the neighbourhood.
- Estimates its own need of energy.
- Brokers the producer that can offer the required energy
- Send a proposal (SLA template)

On the other hand, the producer will:

- Connect and login to the XMPP server
- Wait for incoming proposals
- Evaluate the proposal accepting or not
- Send the related response

Consumers will adopt a ranking mechanism to broker the producer to be contacted. Every consumer associates a rank to each producer, that is an integer that indicates the quality of the producer. Each time the producer accepts the proposal, this number is incremented by 1. The consumer will continue to call the producer which has the highest ranking.

Each producer will try to allocate the consuming workload according to the model presented in the previous section choosing the start time for the incoming task that satisfies the user requirements, but minimizing the use of energy from grid compared to its own production profile.

The negotiation protocol is started by a consumer agent each time a new execution is planned for the handled device sending a proposal. The message body of a proposal is a machine readable SLA template.

Templates used by agents for negotiation define the energy requirements, by the profile discussed above, including user's preferences and constraints (e.g. start date, termination date, cost, etc.). The SLA will complement the energy requirements with the negotiation parties and actual start time.

A producer can eventually withdraw an SLA if its prediction of production change and the negotiation will be restarted by the consumer.

Of course the algorithm may be affected by the problem of local minima that may bring to a not optimal employment of available energy. The final formulation of the optimization algorithm is not available yet, but it is related to a distributed approach where each agent as limited knowledge of the system. We will accept a sub-optimal solution that can be obtained with limited processing and communication resources.

VII. Experiments

After a scouting of available technologies we evaluate performances of SPADE and JADE agent platforms working with different XMPP implementations.

SPADE [11] is an open source agent platform written in Python. SPADE provides a library (SPADE Agent Library) that is a module for the Python programming language for building agents. The library contains a collection of classes, functions and tools for creating agents that can work with the SPADE Agent Platform.

JADE is an agent platform fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications. It also provides a set of graphical tools that supports the debugging and deployment phases [12]. For communication, SPADE is based on the XMPP technology. XMPP [13] is an open, XML-inspired protocol for near-real-time, extensible instant messaging (IM) and presence information. It has also been used for publish-subscribe systems, signaling for VoIP, video, file transfer, gaming, Internet of Things applications. An XMPP server provides various types of services: user account registration, authentication, channel encryption, prevention of address spoofing, message relaying, etc. Nothing prevents to deploy across the network servers that can route and relay messages for workload balancing purpose. SPADE and JADE is fully compliant with FIPA specifications. SPADE use XMPP natively as transport protocol. We developed a plugin for JADE to support XMPP as transport protocol for agent's FIPA-ACL messages .

We experimented three different XMPP server technologies:Tigase, OpenFire and the light implementation provided by SPADE itself. Tigase [14] is an open source project providing XMPP server implementation in Java. OpenFire [15] is an instant messaging and groupchat server that uses XMPP server written in Java and licensed under the Apache License.

The testbed is one host equipped with a 2.67 GHz i5 processor, 4 GB of memory amount and Windows 7 operating system. Here we evaluate the performance running a single consumer and a single producer that always accepts negotiation requests. Message exchanged have an empty payload because the purpose is to evaluate the technology.

In Figure 7 the chart shows the average time to close a transaction. It is clear that with one consumer and one producer, SPADE on OpenFire exhibits the best performance

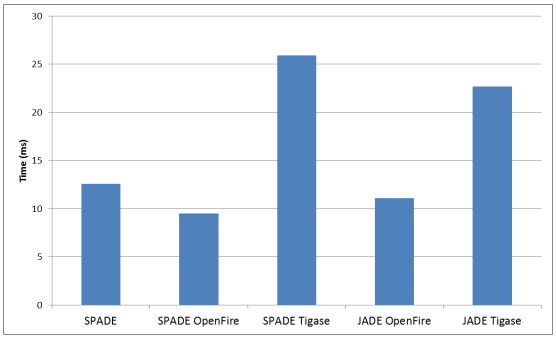


Fig. 7. Performance comparison among SPADE with embedded XMPP server and SPADE and JADE with OpenFire and with Tigase

although the difference with JADE on OpenFire and SPADE with its embedded server is really minimal.

Thanks to this experiment we can affirm that using empty messages the agent platform does not affect performance whereas the time depends on server performance.

In the second experiment we replaced the producer agents that accepted each requests without any computation overhead with a new agent that makes 1.000 floating point operations to simulate the optimization algorithm. We have one producer that always accepts requests and 50 consumers.

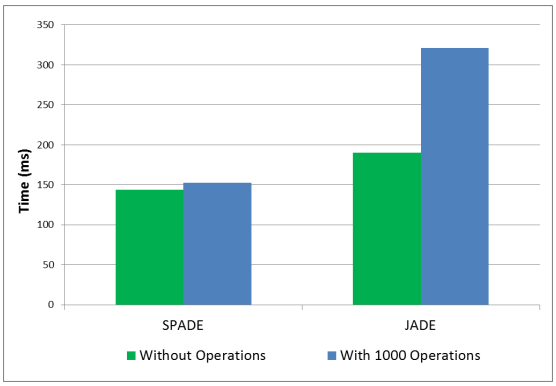


Fig. 8. Comparison between SPADE and JADE with and without operations before response

From figure 8 we noticed that in JADE the time to complete the experiment is greater than SPADE. Moreover, we can note that in SPADE the mean time to close a transaction increases slightly, but in JADE it grows faster. The reason is that Python outperforms Java. As the project trials will use Raspberry Pi to host our software, our choice is SPADE. Because of the limited amount of resources required by its XMPP server we have also decided to host on the Raspberry Pi the SPADE XMPP server. Server to server connection will be used for the communication between agents executing in different households.

In the last experiment we used two RaspberryPi as testbed. It hosts 10 consumers and 1 producer to simulate an household with 10 passive devices and 1 solar panel. Each device is

represented by one SPADE agent. Here we evaluate how the resolution of the profile will affect the performance.

Consumer agents chose randomly the producer to be contacted only the first time whereas for the next time each agent, using a learning mechanism, will contact the most reliable producer. In the experiments to implement the learning capacity we use a ranking mechanism. Each agent creates a vector with all active producers and a ranking is associated to each producer. The ranking is an integer initialized to zero incremented by one each time the corresponding producer accepts the proposal. The consumer will contact the producer that has the maximum ranking and it will continue to contact the same producer until a proposal is refused.

In order to simulate a real scenario we introduce a delay between a request and the subsequent that follows a Poisson distribution with an interarrival of 500 ms.

The request message from consumer is divided in two sections: in the first one there are metadata information, in the second one there is the energy profile.

In listing 1 there is a message example where the meaning of the fields is:

- deviceID: an ID the identify univocally a device in the household;
- EST: Earliest Start Time, the minimum time when the device can be started. The time is expressed using the Unix Epoch Time;
- LST: Latest Start Time, the maximum time when the device can be started. The time is expressed using the Unix epoch time;
- execution_type: the type of execution of that particular device;
- mode: the mode of operation of that particular device;
- taskID: the ID of task;
- dataload: three real coefficients and a value that indicated the duration.

Listing 1. Request Message Example

```
{ "metaload":
  [
    {"deviceID":"61"},
    {"EST":"62340"},
    {"execution_type":"single_run"},
    {"LST":"80340"},
    {"mode":"Delicates"},
    {"taskID":"29"}],
  "dataload":
  ["0.1223","0.2342","0.4351","1426607214"]
}
```

The real parameters represent the coefficients of a 3^{th} degree polynomial curve. For lake of space we do not show the results when the raw profile is sent, but we can see that such solution is not feasible with the available resource. The response message contains the Actual Start Time (AST) for the current run, i.e. the time when the producer can provide energy.

We observed that the mean time to obtain the response is of about 5.3 seconds with a standard deviation of 676 milliseconds in the case of 10 consumers and 1 producer for each Raspberry, using a delay between requests and adopting a ranking protocol for brokering.

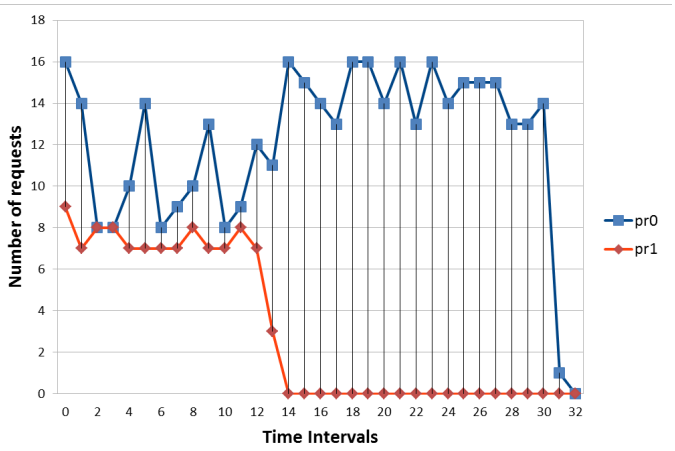


Fig. 9. Number of Requests in each time interval

The chart in Figure 9 represents the number of simultaneous requests per producer within intervals of 10 seconds. The producers manage an average of 11 requests each 10 seconds, more than 1 every second.

It can be seen that when a producer finished the energy, all the consumers move ask to the other producer. We concluded that number of requests that the system greater that the arrival rate in the real case.

VIII. Conclusion

This paper focused the exploitation of software agents as the building blocks of IoT, whose main challenge is the to design and development of application by the interaction of pervasive smart objects. We presented the approach and activities of the European Project CoSSMic that investigates the optimization of the decentralized energy production from photo-voltaic panels. We introduced the concept and the high level architectural. We focused on the design of smart devices that collaborate to find the best schedule of consumptions by a distributed negotiation. In particular a preliminary investigation about the learning model to predict the energy profiles of consuming devices and performance evaluation of the negotiation prototype have been presented.

Acknowledgments

This work has been supported by CoSSMic (Collaborating Smart Solar powered Micro grids - FP7 SMARTCITIES 2013 - Project ID: 608806).

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[2] L. Benedicenti, "Rethinking smart objects: building artificial intelligence with objects." *ACM SIGSOFT Software Engineering Notes*, vol. 25, no. 3, p. 59, 2000.

[3] M. S. Narkhede, S. Chatterji, and S. Ghosh, "Multi-agent systems (mas) controlled smart grid a review," *IJCA Proceedings on International Conference on Recent Trends in Engineering and Technology 2013*, vol. ICRTET, no. 4, pp. 12–17, May 2013.

[4] T. Logenthiran, D. Srinivasan, A. Khambadkone, and H. Aung, "Multi-agent system (mas) for short-term generation scheduling of a microgrid," in *Sustainable Energy Technologies (ICSET), 2010 IEEE International Conference on*, Dec 2010, pp. 1–6.

[5] D. Srinivasan, T. Logenthiran, A. Khambadkone, and H. Aung, "Scalable multi-agent system (mas) for operation of a microgrid in islanded mode," in *Power Electronics, Drives and Energy Systems (PEDES) 2010 Power India, 2010 Joint International Conference on*, Dec 2010, pp. 1–6.

[6] S. K. Barker, S. Kalra, D. E. Irwin, and P. J. Shenoy, "Empirical characterization and modeling of electrical loads in smart homes," in *International Green Computing Conference, IGCC 2013, Arlington, VA, USA, June 27-29, 2013, Proceedings*, 2013, pp. 1–10.

[7] E. Feinberg and D. Genethliou, "Load forecasting," in *Applied Mathematics for Restructured Electric Power Systems*, ser. Power Electronics and Power Systems, J. Chow, F. Wu, and J. Momoh, Eds. Springer US, 2005, pp. 269–285.

[8] A. Amato, R. Aversa, B. Di Martino, M. Scialdone, S. Venticinque, S. Hallsteinsen, and G. Horn, "Software agents for collaborating smart solar-powered micro-grids," in *Smart Organizations and Smart Artifacts*, ser. Lecture Notes in Information Systems and Organisation, L. Caporarello, B. Di Martino, and M. Martinez, Eds. Springer International Publishing, 2014, vol. 7, pp. 125–133.

[9] A. Amato, B. Di Martino, M. Scialdone, S. Venticinque, S. O. Hallsteinsen, and S. Jiang, "A distributed system for smart energy negotiation," in *Internet and Distributed Computing Systems - 7th International Conference, IDCS 2014, Calabria, Italy, September 22-24, 2014. Proceedings*, 2014, pp. 422–434.

[10] A. Amato, B. Di Martino, and S. Venticinque, "Semantically augmented exploitation of pervasive environments by intelligent agents," in *ISPA*. IEEE, 2012, pp. 807–814.

[11] M. E. Gregori, J. P. Cámara, and G. A. Bada, "A jabber-based multi-agent system platform," in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '06. New York, NY, USA: ACM, 2006, pp. 1282–1284.

[12] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, "Jade: a software framework for developing multi-agent applications. lessons learned," *Information and Software Technology Journal*, vol. 50, pp. 10–21, 2008.

[13] P. Saint-Andre, "Extensible messaging and presence protocol (XMPP): Core," Internet Engineering Task Force (IETF), Tech. Rep., 2011.

[14] T. Inc., "Tigase XMPP server," http://projects.tigase.org/projects/tigase-server.

[15] I. Realtime, "OpenFire XMPP server," http://www.igniterealtime.org/projects/openfire/.

Recent Possibilities of Intelligent Agents in Distributed Systems

Marcin Woźniak*
*Institute of Mathematics, Silesian University of Technology,
Kaszubska 23, 44-100 Gliwice, Poland
Email: Marcin.Wozniak@polsl.pl

***Abstract*—The article is to discuss recent advances in various aspects of intelligent agents that perform control functions in workflow management and data processing. Cloud-Computing brings various possibilities of novel approach to data management and efficient computer systems, however the process of distribution must be managed not only to increase efficiency but also lower energy consumption. This is a task for intelligent agents, that can play crucial role in modern computer science. In this article recent possibilities for these type of computer systems are discussed.**

I. Introduction

Development in computational technology sets demands in front of software engineers. The systems and solutions applied in various branches of industry must become more and more intelligent. These solutions are based on intelligent agents that have control functions over systems they manage.

In modern economy we want computers to manage efficiently enough not only to increase the income into the budgets but also to help in environment protection, what means lower energy consumption and more flexible positioning of the controlled systems. All these aspects demand intelligent technology, that will be able to optimize controlled objects. Therefore current research concern two important fields of computer science: Computational Intelligence and Software Engineering. This two, if combined, can give Agents: the intelligent software and systems.

Agents are very important for Cloud-Computing. This service is becoming more popular in the recent years, because of it construction that enables users to use software installed in machines in remote locations. One server can efficiently service many users. The idea for this type of servicing is based on economic calculations. Mainly it is applied in large corporations that have many branches placed in remote locations. Second type of Cloud-Computing is available for everybody in popular services like document sharing, social networks, multimedia streaming and more. First type has mainly economic reasons. For a large corporation it is inefficient to have similar software departments in every branch. Commonly the software is used by some workers in different shifts. Moreover if the corporation is international, time zones play an important role. While users are working with the system in one branch, somewhere else on the other continent the other team is having a break. Therefore proper configuration of a Cloud-Computing increase efficient usage of the software, decrease energy consumption what helps to protect environment and improves budget [1], [2], [3], [4].

Tailored system composition and application of managing agents can improve the overall efficiency [5], [6]. According to research conducted for the European Commission, Cloud-Computing can decrease amount of money spend on IT by 20% and drastically lower energy consumption. Cloud-Computing is usually available for users in one of three ways:

- IaaS (Infrastructure as a Service): special infrastructure with dedicated components.
- PaaS (Platform as a Service): special platform of virtual machines and operation systems.
- SaaS (Software as a Service): users get an access to applications without integrating into the system, which is the most popular model.

However mainly some compositions of the mentioned above are more practical. In this article an idea for intelligent agents managing wrokflows and knowledge distribution will be discussed.

A. Related Works

Intelligent Agents assist in processes where computer systems need proper managing. One can point various aspects of these situations, however the article is to discuss workflow management in various systems i.e. with knowledge retrieval.

Examinations of distributed systems for workflow management can be devoted to proper positioning for request management [7], [8], [9], [10]. Other aspects like intelligent software architectures [11] and efficient management [12] can also improve the overall performance. Since new technology is still introduced all the time, new possibilities that were not available before can be of concern now. Intelligent Agents help in sorting of big data [13], [14]. Application of intelligent technologies helps to increase speed of reasoning over incomplete data and data mining [15], [16]. Image processing also can be improved by advanced software [17], [18], [19], [20].

II. Intelligent Workflow Management

Request processing in Cloud-Computing systems (Fig. 1) is commonly managed by two (or more) separate agents. Each client machine sends a request to the system. These requests are ordered by first managing agent and then proceed to next managing agent. This agent composition has a role of the office, similar to offices in authority or local agencies, where each income is indexed and then send to proper department. Similarly in the agent based system, first managing server is

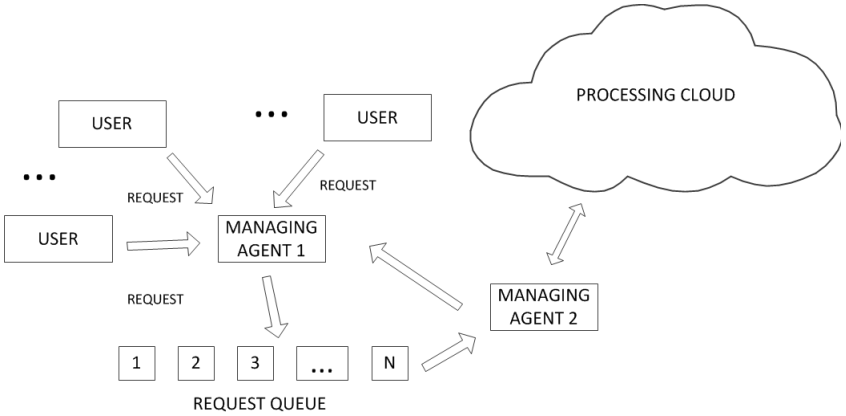


Fig. 1. A common model of requests processing in Cloud-Computing system.

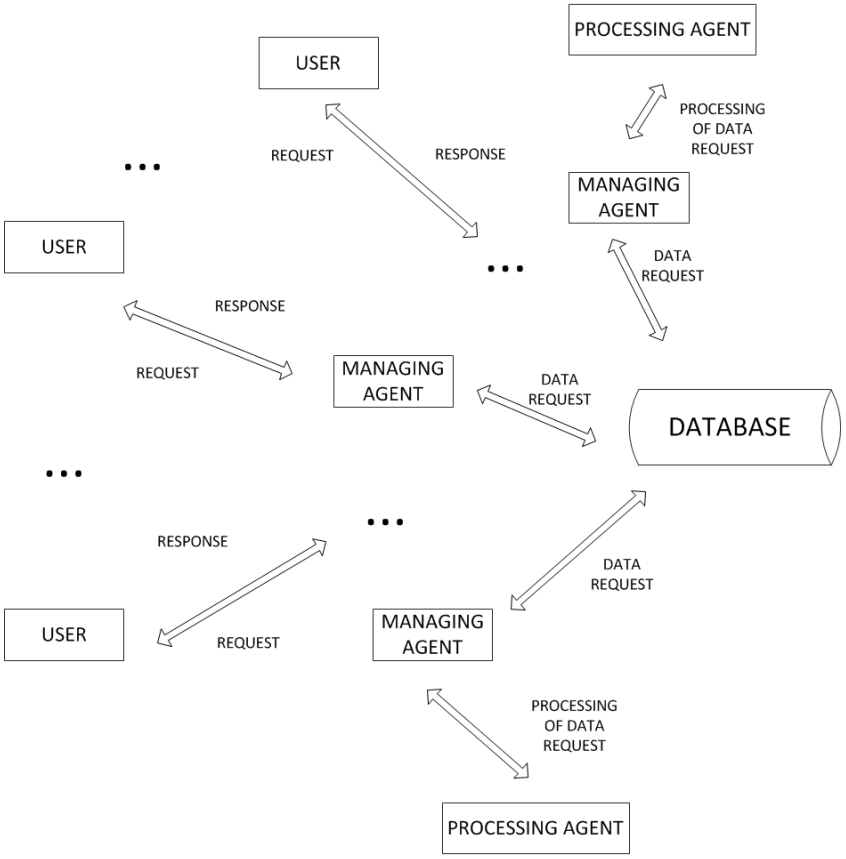


Fig. 2. A common model of knowledge processing for Cloud-Computing system.

indexing incoming requests and then sends them to proper agents responsible for different tasks. These agents proceed indexed incomes using resources or processing units connected to the cloud. These type of Cloud-Computing services is used in various verification systems, where requests go through managing agent to the processing server, which is using connection to database to assists verification. Mainly the system is also equipped in backup server, which is to store data backup. It ensures that any operation in the system is easily erased and the knowledge can be restored if any demand can happen. In Cloud-Computing systems knowledge is stored in a server that via connection with other agents is able to process information (Fig. 2). This processing depends on the requests. Mainly in each type of Cloud-Computing system there is a database (one o more), where all the information is stored. If a user requests information, a signal is addressed to one of managing agents. After confirming the rights to obtain requested data, managing unit starts to proceed. If the agent is asked to provide simple data query the operation is not complicated, from the data base is selected a portion of the data and then it is returned to the user. However if a user requested detailed information or analysis of the data, very often a new process begins. In this case managing agent is sending portion of the data for processing to a dedicated agent which is specialized in this

type of operation. In this time user request is indexed and waits for turn, meanwhile the agent can service another request from the queue. This request has similar service. Simple data query is processed ad-hoc, other is sent with data to processing agent. When processing agent has a result of analysis, it is sent to managing agent. Here according to indexes in the queue it is returned to user. All these types of processing can be modeled to optimize service costs.

A. Service model

Workflow managing is important for Quality of Service (QoS). Service description for cost optimization defines $T_{service}$, T_{income} and $T_{vacation}$, which describe average time of service, average income time and average vacation time (backup, conservation and etc.). See detailed description in [8]. Classical cost structure is considered in [21]. While in [22] and [23] are presented most important aspects of positioning and cost optimization. Various queueing models for applied type of the server are investigated in [24], [25], [26] and [27]. Please see also [28] and [29] for a review of important results on modeling and positioning.

Research on similar objects [30], [31], [32], [33], [9] with it's analytical model for traffic are applied in [8]. In the research a finite-buffer $H_2/M/1/N$-type QS, similar to server traffic modeling functions discussed in [34] and [35] was used. Let it be here presented only a brief description, for details please see [8]. Incoming requests describes 2-order distribution function:

$$F(t) = p_1\big(1 - e^{-\lambda_1 t}\big) + p_2\big(1 - e^{-\lambda_2 t}\big), \quad t > 0, \qquad (1)$$

where $\lambda_i > 0$ for $i = 1, 2$ and $p_1, p_2 \geq 0$. Inter-arrival times are mixed of two exponential distributions with parameters λ_1 and λ_2, which are being "chosen" with probabilities p_1 and p_2. In the system, there are $(N - 1)$ places in queue and one for packet in the service. System starts working at $t = 0$ with at least one packet present. After busy period the server begins vacation which is modeled with 2-order hyper exponential distribution function:

$$V(t) = q_1\big(1 - e^{-\alpha_1 t}\big) + q_2\big(1 - e^{-\alpha_2 t}\big), \quad t > 0. \qquad (2)$$

Interpretation of parameters α_i, $i = 1, 2$ and q_1, q_2 is similar to that for λ_i, $i = 1, 2$ and p_1 and p_2. If at the end of vacation there is no packet present in the system, the server is on standby and waits for first arrival to start service process. If there is at least one packet waiting for service in the buffer at the end of vacation, the service process starts immediately and new busy period begins.

Functions $F(\cdot)$ and $V(\cdot)$ help to simulate inter-arrival times and vacation defined in (1) and (2). In the research optimal values for parameters λ_i, p_i, μ and α_i are calculated to optimize amount of resources to perform all operations. This is modeled in $r_n(c_1)$ defined as:

$$r_n(c_1) = \frac{Q_n(c_1)}{\mathbf{E}_n(c_1)} = \frac{r(\tau_1)\mathbf{E}_n\tau_1 + r(\delta_1)\mathbf{E}_n\delta_1}{\mathbf{E}_n\tau_1 + \mathbf{E}_n\delta_1}, \qquad (3)$$

where the symbols are: $r(\tau_1)$-fixed unit operation costs during busy period τ_1, $r(\delta_1)$-fixed unit operation costs during idle time δ_1, $\mathbf{E}_n\tau_1$-means of busy period τ_1 and $\mathbf{E}_n\delta_1$-idle time δ_1 on condition that system starts with n packets present. In (3) are used means of busy period and vacation (idle) time. The explicit formula with detailed information and description for conditional joint characteristic functions of τ_1, δ_1 and $h(\tau_1)$ is presented in [8] and [36]. General equation to calculate these values is:

$$B_n(s, \varrho, z) = \mathbf{E}\{e^{-s\tau_1 - \varrho\delta_1} z^{h(\tau_1)} \mid X(0) = n\}, 2 \leq n \leq N, \qquad (4)$$

where $s \geq 0$, $\varrho \geq 0$ and $|z| \leq 1$, $n \geq 1$. Details on this equation are discussed in [8], [13] and [36] where using it we can define components of (3) to model total cost of work:

$$\mathbf{E}_n e^{-s\tau_1} = \mathbf{E}\{e^{-s\tau_1} \mid X(0) = n\} = B_n(s, 0, 1), \qquad (5)$$

then for model traffic finally we have:

$$\mathbf{E}_n\tau_1 = -\frac{\partial}{\partial s} B_n(s, 0, 1)\Big|_{s=0}, \qquad (6)$$

similarly we have:

$$\mathbf{E}_n\delta_1 = -\frac{\partial}{\partial \varrho} B_n(0, \varrho, 1)\Big|_{\varrho=0}. \qquad (7)$$

B. Applied Harmony Search Algorithm

Harmony Search Algorithm (HSA) was presented in 2001 by Zong Woo Geem. It's main application is to optimize and position various objects. However the first idea was to apply the HSA to compose artificial music. The method in the beginning was to perform a music by creating tones according to the rules of applied music type. The algorithm is using natural adaptation of sounds to compose tones and finally music theme. It is all based on harmony between sounds that create the music nice in theme for the people. Musician selects the sounds from the scale that best work together. HSA works analogously in the optimization. Each object variable has a specified range. When choosing the best of the possible numbers to have harmony in the process of optimization we get the solution.

HSA can be applied to optimize various objects. If we take parameters values as the sounds that we must adapt to compose a music theme we can have efficient optimization method. HSA will search the space $D = [a_1, b_1] \times \cdots \times [a_n, b_n]; f : D \to R^n$ for optimal values $a_i \leq x_i \leq b_i; i = 1, \ldots, n$ that optimize the object according to fitness condition $f(x_i) \to optimum$. In it's simplicity HSA does not demand any special restriction for $f(\cdot)$ function. It only must be possible to calculate it's value at any point of D. In the HSA we define:

- HM (Harmony Memory) - harmony memory that stores the best harmonies used for music composition:
$$HM = \begin{bmatrix} \mathbf{x_1} = (x_1^1, \ldots, x_1^n) | f(\mathbf{x_1}) \\ \cdots \\ \mathbf{x_{HMS}} = (x_{HMS}^1, \ldots, x_{HMS}^n) | f(\mathbf{x_{HMS}}) \end{bmatrix}$$
As each harmony we understand a set of values representing positioned object. If the new vector of harmony (state of the object) is better than any other among the previous HM vectors, this new vector replaces the worst one in the HM. This procedure is repeated until the stopping criterion is met,
- HMS (Harmony Memory Size) - number of harmonies stored in the memory,

- HMCR (Harmony Memory Considering Rate) - the coefficient of vector choice for memory in the range $(0, 1)$. It helps to decide whether, the new component will be selected from the memory of harmony HM, or will it be the new value of the accepted range of variation variables,
- PAR (Pitch Adjusting Rate) - the tone adjustment factor in the range $(0, 1)$.

Algorithm 1 HSA applied to position workflow traffic

1: Define coefficients: HMS, HMCR, PAR and $generations$–number of harmony search,
2: Dedicated criterion function: lowest cost of operation (3),
3: Create at random initial set HM,
4: t:=0,
5: **while** $t \leq generations$ **do**
6: with probability equal HMCR among all existing harmonies in HM $x_i^j \in \mathbf{x_I^J}$, where $I = 1, \ldots, i, \ldots, HMS$ and $J = 1, \ldots, j, \ldots, n$ compose new harmony vector $\mathbf{x_{new}}$,
7: with a probability equal to the value of PAR change $x_i^j = x_i^j + \alpha$, where $\alpha = b \cdot u$ and $b \in [0.01, 0.001]$ and $u \in [-1, 1]$,
8: with probability equal to 1 - HMCR take randomly new harmony vector $\mathbf{x'_{new}}$ variables $a_i \leq x_i \leq b_i$,
9: **while** $i \leq HMS$ **do**
10: **if** $f(\mathbf{x_{new}})$ is better then f($\mathbf{x}_*$) **then**
11: change $\mathbf{x_{new}}$ with worst harmony vector $\mathbf{x}_*$,
12: **end if**
13: **if** $f(\mathbf{x'_{new}})$ is better then f($\mathbf{x}_*$) **then**
14: change $\mathbf{x'_{new}}$ with worst harmony vector $\mathbf{x}_*$,
15: **end if**
16: **end while**
17: Sort harmonies in HM memory,
18: Next $generation$: $t++$,
19: **end while**
20: Best harmony vector in HM memory is potential optimum.

III. Research Results

Application of HSA to traffic modeling will help to position operation time and optimize service cost $r_n(c_1)$ for system under-load, critical load or overload. HSA simulations were performed for $r(\tau_1) = 0.5$ and $r(\delta_1) = 0.5$. It means that modeled workflow management is simulated for 0.5 energy unit consumption each vacation and work period. These values may be changed in (3). Presented modeling results are averaged of 100 samplings for HMS $= 50$ harmonies in 80 $generations$ with HMCR and PAR taken randomly at each generation, where times in the system have equations:

- Average service time: $T_{service} = \frac{1}{\mu}$,
- Average time between packages income into the system: $T_{income} = \frac{p_1}{\lambda_1} + \frac{p_2}{\lambda_2}$,
- Average vacation time: $T_{vacation} = \frac{q_1}{\alpha_1} + \frac{q_2}{\alpha_2}$,
- Examined system size: N = buffer size +1.

Scenario 1.
In Table I are optimum values for all parameters that affect lowest cost of system operation workflow.

TABLE I. Optimal parameters $\mu, \lambda_i, \alpha_i, p_i, q_i$ for $i = 1, 2$ and lowest value of (3).

λ_1	λ_2	α_1	α_2	p_1	p_2	q_1	q_2
3.1	2.1	1.5	0.3	1.82	1.4	5.8	3.7
μ	0.7			$r_n(c_1)$	0.32		
	$T_{service}$		T_{income}		$T_{vacation}$		
$[sec]$	1.42		1.25		16.2		

Scenario 2.
Request service is set to $T_{service} = 2[sec]$. Research results are shown in Table II.

TABLE II. Optimal parameters $\mu, \lambda_i, \alpha_i, p_i, q_i$ for $i = 1, 2$ and lowest value of (3).

λ_1	λ_2	α_1	α_2	p_1	p_2	q_1	q_2
2.4	3.7	0.6	0.5	64.56	0.91	2.6	13.20
μ	0.45			$r_n(c_1)$	0.43		
	$T_{service}$		T_{income}		$T_{vacation}$		
$[sec]$	2.22		27.14		30.73		

Scenario 3.
Positioning of heavy traffic i.e. when $T_{service} = 0.5[sec]$. Research results with system positioning are shown in Table III.

TABLE III. Optimal parameters $\mu, \lambda_i, \alpha_i, p_i, q_i$ for $i = 1, 2$ and lowest value of (3).

λ_1	λ_2	α_1	α_2	p_1	p_2	q_1	q_2
41.3	25.2	107.3	1.8	2.1	1.4	52.21	15.7
μ	2.34			$r_n(c_1)$	0.27		
	$T_{service}$		T_{income}		$T_{vacation}$		
$[sec]$	0.42		0.1		9.2		

Scenario 4.
If we position peculiar incoming traffic $T_{income} = 2[sec]$. Research results are shown in Table IV.

TABLE IV. Optimal parameters $\mu, \lambda_i, \alpha_i, p_i, q_i$ for $i = 1, 2$ and lowest value of (3).

λ_1	λ_2	α_1	α_2	p_1	p_2	q_1	q_2
3.2	4.8	1.11	1.72	5.73	2.92	14.6	6.71
μ	0.43			$r_n(c_1)$	0.34		
	$T_{service}$		T_{income}		$T_{vacation}$		
$[sec]$	2.32		2.39		17.05		

IV. Final Remarks

In Cloud-Computing or distributed systems where the amount of data to be transfered over the network is large optimal managing can significantly influence workflow and lower traffic. Presented approach to simulation and positioning can be a great advantage for distributed systems. Proposed positioning and modeling will accelerate operations and it is not burdened by typical workflow simulation restrictions.

In the article a novel approach to workflow simulation and modeling is presented. Proposed novel method is easy to implement with possibility to improve. Moreover it can be implemented in Cloud-Computing where data packages influence workflow stability and performance.

REFERENCES

[1] F. Bonanno, G. Capizzi, G. Lo Sciuto, C. Napoli, G. Pappalardo, and E. Tramontana, "A novel cloud-distributed toolbox for optimal energy dispatch management from renewables in igss by using wrnn predictors and gpu parallel solutions," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2014 International Symposium on*. IEEE, 2014, pp. 1077–1084.

[2] F. Bonanno, G. Capizzi, and C. Napoli, "Some remarks on the application of rnn and prnn for the charge-discharge simulation of advanced lithium-ions battery energy storage," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2012 International Symposium on*. IEEE, 2012, pp. 941–945.

[3] G. Capizzi, F. Bonanno, and C. Napoli, "Recurrent neural network-based control strategy for battery energy storage in generation systems with intermittent renewable energy sources," in *International conference on clean electrical power (ICCEP)*. IEEE, 2011, pp. 336–340.

[4] G. Capizzi, F. Bonanno, and C. Napoli, "A new approach for lead-acid batteries modeling by local cosine," in *Power Electronics Electrical Drives Automation and Motion (SPEEDAM), 2010 International Symposium on*. IEEE, 2010, pp. 1074–1079.

[5] C. Napoli, G. Pappalardo, E. Tramontana, and G. Zappalà, "A cloud-distributed gpu architecture for pattern identification in segmented detectors big-data surveys," *The Computer Journal*, p. bxu147, 2014.

[6] C. Napoli, G. Pappalardo, and E. Tramontana, "Improving files availability for bittorrent using a diffusion model," in *23rd IEEE International WETICE Conference*. IEEE, 2014, pp. 191–196.

[7] M. Gabryel, M. Woźniak, and R. Damaševičius, "An application of differential evolution to positioning queueing systems," *Lecture Notes in Artificial Intelligence - ICAISC'2015*, vol. 9120, pp. 379–390, 2015, DOI: 10.1007/978-3-319-19369-4_34.

[8] M. Woźniak, W. M. Kempa, M. Gabryel, and R. K. Nowicki, "A finite-buffer queue with single vacation policy - analytical study with evolutionary positioning," *International Journal of Applied Mathematics and Computer Science*, vol. 24, no. 4, pp. 887–900, 2014, DOI: 10.2478/amcs-2014-0065.

[9] M. Woźniak, "On positioning traffic in nosql database systems by the use of particle swarm algorithm," in *Proceedings of XV Workshop DAGLI OGGETTI AGLI AGENTI - WOA'2014*, vol. 1260. 25-26 September, Catania, Italy: CEUR Workshop Proceedings (CEUR-WS.org), RWTH Aachen University, 2014.

[10] M. Woźniak, "On applying cuckoo search algorithm to positioning $GI/M/1/N$ finite-buffer queue with a single vacation policy," in *Proceedings of the 12th Mexican International Conference on Artificial Intelligence - MICAI'2013*. 24-30 November, Mexico City, Mexico: IEEE, 2013, pp. 59–64, DOI: 10.1109/MICAI.2013.12.

[11] G. Borowik, M. Woźniak, A. Fornaia, R. Giunta, C. Napoli, G. Pappalardo, and E. Tramontana, "A software architecture assisting workflow executions on cloud resources," *International Journal of Electronics and Telecommunications*, vol. 61, no. 1, pp. 17–23, 2015, DOI: 10.1515/eletel-2015-0002.

[12] G. Ćwikła, A. Sekala, and M. Woźniak, "The expert system supporting design of the manufacturing information acquisition system (MIAS) for production management," *Advanced Materials Research*, vol. 1036, pp. 852–857, 2014, DOI: 10.4028/www.scientific.net/AMR.1036.852.

[13] M. Woźniak, Z. Marszałek, M. Gabryel, and R. K. Nowicki, "Modified merge sort algorithm for large scale data sets," *Lecture Notes in Artificial Intelligence - ICAISC'2013*, vol. 7895, pp. 612–622, 2013, DOI: 10.1007/978-3-642-38610-7_56.

[14] M. Woźniak and Z. Marszałek, *Extended Algorithms for Sorting Large Data Sets*. Poland: Silesian University of Technology Press, 2014.

[15] R. K. Nowicki, B. Nowak, and M. Woźniak, "Rough k nearest neighbours for classification in the case of missing input data," in *Proceedings of the 9th International Conference on Knowledge, Information and Creativity Support Systems*, G. A. Papadopoulos, Ed. 6-8 November, Limassol, Cyprus: University of Cyprus Press, 2014, pp. 196–207.

[16] B. Nowak, R. K. Nowicki, M. Woźniak, and C. Napoli, "Multi-class nearest neighbor classifier for incomplete data handling," *Lecture Notes in Artificial Intelligence - ICAISC'2015*, vol. 9119, pp. 465–476, 2015.

[17] M. Woźniak and Z. Marszałek, "An idea to apply firefly algorithm in 2D images key-points search," *Communications in Computer and Information Science - ICIST'2014*, vol. 465, pp. 312–323, 2014, DOI: 10.1007/978-3-319-11958-8_25.

[18] M. Woźniak and D. Połap, "Basic concept of cuckoo search algorithm for 2D images processing with some research results : An idea to apply cuckoo search algorithm in 2d images key-points search," in *SIGMAP 2014 - Proceedings of the 11th International Conference on Signal Processing and Multimedia Applications, Part of ICETE 2014 - 11th International Joint Conference on e-Business and Telecommunications*. 28-30 August, Vienna, Austria: SciTePress, 2014, pp. 157–164, DOI: 10.5220/0005015801570164.

[19] C. Napoli, G. Pappalardo, E. Tramontana, Z. Marszałek, D. Połap, and M. Woźniak, "Simplified firefly algorithm for 2D image key-points search," in *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - CIHLI 2014: 2014 IEEE Symposium on Computational Intelligence for Human-Like Intelligence, Proceedings*. 9-12 December, Orlando, Florida, USA: IEEE, 2014, pp. 118–125, DOI: 10.1109/CIHLI.2014.7013395.

[20] M. Woźniak, D. Połap, M. Gabryel, R. K. Nowicki, C. Napoli, and E. Tramontana, "Can we preprocess 2d images using artificial bee colony?" *Lecture Notes in Artificial Intelligence - ICAISC'2015*, vol. 9119, pp. 660–671, 2015, DOI: 10.1007/978-3-319-19324-3_59.

[21] J. Teghem, "Control of the service process in a queueing system," *Europ. Jour. of Operations Research*, vol. 1, no. 23, pp. 141–158, 1986.

[22] O. Kella, "Optimal control of the vacation scheme in an $M/G/1$ queue," *Operations Research Journal*, vol. 4, no. 38, pp. 724–728, 1990.

[23] R. Lillo, "Optimal operating policy for an $M/G/1$ exhaustive server-vacation model," *Methodology and Computing in Applied Probability*, vol. 2, no. 2, pp. 153–167, 2000.

[24] U. C. Gupta, A. D. Banik, and S. Pathak, "Complete analysis of $MAP/G/1/N$ queue with single (multiple) vacation(s) under limited service discipline," *Journal of Applied Mathematics and Stochastic Analysis*, no. 3, pp. 353–373, 2005.

[25] U. C. Gupta and K. Sikdar, "Computing queue length distributions in $MAP/G/1/N$ queue under single and multiple vacation," *Applied Mathematics and Computation*, vol. 2, no. 174, pp. 1498–1525, 2006.

[26] Z. Niu and Y. Takahashi, "A finite-capacity queue with exhaustive vacation/close-down/setup times and markovian arrival processes," *Queueing Systems*, vol. 1, no. 31, pp. 1–23, 1999.

[27] Z. Niu, T. Shu, and Y. Takahashi, "A vacation queue with setup and close-down times and batch markovian arrival processes," *Performance Evaluation Journal*, vol. 3, no. 54, pp. 225–248, 2003.

[28] H. Takagi, *Queueing Analysis, vol. 1: Vacation and Priority Systems, vol. 2. Finite Systems*. Amsterdam: North-Holland, 1993.

[29] N. Tian and Z. G. Zhang, *Vacation queueing models. Theory and applications*. Berlin, Heidelberg: Springer - Verlag, 2006.

[30] W. M. Kempa, "$GI/G/1/\infty$ batch arrival queuing system with a single exponential vacation," *Mathematical Methods of Operations Research*, vol. 1, no. 69, pp. 81–97, 2009.

[31] W. M. Kempa, "Characteristics of vacation cycle in the batch arrival queuing system with single vacations and exhaustive service," *Int. Jour. of Applied Mathematics*, vol. 4, no. 23, pp. 747–758, 2010.

[32] W. M. Kempa, "Some new results for departure process in the $M^X/G/1$ queuing system with a single vacation and exhaustive service," *Stochastic Analysis and Applications*, vol. 1, no. 28, pp. 26–43, 2009.

[33] W. M. Kempa, "On departure process in the batch arrival queue with single vacation and setup time," *Annales UMCS Informatica*, vol. 1, no. 10, pp. 93–102, 2010.

[34] D. Hongwei, Z. Dongfeng, and Z. Yifan, "Performance analysis of wireless sensor networks of serial transmission mode with vacation on fire prevention," *ICCET'10 IEEE CPS*, pp. 153–155, 2010.

[35] V. Mancuso and S. Alouf, "Analysis of power saving with continuous connectivity," *Comp. Networks*, vol. 56, no. 10, pp. 2481–2493, 2012.

[36] M. Woźniak, W. M. Kempa, M. Gabryel, R. K. Nowicki, and Z. Shao, "On applying evolutionary computation methods to optimization of vacation cycle costs in finite-buffer queue," *Lecture Notes in Artificial Intelligence - ICAISC'2014*, vol. 8467, pp. 480–491, 2014, DOI: 10.1007/978-3-319-07173-2_41.

An AO system for OO-GPU programming

Andrea Fornaia, Christian Napoli, Giuseppe Pappalardo, and Emiliano Tramontana
Department of Mathematics and Informatics
University of Catania, Viale A. Doria 6, 95125 Catania, Italy
{fornaia, napoli, pappalardo, tramontana}@dmi.unict.it

***Abstract*—Recent technologies, like general purpose computing GPU, have a major limitation consisting in the difficulties that developers face when implementing parallel code using device-oriented languages. This paper aims to assist developers by automatically producing snippets of code handling GPU-oriented tasks. Our proposed approach is based on Aspect-Oriented-Programming and generates modules in CUDA C compliant code, which are encapsulated and connected by means of JNI. By means of a set of predefined functions we separate the application code from device-dependent concerns, including device memory allocation and management. Moreover, bandwidth utilisation and cores occupancy is automatically handled in order to minimise the overhead caused by host to device communications and the computational imbalance, which often tampers with the effective speedup of a GPU parallelised code.**

Keywords*—Code generation, GPU programming, separation of concerns.

I. Introduction

For device-oriented parallel code, such as distributed high performance computing (HPC) and hardware-dependent paradigms, developers have an additional task when building an application which is taking into account the physical structure of the host (or network) [1], [2]. Developers have to consider parallelisation and communication overhead, the required bandwidth etc. [3]. Therefore, developers strive to achieve in their solutions both flexibility and high modularity. This results in increased development time and costs, sometimes with a low-performing code. Moreover, current development tools do not offer a sufficient abstraction level, instead provide a low degree of modularity to applications [4]. Furthermore, developers should take into account very complex scenarios in order to parameterise their code e.g. for different sizes for the solution domains, different number of threads and block, different data sizes, and therefore different solutions to obtain maximum core occupancy and transfer bandwidth. As a result it is difficult to separate the different concerns, overcharging the developer (and the code) with the handling of a multitude of responsibilities [5].

This paper proposes a new paradigmatic solution letting developers that use object-oriented (OO) code to develop GPU-specific code or low-level device-oriented code by means of a friendly toolbox. This toolbox uses cooperating agents to assist the development of scalable modular code that take advantage of GPU devices, freeing developers from the need to handle a device oriented language. It also provides the management of memory allocations and overall communications between host and computing device. Aspect-oriented programming [6], [7] is used as a glue to enhance an application with environment-specific choices, such as the selection of a specific task-driven code at runtime. Therefore, our proposed approach brings a substantial improvement in terms of modularity, performance, reusability of code and separation of concerns [8], [9], [10], [11], [12].

The developed toolbox provides enhanced reusability for parallel computation of previously written code (both object based or device oriented), by using several agents which interprete the behaviour of the OO code and uses a dedicated translation utility [13], [14]. Tools such as LIME [15] or AeminimumGPU [16] derive a customised language and a run-time environment, however require specific compilers and force developers to use a non-standard programming language while giving no options for standard code reusability (it is impossible for such paradigms to take advantage of reused sequential OO code and obtain parallel versions).

While OpenCL provides developers with fine-grain control of host and kernel code, the handling of low-level details is a significant overhead for the developer. The proposed toolbox, instead, requires no knowledge of the device-oriented language, instead the developer writes standard OO code, and takes care of connecting the toolbox by using some annotations.

Recent works, such as [17], have partially automated several processes in the field of code control to avoid conflicts or misleading behaviours, but even in this case it is ultimately the programmer's responsibility to structure their codes in the appropriate manner. An approach has been derived to mechanically determine how a program accesses data [18], and, other analysers have focused on extracting the structure of a software system to determine some structural properties [19], [20], [21], [22], [23], [24], [25], [26], [27]. Such analyses are paramount for assessing the possibility to transform a program in such a way to have parallelism while avoiding data inconsistency. In [28] a Java software system has been presented, based on an approach that derives an entirely new set of syntactical rules for the use of a proprietary meta-compiler. As far as our understanding, no significative further step has been made towards an high-level and self contained toolbox for easy development of GPU oriented software within an OO paradigm.

II. GPGPU and CUDA programming

GPU programmers have to consider the underlying hardware in order to write any GPU-enabled code (now on simply GPU *kernel*). Graphics processors provide a big number of simple multithreaded cores offering the potential for dramatic speedups for a variety of general purpose applications when compared to the CPU sequential computation [29], [30], [31], [32], [33].

The launch of the Nvdia™ CUDA technology has opened a new era for GPGPU computing allowing the design and implementation of parallel GPU oriented algorithms without any knowledge of OpenGL, DirectX or the graphics pipeline. A CUDA-enabled GPU is composed of several MIMD (multiple instruction multiple data) multiprocessors that contain a set of SIMD processors (single instruction single data). Each multiprocessor has a shared memory that can be accessed from each of its processors, and also shares a bigger global memory common to all the multiprocessors. Basically, a CUDA kernel makes use of threading between the SIMD processors, where a single computation is performed [3]. Moreover, the GPU card allows an advanced geometrical enumeration for threads described by a 3–dimensional structure for the 3 spatial axis (even if the z axis is actually only a logical extension) [5]. Furthermore, it is possible to collect a set of threads in logical 3–dimensional blocks that are executed on the same multiprocessor.

In CUDA programming model, an application consists of a *host* program that executes on the CPU and other parallel *kernel* programs executing on the GPU [34], [35]. A *kernel* program is executed by a set of parallel threads. The *host* program can dynamically allocate device *global* memory to the GPU and copy data to (and from) such a memory from (and to) the memory on the CPU. Moreover, the *host* program can dynamically set the number of threads that run on a *kernel* program. Threads are organised in blocks, and each block has its own *shared* memory, which can be accessed only by each thread on the same block.

It is paramount that interactions between CPU and GPU are minimised, this avoids communication bottlenecks and delays due to data transfers. Necessary data transfers should try to maximise the bandwidth usage, i.e. CPU and GPU perform as least as possible interactions and transfer a large amount of data each time.

A. Bandwidth measurements

The Bandwidth is indeed one of the most important factors for performance. The best practice in CUDA C programming recommends that almost all GPU adaptation changes to code should be made in the context of how they affect bandwidth.

Bandwidth can be dramatically affected by the choice of memory in which data is stored, how the data is laid out and the order in which it is accessed, as well as other factors due to the computation itself. In order to obtain an accurate estimate of the possible performances it is required to calculate the effective bandwidth which, generally, strongly differs from the theoretical bandwidth (the latter is much greater than the former). The theoretical maximum bandwidth B_{TH} is

$$B_{\text{TH}} = n_{\text{M}} C_{\text{M}} R_{\text{M}} \tag{1}$$

where C_{M} is the maximum memory clock, n_{M} is the number of bits of the memory interface, and R_{M} is the memory data rate (1 if single rate, 2 if double rate, etc.). Moreover, to obtain an accurate estimate of the effective bandwidth B_{EFF}, such computations should be performed at execution time by means of the following equation

$$B_{\text{EFF}} = \frac{(B_{\text{R}} + B_{\text{W}})}{t} \tag{2}$$

where B_{R} is the number of bytes read per kernel, B_{W} is the number of bytes written per kernel, and t is the time. On the other hand, B_{EFF} cannot be computed before hand, but only after observing the runtime execution. The presented solution enable us to perform these operations and to obtain a real time estimate of the bandwidth occupancy rateo

$$B_{\text{OR}} = \frac{B_{\text{EFF}}}{B_{\text{TH}}} \tag{3}$$

At runtime it could be useful to compute B_{EFF} as

$$B_{\text{EFF}} = n_{\text{O}} \left(\prod_{i=1}^{D} l_i \right) \texttt{sizeof(}\textit{TYPE}\texttt{)} \tag{4}$$

where n_{O} is the number of operations (e.g. 2 for read and write), D the maximum number of dimensions of the data structure in transfer, l_i the length along the i-esime dimension, and `sizeof(`*TYPE*`)` the dimension in bytes of one unit of data for the specified type. Therefore it follows that

$$B_{\text{OR}} = \frac{n_{\text{O}}}{n_{\text{M}} C_{\text{M}} R_{\text{M}}} \left(\prod_{i=1}^{D} l_i \right) \texttt{sizeof(}\textit{TYPE}\texttt{)} \tag{5}$$

Since we are interested in host to device transfers (and vice versa) then $n_{\text{O}} = 2$, moreover C_{M} and R_{M} depend on the hardware, and they are un-mutable during the execution. Therefore, given a fixed constant

$$K_{\text{HW}} = \frac{n_{\text{O}}}{n_{\text{M}} C_{\text{M}} R_{\text{M}}} \tag{6}$$

it follows that

$$B_{\text{OR}} = K_{\text{HW}} \left(\prod_{i=1}^{D} l_i \right) \texttt{sizeof(}\textit{TYPE}\texttt{)} \tag{7}$$

This latter gives the exact bandwidth usage.

B. Memory optimisations

While the bandwidth occupancy rateo give us an estimate of the performance of the code, in order to improve such performances a major part of the effort should be directed toward memory optimisations. A performant code should indeed maximise the bandwidth occupancy rateo, but such a bandwidth is best served by using as much fast memory and as little slow-access memory as possible (this practice applies both to the device memory and to the host memory).

In order to gain performance, it is important to reduce the number of data transfers between host and device, sometimes also by running directly on the GPU portions of serial code (or portions of code the CPU could easily outperform).

For the same reason data structures could be created both in the device and in the host in order to serve as an intermediate buffer. Such a buffer could also be useful to avoid small transfers, organising larger transfers which should perform better even in case of non-contiguous regions of memory (these would be packed in an unique compact buffer and then unpacked at their destination).

A major improvement in memory usage is finally granted by using page-locked memory, also known as *pinned* memory. By using the pinned memory the bandwidth usage should be maximised (hence, limiting to the transfers between host and device). In order to use the pinned memory the CUDA libraries provide the `cudaHostAlloc()` and `cudaHostRegister()` functions: the first allocates region of memory in pinned modality, while the latter is used to pin the memory on the fly without allocating a separate buffer.

While the use of pinned memory could improve the performance, this practice is likely to be difficult for the developers, who risk to take on too much responsibilities. Moreover, the usage of pinned memory does not give a general solution for every code since pinned memory is a rare resource and an excessive use could end up in an overall reduction of the system performances. Finally, memory paging is often a heavyweight operation when compared to normal memory management. This results in a trade-off situation which should be carefully analysed before taking any action. The proposed solution is intended to spare the developers from such concerns by taking care of this issue with automatic evaluations and countermeasures.

C. Asynchronous transfers

In a standard situation the developer may decide to transfer data between host and device by means of the `cudaMemcpy()` function, which is a blocking transfer. In other words, such an operation constitutes a barrier and returns the control to the thread only after the entire data transfer is completed.

CUDA architecture offers a different solution for memory transfer by means of the `cudaMemcpyAsync()` function which is a non-blocking variant of the previous one. This function returns control immediately with the related consequences. Moreover, this function requires the use of the pinned memory, and, for security reasons, to use the so called *streams*. A stream is a sequence of operations that are performed on the device following a certain order. Streams must be properly used while using asynchronous transfers in order to correctly access data only after they have been transferred. On the other hand different streams can be overlapped. Asynchronous transfers enable us to overlap data transfers with computations, therefore their proper use could tremendously increase performances, however they could be very tricky for the developer, and again assistance could be required. Also this kind of assistance is provided by our developed solution.

D. Cores occupancy

Another key point in order to maximise the GPU performances is the core occupancy. While a task should run unconstrained, its workload should be correctly designed so as to take advantage of a number of threads that exactly matches the number of available GPU cores. The best practice recommends to keep the multiprocessors on the device as busy as possible. It follows that a poorly balanced workload will result in suboptimal performances. Hence, it is important to implement a correct application design with an optimal task distribution on threads and blocks. The proposed toolbox aims to spare the developers from such an effort.

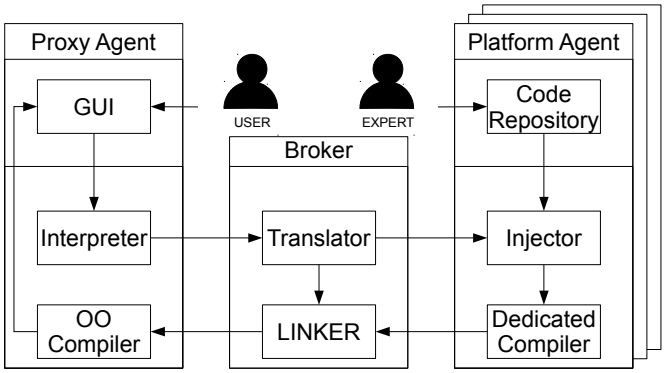


Fig. 1. An overall schema of the developed agent oriented system

III. The proposed toolbox

To strictly separate all the algorithmic development related to the application domain from different concerns (i.e. the GPU handling), which should not be taken into account by the programmer. All the GPU device-related managements are performed automatically by the proposed toolbox (e.g. the choice of the best number of threads and blocks, the needed modifications to the code in order to enable it for asynchronous stream execution, etc.).

This toolbox aims at providing a simplified and modular support for GPU computing that developers could use without having to learn how to program in CUDA. The purpose of this work is to develop such a toolbox for OO Programming to run specific tasks on the MIMD environment provided by a GPU accelerator without any need to divert from an OO paradigm and the related OO language (e.g. Java).

Figure 1 shows the proposed solution, which consists of three main agents:

- Proxy agents
- Broker
- Platform agents

The *proxy agent* provides a graphical user interface (*GUI*) which could be typically intended as a web portal to upload OO code complying with several syntactic constraints (such as the proper use of annotations). The uploaded code is then interpreted by an *interpreter* software module which creates an XML file to instruct the *translator* on the behaviour that some portions of code have to obtain. Finally, the *proxy agent* contains the OO oriented *compiler* which has the responsibility to link all the produced software modules with the unchanged portion of the original OO code and then compiles it creating an executable binary. The binary is finally returned to the user by means of the same GUI when ready.

The said *translator* module is part of the *Broker*, it receives an XML description of the behaviours from the *interpreter*. While the latter has the responsibility to detect the code behaviours and accordingly prepare an input for the translator, the interpreter has nothing to with the translation of code itself. The translator maintains a reference to the different *platform*

agents which are designed to match different hardware infrastructures. The translator then is able to understand the interpreted behaviours, and then choose the proper architecture-oriented broker agent. Once the broker has been chosen, the translator instruct it to inject some portions of translated code, on the other hand the translator itself prepares the unchanged code to be linked with the compiled device-dedicated software modules. The *linker* module has the latter responsibility, linking the remaining OO code with the generated device-dedicated executables. The latter responsibility is all but trivial, since it is up to the linker any choice regarding the best linking approach (e.g. whether to use external functions, native interfaces, or other approaches in order to permit to let the generated binary to be called from the OO portion of the code).

The effective injection of code and generation of executable software modules is performed within the dedicated *Platform agent*. The core module of the *Platform agent* is the *injector*. As the name suggest the responsibility of this module is to inject device oriented code in order to create a separately compiled software-module to be then linked by the broker, as said before. The *injector* has knowledge of a selection of codes within a *related repository*. Basing on the indications given by the *translator*, the *injector* requires the corresponding code to the *repository*. The latter is maintained by experts of the target architecture to which the repository is related. The proposed approach is similar to a *wiki* project where experts add new code to be used. Of course the uploaded code must be compliant with all the standards of the device for which it is intended, and, moreover, should be accompanied by an adeguate descriptor within the constraints of the presented agent oriented system itself. Finally the injected code si compiled with a *dedicated compiler*, a binary file is produced and passed to the linker which performs its duties closing the cycle.

In the following section we will give some details regarding the injection procedure and the involved modules.

IV. The designed modules

A. Compile time

In our approach, fragments of device-related code are automatically linked by using a predefined library of common functions of general purpose. Moreover, the designed system makes it possible to define custom CUDA compliant tasks to be executed on the GPU. At runtime, a component, which makes use of *aspect-oriented* programming, provides with the optimal management of the memory transfers and allocation by monitoring the effective allocations, initialisations, and values of the stored variables both on the host and device. By including the predefined classes or by using the precompiled executable as a linked binary, it is possible to use a predefined set of functions and also to implement custom CUDA compliant functions and then invoke them within an OO paradigm by means of Java Native Interface (JNI).

The approach provides the developer with a set of functions that, when invoked, take care of all needed management, including: the data transfers between CPU and device, the memory allocations or the use of pinned memory, the possible asynchronous execution of different threads, the optimal sizing and dimensioning with respect to best performing number of threads and blocks. Such functions are implemented as part of a set of choices made by the toolbox in order to obtain a CUDA compliant code which satisfies certain requirements that we call *behaviours*.

A behaviour is a set of fixed parameters concerning the management of the GPU card and all the related optimisation that does not involve the application logic. While the developer is responsible for the application logic, the implementation of the said *behaviour* (therefore all the choices and related implementation in terms of specific kernels, functions, parameters and strategies) are up to the presented support. Finally, the developer can also freely compose predefined behaviours or create new ones.

B. Code repository

A predefined set of behaviours and related implementing functions are included in a code repository, so that the programmer will have no need to directly implement CUDA kernels and calls, nor even to use C or C++ languages. With our approach, a developer could write his code in an OO programming language such as Java, an then use a few of the classes in our repository to provide some parameters, in order to configure the whole ensemble of application and CUDA code (examples of parameters are the number of threads, blocks, the maximum bandwidth, etc..) or let our toolbox take all the decisions.

As far as CUDA compliant code is concerned, this is implemented by the toolbox, some function pointers are predefined and enlisted so that the application developer will have the possibility to choose between a given set of functions, or manually add to the list a custom function written in CUDA C compliant code. In this way the developer could even write non-OO code (specifically CUDA code) and then use it within a more comprehensive OO application.

Any function defined in the library, hence also any custom function, is a `__device__` function: the inputs of such GPU functions consist of a unique pointer to the ensemble of data, operands and outputs; in addition to this pointer some control data are given. Note that such a code will be generated by the toolbox, which takes care of all such details. These functions are then executed on the GPU device and called by a properly generated `__global__` kernel.

The provided functions work for an arbitrary number of parameters, (i.e. operands) and functions.

C. Code injection choice

This agent oriented system was created to make use of several classes that properly realise a usable set of data assisting the computation to be performed on a GPU. These classes are adapted and interpreted as C-like defined structures of standard types, which are then transferred to the GPU device.

The JNI layer provides the needed "glue" to manage calls and data transfer towards the C++ side, which will use such classes as primitive structures. While the memory address of an object is not available under a Java framework, once objects are passed by means of JNI calls to the C++ layer, it becomes possible (within the C++ portion of code) to manipulate and

pass data by means of their memory addresses. This makes it possible to ignore the number of dimensions for arrays, matrices, tensors, handled by such data types.

An important strategy has been used to reduce the size of data transferred and consists of an a priori selection and rearrangement of the operands and functions encoded, as said before, in unique arrays.

Since the application developer has to provide the starting and ending points of the operands, before any allocation or communication in and to the device, the toolbox rearranges only the necessary part of the data in a communication buffer. Under the stated conditions the memory allocated and the data transferred to the device are minimised, on the other hand the total size of such a buffer maximises the bandwidth usage. With this selection of data another advantage is to minimise the operations on the device due to the indexing of the operands.

Another important feature of the proposed toolbox is the simplified interface to memory transfer between host and device. It is known that, as far as execution time is concerned, generally the more costly part is memory allocation and data transfer from the host to the GPU device and vice versa, which, for the best part, is at the origin of the total overhead. Memory transfer is not only expensive at runtime, indeed it is considered the ticklish and misleading part in GPU programming being also expensive in terms of coding time.

This toolbox takes care of memory allocations on the device and offers an advanced management of communications between host and GPU device. For this reason, e.g. when a variable is used twice on the GPU device during the same execution of the program, and if it is not reassigned or redeclared in the meantime, the toolbox will avoid to repeat a memory transfer, preserving a copy on the device for future use. This feature gives an easy way for the programmer to develop GPU-ready code without any need to take care of these tricking side considerations, focusing only on the algorithm she wants to implement.

D. Translator module

As said above, our toolbox uses several independent agents in order to generate device oriented software modules that can be connected with any other Java application code by using the Java Native Interface. The software modules are created by means of a common nvcc compiler without any other precompiler. In fact, it comes with the needed computational libraries which can also be precompiled and linked to an existing software system.

In order to enable application developers to produce modular code, the proposed translation system makes use of behaviours to obtain certain features of the code. Such behaviours could be intended for CUDA-compliant code in the same way as *Design Patterns* are intended for OO code. In such a context, aspect-oriented programming (AOP) has been proven effective to implement OO design patterns while preserving the independence of classes and the separation of concerns [36]. In the same way, AOP is useful in order to connect an OO application with CUDA native GPU code.

Since the proposed toolbox takes advantage of some specified parameters and several annotations given by the developer

```
public @interface GPUstream {
    int value();
}
public @interface GPUparal {
    String fixed();
}

public class Behaviour {
    private static Behaviour b = new Behaviour();
    private String status;
    private String tmpStatus;

    private Behaviour() { }

    public static Behaviour getInstance() {
        return b;
    }

    public String eval(String s) { }
    public String wise() { }
    public String add(String s) { }
    public void set(String s) { }

    public void init(String a) {
        add(a);
        add(eval(tmpStatus));
        add(eval(wise()));
    }

    public String get() { }
}
```

Fig. 2. Examples of predefined annotations and classes

in order to identify the correct behaviour (or composition of behaviours), the adopted solution could be classified as an Annotated Aspect oriented solution (AA). In such an AA solution several aspects are responsible for the interception of the relevant OO methods, whose execution is ultimately substituted with native code, interacting with the remaining portion of the software by means of JNI. In this case, we want to run JNI instances driven by some parameters. Some of them are embedded into the OO code, others should be evaluated at runtime, e.g. the core occupancy, whether or not to use pinned memory, the effective bandwidth utilisation as in equation (2). While the presented toolbox makes all the needed computation by means of a meta-layer which reflects on the OO code, in order to correctly interpret the desired behaviour it is up to the developer to annotate his application code. In order to minimise this concern we have developed an easy way to set up all the needed parameters and to select the portions of code to parallelise.

Some annotations are given by the toolbox itself and are part of a library. Among such default annotations some of them are used to let the aspects inject the appropriate code in the right points within the code. Figure 2 shows annotation `@GPUparal` that allows identifying a class that has to be substituted with CUDA compliant code, then some code is executed on the GPU card and connected with the OO software by means of JNI. Such a parallel execution could be organised into several streams by means of a `@GPUstream` annotation. The latter takes as input an ID in order to univocally identify an execution stream (mandatory in case of asynchronous execution and pinned memory utilisation). Moreover, `@GPUparal` annotation allows the developer to define mandatory behaviours for certain classes or methods. Such mandatory behaviours could be defined along with the implementation of application methods and classes and become proper directives, when the

```
public class Main {
    @GPU_param(threads=64, blocks=8, async=1,
        pinned=1, buff=1024, streams=1, mixbehav=0,
        fixed="threads,async,pinned")
    public static void main(String[] args) {
        // ...
    }
}

@GPUparal(fixed="default")
public class MyClass {

    @GPUstream(1)
    public void bigMethod1() {
        // ...
        Behaviour.getInstance().set("none");
        smallMethod_S1();
        // ...
    }

    public void bigMethod_S0() {
    }

    @GPUstream(2)
    public void bigMethod3() {
        // ...
        otherClass.smallMethod_OC();
        // ...
        smallMethod_S2();
        // ...
    }
}
```

Fig. 3. Examples of application customisation by using Java annotations

implemented methods (or classes) are called (or used).

Some specific behaviours can be implemented to be enabled at a given moment in the code, e.g. before a call to a method, then method `set()` on an instance of class `Behaviour` has to be called (see also Figure 3). The said method `set()` would then override any other behaviour except for the mandatory behaviour.

Figure 3 shows how an application takes advantage of the proposed toolbox by means of Java annotations. These set important parameters, or communicate some global behaviour, which is then implemented for all the parallelised code. Another behaviour could be introduced by the aspects when appropriate.

E. Injection module

The developed aspect `GPUinjector` (see Figure 4) takes into account the behaviour resulting from: (i) class-related annotation `@GPU_parallel`, (ii) method-related annotation `@GPUstream`), and (iii) the directives given by means of predefined annotation `@GPU_param`. When the aspect intercepts a called method for a given instance, it observes all behavioural directives (given on the code by means of the method `set()` for class `Behaviour`). Moreover, it takes into account the overall ensemble of parameters and circumstances that intervene at runtime. This latter reasoning could lead the aspect to find a more profitable or advantageous setup in order to enrich or modify the given set of behaviours (except in the case of mandatory behaviours).

At the beginning of an application execution, as soon as class `Behaviour` is loaded, it is populated by proper data values, then using the mandatory instructions, several behaviours are configured. Such mandatory instructions, as

```
public aspect GPUinjector {
    pointcut GPUpar(GPUparal ann, Object obj):
        this(obj) && execution(@GPUparal void *.*(..))
        && @annotation(ann);

    void around(GPUparal ann, Object obj):
                        GPUpar(ann, obj) {
        try {
            // ...
            Behaviour.getInstance().init(ann.fixed());
            // ...
        } catch (Exception e) { }
    }
}

extern "C" JNIEXPORT void JNICALL
Java_Injected_CUDAcode(JNIEnv* env, Gstruct gpudata){
    env-> ...
}
```

Fig. 4. Aspect for CUDA code injection

well as all the other initialisation directives are given in the application code by means of the default annotations.

When an instance of an annotated application class will be intercepted by `GPUinjector`, the class-related annotations are taken and then stored on an appropriate hash table handled by the aspect. The same elaboration is performed for method-related annotations, which are stored in another dedicated hash table. Data stored in such hash tables are shared by the several instances of the same class, once they are intercepted, since the same parallelisation is desired for all instances of the same class.

Data gathered by method-related and class-related annotations are then merged with the mandatory behaviours and other specifications given by the application developer in order to exclude conflicting behaviours. Then, the resulting behaviours are given by the attribute `tmpStatus`, as extracted by the aspect from the annotations. Such behaviours are evaluated by method `eval()` in class `Behaviour` in order to check the compatibility with the mandatory behaviours before modifying the general behaviour encoded as a string `Status`.

Finally, aspect `GPUinjector` enables us to integrate on-the-fly new behaviours for some advantageous circumstances. This latter integration is made by means of method `wise()` in class `Behaviour`. After the evaluation the behaviours are added (or modified) with method `add()` on the same class.

When the whole image of the behaviour is composed, then the aspect calls the code generator that joins preexistent portions of code related to each behaviour composition (or custom made CUDA compliant code linked by the developer to a certain composition on default or custom behaviours). Figure 5 shows the Java code and annotations and calls that connect with our toolbox.

V. CONCLUDING REMARKS

The possibility to have an easy to use and modular toolbox for GPGPU programming opens an entire new range of possibilities in the field of fast and performance oriented computing. By means of our toolbox the developer need not use any external or proprietary compiler. Consequently, this toolbox offers virtually unlimited reusability with the possibility to link

```
import java.lang.annotation.*;
import Gclass.*;


@GPUparal("none")
public class MyGPUclass {
    public void myGPUmethod(Ptype data) {
    }
}

public class Test {
    @GPU_param(threads=48, async=0, mixbehav=1,
        fixed="threads,split1D")
    public static void main(String[] args) {
        Ptype data = new Ptype();

        Behaviour.getInstance().set("default,split1D");
        MyGPUclass.myGPUmethod(data);

    }
}
```

Fig. 5. Example of Java code before injection

with CUDA-compliant code. The implementation of advanced features is made by using aspect-oriented code. In this way, it is possible to have an high customisation level for the definition of new behaviours and performances improvements due to an advanced management of the allocation and freeing of memory on the device. This will provide means to control the lifecycle of variables stored on the device. All this without compromising the modularity and simplicity of the implementation which are the main driving forces of this work.

Moreover, the presented toolbox works as an integrated translational utility for the automatic conversion between sequential OO code as well as for the integration of CUDA-compliant code, providing an advanced interpretation method. In this way programmers that intend to make use of the advantages offered by this toolbox will be able to reuse written code by translating it into a GPU-enabled implementation, with a robust compatibility between this toolbox and any independent OO code.

REFERENCES

[1] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance analysis of high performance computing applications on the amazon web services cloud," in *Proceedings of International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 159–168, IEEE, 2010.

[2] C. Napoli, G. Pappalardo, and E. Tramontana, "Improving files availability for bittorrent using a diffusion model," in *23rd IEEE International WETICE Conference*, pp. 191–196, IEEE, 2014.

[3] S. Che, M. Boyer, J. Meng, D. Tarjan, J. Sheaffer, and K. Skadron, "A performance study of general-purpose applications on graphics processors using cuda," *Journal of Parallel and Distribuited Computing*, vol. 68, pp. 1370–1380, 2008.

[4] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," *Queue*, vol. 6, no. 2, pp. 40–53, 2008.

[5] A. Rueda and L. Ortega, "Geometric algorithms on cuda," *Journal of Virtual Reality and Broadcasting*, no. 200, 2008.

[6] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, and J. Irwin, "Aspect-oriented programming," *ECOOP 97 Object Oriented Programming*, pp. 220–242, 1997.

[7] G. Borowik, M. Woźniak, A. Fornaia, R. Giunta, C. Napoli, G. Pappalardo, and E. Tramontana, "A software architecture assisting workflow executions on cloud resources," *International Journal of Electronics and Telecommunications*, vol. 61, no. 1, pp. 17–23, 2015. DOI: 10.1515/eletel-2015-0002.

[8] R. Giunta, G. Pappalardo, and E. Tramontana, "An aspect-generated approach for the integration of applications into grid.," in *Proceedings of the symposium on Applied computing*, ACM, 2007.

[9] C. Napoli, F. Bonanno, and G. Capizzi, "An hybrid neuro-wavelet approach for long-term prediction of solar wind," in *IAU Symposium*, no. 274, pp. 247–249, 2010.

[10] R. Giunta, G. Pappalardo, and E. Tramontana, "Superimposing roles for design patterns into application classes by means of aspects," in *Proceedings of Symposium on Applied Computing (SAC)*, ACM, 2012.

[11] M. Woźniak, Z. Marszałek, M. Gabryel, and R. K. Nowicki, "Modified merge sort algorithm for large scale data sets," *Lecture Notes in Artificial Intelligence - ICAISC'2013*, vol. 7895, pp. 612–622, 2013. DOI: 10.1007/978-3-642-38610-7_56.

[12] E. Tramontana, "Automatically characterising components with concerns and reducing tangling," in *Proceedings of Computer Software and Applications Conference (COMPSAC) workshop QUORS*, IEEE, 2013.

[13] C. Krueger, "Software reuse," *ACM Computing Surveys (CSUR)*, vol. 24, no. 2, pp. 131–183, 1992.

[14] C. Napoli, G. Pappalardo, and E. Tramontana, "An agent-driven semantical identifier using radial basis neural networks and reinforcement learning," in *XV Workshop "Dagli Oggetti agli Agenti"*, vol. 1260, CEUR-WS, 2014.

[15] J. Auerbach, D. F. Bacon, P. Cheng, and R. Rabbah, "Lime: a java-compatible and synthesizable language for heterogeneous architectures.," in *Proceedings of the ACM International Conference on Object-Oriented Programming Systems, Languages, and Applications*, pp. 89–108, ACM, 2010.

[16] A. Fonseca and B. Cabral, "Aeminiumgpu: An intelligent framework for gpu programming," in *Proceedings of Facing the Multicore-Challenge III, At Stuttgart*, 2012.

[17] M. Boyer, K. Skadron, and W. Weimer, "Automated dynamic analysis of cuda programs," in *Third Workshop on Software Tools for MultiCore Systems*, 2008.

[18] M. Mongiovi, G. Giannone, A. Fornaia, G. Pappalardo, and E. Tramontana, "Combining static and dynamic data flow analysis: a hybrid approach for detecting data leaks in Java applications," in *Proceedings of Symposium on Applied Computing (SAC)*, ACM, 2015.

[19] A. Calvagna and E. Tramontana, "Delivering dependable reusable components by expressing and enforcing design decisions," in *Proceedings of Computer Software and Applications Conference (COMPSAC) Workshop QUORS*, pp. 493–498, IEEE, July 2013.

[20] F. Bonanno, G. Capizzi, S. Coco, C. Napoli, A. Laudani, and G. Lo Sciuto, "Optimal thicknesses determination in a multilayer structure to improve the spp efficiency for photovoltaic devices by an hybrid fem—cascade neural network based approach," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2014 International Symposium on*, pp. 355–362, IEEE, 2014.

[21] R. Giunta, G. Pappalardo, and E. Tramontana, "A redundancy-based attack detection technique for java card bytecode," in *Proceedings of International WETICE Conference*, pp. 384–389, IEEE, 2014.

[22] C. Napoli, G. Pappalardo, and E. Tramontana, "Using modularity metrics to assist move method refactoring of large systems," in *Proceedings of International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, pp. 529–534, IEEE, 2013.

[23] G. Pappalardo and E. Tramontana, "Suggesting extract class refactoring opportunities by measuring strength of method interactions," in *Proceedings of Asia Pacific Software Engineering Conference (APSEC)*, IEEE, 2013.

[24] F. Bonanno, G. Capizzi, and C. Napoli, "Some remarks on the application of rnn and prnn for the charge-discharge simulation of advanced lithium-ions battery energy storage," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2012 International Symposium on*, pp. 941–945, IEEE, 2012.

[25] E. Tramontana, "Detecting extra relationships for design patterns roles," in *Proceedings of AsianPlop*, March 2014.

[26] F. Bonanno, G. Capizzi, G. Lo Sciuto, C. Napoli, G. Pappalardo, and E. Tramontana, "A cascade neural network architecture investigating surface plasmon polaritons propagation for thin metals in openmp," in *Artificial Intelligence and Soft Computing*, pp. 22–33, Springer International Publishing, 2014.

[27] G. Capizzi, F. Bonanno, and C. Napoli, "A new approach for lead-acid batteries modeling by local cosine," in *Power Electronics Electrical Drives Automation and Motion (SPEEDAM), 2010 International Symposium on*, pp. 1074–1079, IEEE, 2010.

[28] M. Ioki, S. Hozumi, and S. Chiba, "Writing a modular gpgpu program in java," in *Proceedings of workshop on Modularity in Systems Software (MISS)*, (New York, NY, USA), pp. 27–32, ACM, 2012.

[29] H. Nguyen, *GPU Gems 3*. Addison-Wesley, 2008.

[30] C. Napoli, G. Pappalardo, E. Tramontana, Z. Marszałek, D. Połap, and M. Woźniak, "Simplified firefly algorithm for 2d image key-points search," in *2014 IEEE Symposium on Computational Intelligence for Human-like Intelligence*, pp. 118–125, IEEE, 2014.

[31] G. Capizzi, F. Bonanno, and C. Napoli, "Recurrent neural network-based control strategy for battery energy storage in generation systems with intermittent renewable energy sources," in *IEEE international conference on clean electrical power (ICCEP)*, pp. 336–340, IEEE, 2011.

[32] M. Woźniak and D. Połap, "Basic concept of cuckoo search algorithm for 2D images processing with some research results : An idea to apply cuckoo search algorithm in 2d images key-points search," in *SIGMAP 2014 - Proceedings of the 11th International Conference on Signal Processing and Multimedia Applications, Part of ICETE 2014 - 11th International Joint Conference on e-Business and Telecommunications*, (28-30 August, Vienna, Austria), pp. 157–164, SciTePress, 2014. DOI: 10.5220/0005015801570164.

[33] F. Bonanno, G. Capizzi, G. Lo Sciuto, C. Napoli, G. Pappalardo, and E. Tramontana, "A novel cloud-distributed toolbox for optimal energy dispatch management from renewables in igss by using wrnn predictors and gpu parallel solutions," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2014 International Symposium on*, pp. 1077–1084, IEEE, 2014.

[34] Nvidia corporation, *NVIDIA CUDA Compute Unified Device Architecture programming guide*. 2007.

[35] C. Napoli, G. Pappalardo, E. Tramontana, and G. Zappalà, "A cloud-distributed gpu architecture for pattern identification in segmented detectors big-data surveys," *The Computer Journal*, p. bxu147, 2014.

[36] R. Giunta, G. Pappalardo, and E. Tramontana, "Aspects and annotations for controlling the roles application classes play for design patterns," in *Proceedings of the Asia Pacific Software Engineering Conference (APSEC)*, pp. 306–314, IEEE, 2011.

Comparing a Social Robot and a Mobile Application for Movie Recommendation: A Pilot Study

Francesco Cervone, Valentina Sica, Mariacarla Staffa, Anna Tamburro, Silvia Rossi
Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione
Università degli Studi di Napoli "Federico II"
via Claudio 21, 80125 Napoli, Italy
{mariacarla.staffa,silvia.rossi}@unina.it

***Abstract*—Social robots can be used as interfaces to provide recommendations to users. While a vast literature compares the user's behavior when interacting with a robot with respect to a virtual agent, in this paper, we conduct a first evaluation on how the user's choices are affected if the recommendations are provided respectively by a mobile application or by social robots with different degree of interaction capabilities. This pilot study shows that the sole embodiment condition of the robot does not imply significant changes in the users' choices that prefer to interact with the mobile application. However, the adoption of additional communication channels such as gestures, gaze and voice pitch, which change accordingly to the suggested movie genre, improves the users acceptability.**

I. Introduction

Social robots will be used in the next future in many application domains, which span from entertainment and education to health–care. In order to be accepted in our houses, they should be perceived as trusting, helpful, reliable and engaging [1]. This is particularly important in case the robot is in charge to convey information to a person (such teaching skills, collaborating towards a particular goal or providing advises). Social robots, as well as virtual agents, can be used as interfaces to provide recommendations. Such embodied social agents make interaction more meaningful than it is when provided by simple interfaces (which do not display actions or speech), because users' attitude towards social agents is similar to that they show towards other people.

Recommendation systems aim to provide the user with personalized advises and suggestions in many different domains, such as books, movies or music. Such suggestions are provided according to the available information the system has on the user (e.g., his/her preferences or his/her past interaction with the system). Hence, recommendations can be provided suggesting items similar to items liked by the user or liked by similar users. The effectiveness of the provision of recommendations relies itself on the concept of trust [2] with respect to the system that proposes the recommendation. Such trust on the recommendation depends upon machine accuracy, predictability and dependability [3] (e.g., by recommending items which are positively evaluated by the users). In literature, different studies compared the impact of recommendation and the advises as provided by social robot with respect to virtual agents [4], [5], by showing that the embodiment condition, as provided by the robot, has more impact with respect to 2D/3D virtual agents on a screen. Real robots affect subject decision-making more effectively than computer agents in real world environments [6]. Moreover, non-verbal behaviors serve important functions in affecting the trustworthiness of a recommendation [7]. In fact, a robot ability to build a trust relationship depends on its capacity to help people understand it, in part through non-verbal behavior. Emotion-related signals, such as those provided by voice pitch changes in speech or gestures are non-verbal behaviors that influence human trust [8]. It has been, indeed, well-documented that humans expect from humanoid robots socially intelligent responses [9]. This leaves the possibility that an agent may influence how humans perceive a recommendation through the presence of more or less communication abilities.

In this paper, we present a pilot study to evaluate the extent of the use of a robotic system in accepting a recommendation not with respect to a virtual agent, but to very common interfaces such as mobile applications. Our experiments aim at evaluating the users' acceptance of recommendations as well as their engagement when the robot or a mobile application are displaying such advises. In particular, we provided the same information contents on recommended movies, but using three different interaction conditions. In the first condition, by using a mobile application, contents will be provided by text shown on the mobile screen. In the second one, the same contents will be provided by using a humanoid robot interacting using speech. Finally, in the third condition the humanoid robot will encompass both voice and genre-driven motion primitives.

II. Related Works

A vast literature compared the behavior as well as the acceptance of robots with respect to their virtual counterparts. Embodied robots are consistently perceived as more engaging than a character on a video display, and sometimes as engaging as a human. For example, in Kidd and Breazeals [1] work subjects were instructed by an agent (either to a human, a robot, or a cartoon robot), which showed only its eyes to the subjects. All three visual presentations were accompanied by the same vocal instructions. The Authors' purpose was to understand which types of interaction involved more the user (evaluated by a questionnaire), and showed more reliability, usefulness and trust. The results showed that the robot, given its physical presence, was considered as more engaging, credible and informative, as well as being more pleasant as an interaction partner. As in [1], in our experiment, we provided the same information contents with very simple and controlled interfaces, but using different interaction modalities.

How physical embodiment, as opposed to virtual presence, affects human perception of social engagement with an artificial agent was addressed in [4], [6], [10]. In [6] the Authors evaluated the persuasion effects of a computer agent and of a robot in various tasks as, for example, in following indications. The results showed that the user has shown more confidence and more trust for the physical robot. User's behavior in accepting advises was investigated also in [4]. The results showed that the user preferred to interact with the robot because it was more effective in providing recommendations. Shinozawa et al. considered the effect of persuasion in a laboratory environment comparing a robot and a computer agent (with a 2D or 3D appearance) displayed on a monitor [10]. The results showed that the geometric coherence between a social agent and the environment was an important factor in the interaction, independently whether it is 2D or 3D. Conversely to these approaches, in our study, we compared the effects of adopting for a recommending task a robot with respect to a mobile application. This is, up to our knowledge, the first attempt to provide such comparison.

Finally, in [5] the Authors studied the impact of the robot size with respect to the user reactions in an advertising context. The purpose was to understand which robot was more suitable for interaction for advertising purposes. The results showed that, in the presence of robots of different sizes, the user considers it easier to interact with a smaller robot.

III. System Architecture

In order to evaluate our hypothesis, we developed a client/server application, where the server provides the recommendation service and the possible clients can be a humanoid robot or a mobile application. Clients are in charge to ask for a list of recommendations (in particular of movies) to the server and to show them to users. The social robots and the mobile application will provide the same information, but in different ways (i.e., through different communication channels). This diversity should be reflected in a different perception of the recommendations by the users, and, presumably, it will affect their choices. In order to provide recommendations, the *Recommendation Engine* needs some initial movie ratings from the users. Hence, independently from the client type the users interact with, the initial ratings are performed by using the mobile application, which allows users to easily evaluate movies by means of a friendly graphical interface. The main blocks of the developed framework are detailed in the following subsections.

A. Movie Recommendation Server

The server layer of this architecture is characterized by the recommendation system (see Figure 1). It is developed in Java and it is hosted by a Tomcat servelet container. The core of this layer is a Web Server, used to store, process and deliver requested content to clients, which are provided through JSON-based API both to communicate with the humanoid robot and with the android mobile application. The module *Recommendation Engine* is the core of the recommendation layer. We adopted the Apache Mahout library[1] to predict the user ratings, and chosen the MovieTweetings [11] dataset to train the system

[1] http://mahout.apache.org/

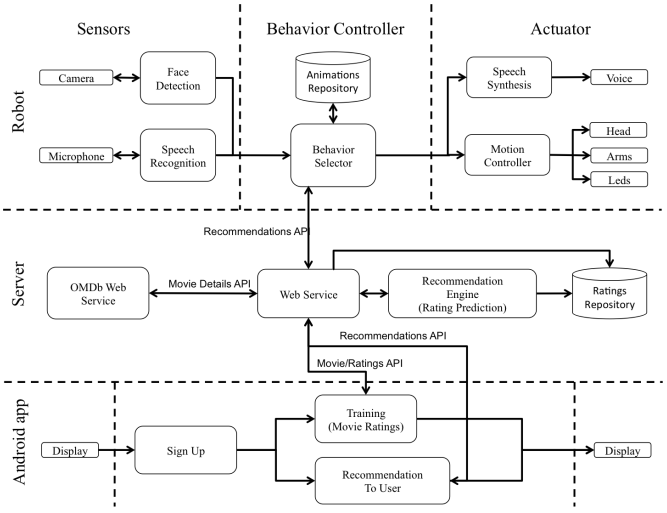


Fig. 1. Client/Server application.

and to populate the *Ratings Repository*. MovieTweetings consists of movie ratings contained in well-structured tweets on the *Twitter.com* social network. This information is contained in three files: *users.dat*, *ratings.dat* and *movies.dat*, which provide respectively the user identification number, his/her associated ratings and a list of movies. The dataset is updated every day, therefore its size is constantly changing. At the last access, it contained about 35000 users, 360000 ratings and 20000 movies.

The recommendation engine provides rating predictions when the recommendation API is invoked. To achieve this goal, we used *item-based City Block distance*, also known as *Manhattan distance*. In Mahout implementation, the generic movie i is represented by a boolean vector:

$$i = [r_{1i}, r_{2i}, ..., r_{ni}],$$

where n is the number of users in the dataset and $r_{ui} = 1$ if user u rated the movie i. The distance between two movies rated by user u is the sum of the absolute value of the differences of the two associated vector components. More formally, the distance between items i and j is:

$$d(i,j) = \sum_{u=1}^{n} |r_{ui} - r_{uj}|.$$

B. Android Application

On the bottom of Figure 1, the architecture design of the mobile Android application is depicted. The first duty for the user, when he/she accesses the application, is to sign up/sign into the system. As explained above, when the interaction starts, users have first to provide a certain number of movie ratings (at least 20 movies). The *Training* module is dedicated to provide an interface to get movie lists and to store movies ratings. If a user is in the training stage, he/she can browse movies by ordering them by most rated or randomly, or search for a movie (filtering by genre or title, see Figure 2).

After this first stage, user can get movie recommendations from the server. When the server gets the recommendation request, once calculated the best movies for the user, it

retrieves additional details about the film, like, for example, the director, writers, actors and genres using OMDb[2] web service. Fortunately, MovieTweetings data set stores, for each movie, its IMDb id, which can be used to address the OMDb service. The Android application shows on the screen the recommendations for the users through textual and graphical descriptions.

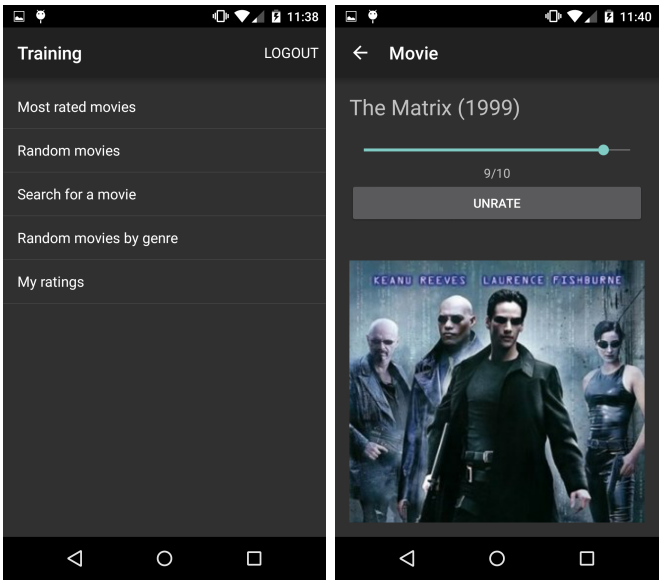


Fig. 2. Snapshots from the movie-app training phase.

C. The Robot Client

The Robot Client architecture has been designed considering the adoption of a NAO T14 robot model, consisting in a humanoid torso with 14 degrees of freedom (2 for the head and 12 for the arms) developed by Aldebaran Robotics[3]. We controlled the NAO platform by means of the standard robotic operating system (ROS) and using the Python programming language for developing the ROS nodes. NAO is endowed with two main sensors: a camera and a microphone through which it receives signals from the external environment. Camera frames are processed by a *Face Detection* module to detect users presence into its visual field. Sounds obtained from the microphone are processed by *Speech Recognition* module.

As actuators NAO can use the following communication channels: voice, arms, head and eyes led. The *Behavior Selector* module is in charge of providing two different interaction conditions. In the first case, it presents to users the recommended movies and their relative information only through speech, while, in the second case, such description is accompanied with gestures, gaze, eyes coloring through the *Motion Controller*, and pitch voice changes through *Speech Synthesis*. In this latter case, the *Behavior Selector* gets motion animations from an *Animations Repository*, based on the genre of the recommended movie, to execute animated speech, as will be detailed in the followings.

[2]http://www.omdbapi.com - The Open Movie Database is a free web service to obtain movie information.

[3]https://www.aldebaran.com/en

a) Face Detection: This module is based on a face detection/recognition solution provided by OKI and included in the Python SDK for NAO. Such module continuously processes frames from the NAO camera in order to detect a human face. Once a face is detected, it provides its position. Moreover, in the third condition, the module continues to provide the user position coordinates to the *Motion Controller*, that allows NAO to track the face by moving its head.

b) Speech Recognition: This module gives to the robot the ability to recognize a predefined words list, and specifically the usernames and the acceptance/rejection of a recommendation. It is based on module provided by Aldebaran, which relies on sophisticated speech recognition technologies provided by NUANCE for NAO Version 4. Before starting, the robot needs to receive the list of usernames (UsersList). Then, once the system has detected a face through *Face Detection*, NAO asks for a username and listens until a word is recognized. Currently, system does not provide a real authentication when interacting with the robot because the only way to communicate with NAO is the speech. Users should sign in through an input system like a keyboard or a mobile application.

c) Behavior Selector: Through this module, we generate all the gestures, gaze and the feedback for the user. Once a user has been recognized, a user tracking system allows the robot to track the target by moving its head. Movie information is provided to the user with the *Speech Synthesis* module with different speech intonations, but it can be accompanied with arms gestures and facial expressions (e.g., different eyes colors) generated through *Motion Controller* module. The *Behavior Selector* gets recommendations from the Web Service and related animations from the *Animations Repository*. The main task of this module is mapping the movie genre into a predefined set of animations and eyes colors. For example, if NAO recommends a drama, led eyes become red and gestures are more serious, while for a comedy led eyes become green and gestures are more joyous. The pitch of the voice is accordingly manipulated by the *Speech Synthesis* module.

IV. A Pilot Study

The pilot study is conducted by considering three different interaction conditions, where participants receive two movie recommendations for each condition.

A. Procedure

The testing procedure main steps are: (i) the user provides new rates for a list of movies (training phase) at the beginning of the interaction; (ii) the recommendation system generates the top-six recommendations for each user, which will be shown to users through the three interaction conditions in a random way (two for each condition); (iii) after each test condition the participant has to answer to a questionnaire concerning the specific condition and at the end of the overall experiment to a general questionnaire.

B. Method

The design of this study is a within-subjects, counterbalanced, repeated measures experiment. The three considered interactive conditions are the following:

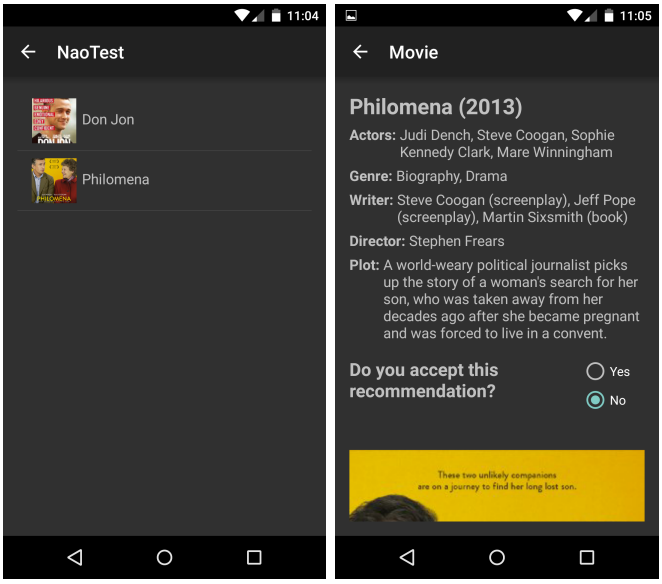


Fig. 3. Snapshots from the *App* recommendation phase.

Condition 1 (App): in this setting, neither of the robot modalities are used. The user only interacts with the mobile application that provides to the user two different movies suggestions. For each movie the app provides the title and additional information by displaying text and images on the screen. For each recommended movie, the user has to reply about his/her likelihood to see it (see Figure 3).

Condition 2 (Nao): in this setting, the robot is located on a table standing still and waiting for a person to interact with. When the robot recognizes a face in its filed of view, it greets the person, introduces itself, and asks for a username. NAO presents the two recommendations by telling the movie title accompanied with the same information provided in *Condition 1* (plot, genre, actors, and so on). Finally, it asks the user if she/he agrees to see this movie and stores the answer.

Condition 3 (ENao): in this setting, differently from the previous condition, the robot is endowed with the motion controller module. When the robot is not interacting with anyone, it simply looks around and waits for a person to talk with. In this interaction condition NAO, in addition to tell movie information, gesticulates, changes eyes led color and voice pitch according to the recommended movie genre (see Figure 4).

C. Participants

18 subjects participated in this experiment with an average age of 32 years and a graduate education, for a total of 13 males and 5 females. All the participants were Italian native speakers with an average English level of 2.39 and Robotic skills of 3.17 on a likert–scale from 1 to 5. The language adopted for the experiment was the English both for text description and for the robot's voice synthesizer. The testers were not informed about the NAO interaction capabilities. In Table I, personnel data of participants are collected.

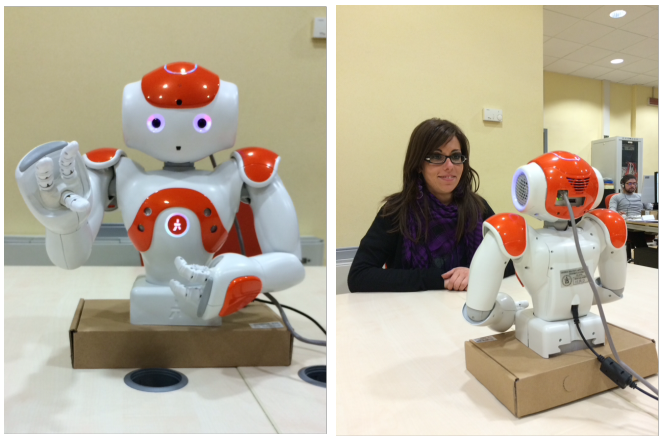


Fig. 4. NAO and ENAO conditions.

TABLE I. 18 PARTICIPANTS DATA.

	min	max	avg
Age	22	55	32
	male	female	
Gender	72%	28%	
	low	high	avg
English Level	61%	39%	2.39
Robotic Skills	44%	56%	3.17

V. RESULTS

We hypothesized that the robot as compared with the application will be more engaging and better liked, and hence recommendation provided by the robot should be more likely to be accepted. Moreover, the condition with animated motion should be more engaging and better liked with respect to the simple robot.

A. Quantitative Analysis

In order to evaluate the degree of acceptance of the recommendations when provided by different conditions, we calculated the *selection ratio* indicating the number of accepted recommendations with respect to the total number of recommendations for a each specific condition.

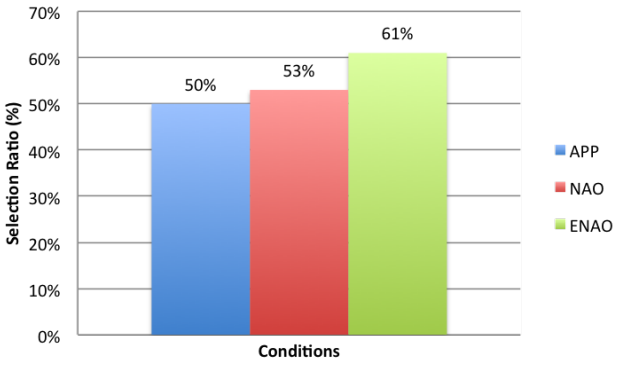


Fig. 5. Percentage of accepted movie recommendations for each Condition.

In Figure 5 the selection ratio is expressed in percentage shows that there is a minimum difference in the acceptance rate

between the recommendations provided by *App* and *Nao*, while there is a slightly bigger difference between *App* and *ENao* conditions. This fact is in accordance with our hypothesis that people are inclined to accept more recommendations provided through a more natural interaction, even if the sole embodiment condition (*Nao*) does not imply significant changes in the testers acceptability level. However, due to the limited number of participants and recommendations provided to each participant these differences, evaluated using ANOVA, are not yet statistically significant, while there is a significant Pearson correlation between *App* and *Nao* conditions ($r = 0.43$ with $p = 0.08$ that is significant at $p < 0.10$) acceptance trends. As future work we will extend such experimentation with a greater number of users.

Since users selected for testing are all Italian native speakers, and not all have the same level of familiarity with robotics applications, we felt it appropriate to consider data by aggregating the results by both the level of English proficiency (e.g., the language used to provide recommendations) and the degree of experience with robots. We thus computed correlations for the acceptance ratio and among conditions couples by grouping users with a high (from 4 to 5) or low English level (from 1 to 3) and a high or low familiarity with robots (see Section V-B):

- high English level: Pearson showed a negative strong correlation ($r = -0.76$ with $p < 0.05$) between *Nao* and *ENao*;
- low English level: there is a significant correlation between *App* and *Nao* ($r = 0.57$ with $p = 0.07$);
- high familiarity: nothing significant;
- low familiarity: once again we had a moderate correlation between *App* and *Nao* ($r = 0.65$ with $p = 0.08$), but in this case also *Nao* and *ENao* have a moderate correlation ($r = 0.65$ with $p = 0.08$).

These results show that for testers with a low English level reading text from an application or hearing speech from a robot does not have a relevant impact on the decision making, while for good English skilled participants adding an animated behavior changes the acceptance trend. This could be due to the fact that the users attention in the first case is quite all focused on understanding the text or speech. Moreover, users with low familiarity in robotics shows acceptance trend similar in both *Nao* and *ENao* cases.

B. Qualitative Analysis

Our evaluation also takes into account the impressions of users with respect to the interaction with the different conditions. For this aim we propose a qualitative questionnaire organized in three specific sections: (i) personal information for collecting information about the user (age, gender, english level, and familiarity with robotics); (ii) Qualitative questions regarding the application easy of use, naturalness and satisfiability consisting of 6 questions; (iii) two question specific for the conditions involving the humanoid, dealing with the sense of trust and movements naturalness of the robot. While the general information have been asked at the beginning of the tests, the testers have been asked to reply to the specific questions at the end of each experiment. We adopted a classical likert scale from 1 to 5. Only for question 6 we explicitly ask for a preference by the users where index from 1 to 3 represent respectively the preference for *APP*, *NAO* and *ENAO*. The questionnaire structure is reported in Table II.

TABLE II. QUESTIONNAIRE.

Section	Question
Personal Information	Age? Gender? English level? (1 to 5) How familiarized are you with robotic applications? (1 to 5)
Qualitative Questions	Q1. How easy was to perform the task? (1 to 5) Q2. Did the system react accordingly to your expectations? (1 to 5) Q3. How natural is this kind of interaction? (1 to 5) Q4. How satisfying do you find the interactive system? (1 to 5) Q5. You were sure (5) or unsure (1) of your answers? Q6. Which mode of interaction you preferred? (1 to 3).
Robot-related Questions	Q7. The agent was believable (5) or unbelievable (1). Q8. The agents motions were natural (5) or unnatural (1).

Figure 6-(a) shows the mean value of the answers to the qualitative questions for each interactive condition. Users found the interaction with *App* easier than the interaction with *Nao* and *ENao* (Q1 in Table II). ANOVA test endorsed this result by showing that differences between *App* and *Nao* ($F = 6.48$ with $p = 0.02$) and between *App* and *ENao* ($F = 3.34$ with $p = 0.08$) were statistically significant. A slightly preference for the interaction with the *App* was also shown by the answers to Q3 and Q4 questions. In this case, the only statistical significant differences were between *App* and *Nao* conditions for Q3 ($F = 4.25$, $p = 0.05$) and Q4 ($F = 3.89$, $p = 0.06$), thus the *App* was more natural and satisfying than *Nao* interacting with speech.

For each question, we computed correlations between *App*, *Nao* and *ENao*:

- *App-Nao*: we notice a moderate correlation for Q2 ($r = 0.50$, $p = 0.03$), Q3 ($r = 0.44$, $p = 0.07$) and Q4 ($r = 0.52$, $p = 0.03$);
- *Nao-ENao*: there is a moderate correlation for Q2 ($r = 0.59$, $p = 0.01$) and Q4 ($r = 0.48$, $p = 0.04$) and a strong correlation for Q5 ($r = 0.72$, $p < 0.01$);
- *App-ENao*: there are no significant correlations.

If we observe the histogram of question 6 (see Figure 6-(b)), a part from the approval ratings average of the qualitative questions from Q1 to Q5, it is quite evident that the major part of the users prefers to interact with the Humanoid endowed with emotion-based capabilities. This is probably due to the fact that the humanoid robot has the potential to portray a rich repertoire of non–verbal behaviors that have familiar social meaning for users, who perceive the interaction more natural and engaging because of the received socially intelligent responses by the robot. Histogram in Figure 6-(c) shows that the robotic agent is perceived in the average believeble both if it shows or not non-verbal feedback, and the agent motion is perceived as natural.

As for the quantitative case, for each pair of conditions, we try to correlate answers considering only users with high or low English level or familiarity with robots applications:

- high English level: there is a moderate correlation between *App* and *Nao* ($r = 0.53$, $p < 0.01$), and *Nao* and *ENao* ($r = 0.41$, $p < 0.01$);

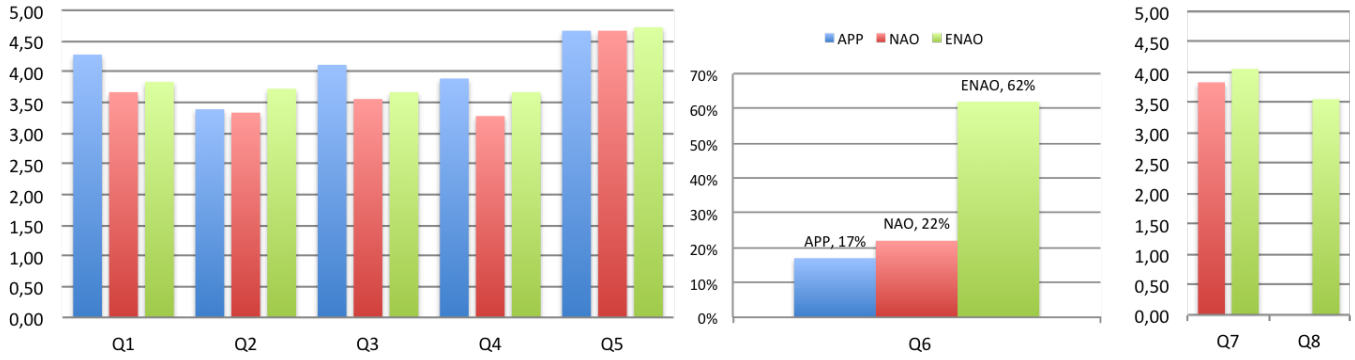


Fig. 6. Approval ratings average with respect to the qualitative questions.

- low English level: as in the previous case, *App* and *Nao* are significantly correlated ($r = 0.43$, $p < 0.01$), as well as *Nao* and *ENao* ($r = 0.64$, $p < 0.01$);
- high familiarity: Pearson shows a strong correlation between *App* and *Nao* ($r = 0.75$, $p < 0.01$) and between *Nao* and *ENao* ($r = 0.82$, $p < 0.01$). There is a moderate correlation for *App* and *ENao* ($r = 0.60$, $p < 0.01$);
- low familiarity: finally, we have a moderate correlation between *App* and *Nao* ($r = 0.39$, $p < 0.01$) and between *Nao* and *ENao* ($r = 0.45$, $p < 0.01$).

Both for *Nao* ($F = 3.95$, $p = 0.05$) and *ENao* ($F = 4.89$, $p = 0.03$), in the case of low and high familiarity with robots, ANOVA shows significant differences between these categories of users. In both cases, the mean values of answers of users with high familiarity is greater than other users. There are not significant differences in grouping per English skills.

VI. Conclusion

Social robots have been used in advertisements in public spaces mainly because of their greater ability to grab customer attention with respect to displays. Previous research mainly investigated the advantage of a physical body in engaging the user in an interaction with respect to its virtual counterpart. In this work, we compared, in a pilot study, the effect of a social robot with different communication channels, with respect to a well–known interface such as a mobile application in providing recommendations, and evaluated the human behavior through quantitative and qualitative analysis. From the qualitative questionnaire it arises that the users, on average, perceive the interaction with the mobile application (*App*) easier than those with the social robot (*Nao*, *ENao*) independently from the degree of interaction capabilities. Furthermore, *App* appeared more natural and satisfying than both *Nao* and *ENao* robot. This result naturally arises from the fact that the most of the users have more familiarity with mobile applications rather than with robots. Moreover, the sole presence of the robot does not provide an improvement in the acceptance rate, while the additional communication capabilities provided by the *ENao* humanoid robot generate for the users a higher level of satisfaction with respect to the expectations compared with the other two interaction modalities, and a slightly increase in the acceptance rate (but not yet significant). In fact, when involved in an interaction, humans expect non-verbal signals from humanoid robots as well as they did with people. Indeed, when robotic emphatic responses (*Nao*) are absent or not sufficient, trust decreases.

In most cases, there are correlations between *App* results and *Nao*, and *Nao* and *ENao*, but not between *App* and *ENao*. In our opinion, the leading cause of these results is due to the smaller difference between the interaction with the mobile application and *Nao* condition (e.g., they provide the same content, but one with text and the other through speech), and between *Nao* and *Enao* conditions (e.g., they share the same interface – an embodied agent – but with different interaction capabilities). Regarding the first and the third conditions, the large difference between the two modes of interaction implies no significant correlations between each other.

In future works we will extend the pilot study by selecting more users in order to extend our evaluation and to achieve more significant results.

References

[1] C. Kidd and C. Breazeal, "Effect of a robot on user perceptions," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ Int. Conf. on*, vol. 4, Sept 2004, pp. 3559–3564 vol.4.

[2] W. Wang and I. Benbasat, "Trust in and adoption of online recommendation agents," *Journal of the Association for Information Systems*, vol. 6, no. 3, p. 72, March 2005.

[3] S. M. Merritt and D. R. Ilgen, "Not all trust is created equal: Dispositional and history-based trust in human-automation interactions," *Human Factors*, vol. 50, no. 2, pp. 194–210, 2008.

[4] A. Powers, S. Kiesler, S. Fussell, and C. Torrey, "Comparing a computer agent with a humanoid robot," in *2nd ACM/IEEE Int. Conf. on Human-Robot Interaction (HRI)*, March 2007, pp. 145–152.

[5] M. Shiomi, K. Shinozawa, Y. Nakagawa, T. Miyashita, T. Sakamoto, T. Terakubo, H. Ishiguro, and N. Hagita, "Recommendation effects of a social robot for advertisement-use context in a shopping mall," *Int. Journal of Social Robotics*, vol. 5, no. 2, pp. 251–262, 2013.

[6] W. Bainbridge, J. Hart, E. Kim, and B. Scassellati, "The effect of presence on human-robot interaction," in *The 17th IEEE Int. Symposium on Robot and Human Interactive Communication (RO-MAN)*, Aug 2008, pp. 701–706.

[7] C. M. de Melo, L. Zheng, and J. Gratch, "Expression of Moral Emotions in Cooperating Agents," in *Proceedings of the 9th International Conference on Intelligent Virtual Agents (IVA)*, Amsterdam, The Netherlands, Sep. 2009.

[8] T. B. R and B. Ross, "Emotional expressivity and trustworthiness: The role of nonverbal behavior in the evolution of cooperation," in *Journal of Nonverbal Behavior*, A. R. Library, Ed., vol. 23(3), 2003.

[9] C. Nass and Y. Moon, "Machines and mindlessness: Social responses to computers," *Journal of Social Issues*, vol. 56, pp. 81–103, 2000.

[10] K. Shinozawa, F. Naya, J. Yamato, and K. Kogure, "Differences in effect of robot and screen agent recommendations on human decision-making," *Int. J. Hum.-Comput. Stud.*, vol. 62, no. 2, pp. 267–279, Feb. 2005.

[11] S. Dooms, T. De Pessemier, and L. Martens, "Movietweetings: a movie rating dataset collected from twitter," in *Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys 2013*, 2013.

The Positive Power of Prejudice: A Computational Model for MAS

Alessandro Sapienza, Rino Falcone and Cristiano Castelfranchi
Institute of Cognitive Science and Technologies, ISTC-CNR, Rome, Italy
{alessandro.sapienza, rino.falcone, cristiano.castelfranchi}@istc.cnr.it

***Abstract*— In MAS studies on Trust building and dynamics the role of direct/personal experience and of recommendations and reputation is proportionally overrated; while the importance of inferential processes in deriving the evaluation of trustees' trustworthiness is underestimated and not enough exploited.**

In this paper we focus on the importance of generalized knowledge: agents' categories. The cognitive advantage of generalized knowledge can be synthesized in this claim: "It allows us to know a lot about something/somebody we do not directly know". At a social level this means that I can know a lot of things on people that I never met; it is social "prejudice" with its good side and fundamental contribution to social exchange. In this study we experimentally inquire the role played by categories' reputation with respect to the reputation and opinion on single agents: when it is better to rely on the first ones and when are more reliable the second ones. Our claim is that: the larger the population and the ignorance about the trustworthiness of each individual (as it happens in an open world) the more precious the role of trust in categories. In particular, we want investigate how the parameters defining the specific environment (number of agents, their interactions, transfer of reputation, and so on) determine the use of categories' reputation.

This powerful inferential device has to be strongly present in WEB societies.

I. INTRODUCTION

In MultiAgent Systems (MAS) and Online Social Networks (OSN) studies on Trust building and dynamics the role of direct/personal experience and of recommendations and reputation (although important) is proportionally overrated; while the importance of inferential processes in deriving the evaluation of trustee's trustworthiness is underestimated and not sufficiently exploited (a part from the so called "transitivity", which is also, very often, wrongly founded).

In particular, generalization and instantiation from classes, categories [8] and analogical reasoning (from task to task and from agent to agent) really should receive much more attention. In this paper we focus on the importance of generalized knowledge: agents' categories. The cognitive advantage of generalized knowledge (building *classes*, *prototypes*, *categories*, etc.), can be synthesized in this obvious claim: "It allows us to know a lot about something/somebody we do not directly know" (for example, I never saw Mary's dog, but - since it is *a dog* - I know hundreds of things about it). At a social level this means that I can know *a lot of things on people that I never met*; it is social "prejudice" with its good side and fundamental contribution to social exchange. How can I trust (for drugs prescription) a medical doctor that I never met before and nobody of my friends knows? Because he is a doctor!

Of course we are underlining the positive aspects of generalized knowledge, its essential role for having information on people never met before and about whom no one gave testimony. The more rich and accurate this knowledge is, the more it is useful. It offers huge opportunity both for realizing productive cooperation and for avoiding risky interactions. The problem is when the *uncertainty about the features* of the categories is too large or it is too wide the *variability of the performers* within them. In our culture we attribute a negative sense to the concept of prejudice, and this because we want to underline how generalized knowledge can produce unjust judgments against individuals (or groups) when superficially applied (or worst, on the basis of precise discriminatory intents). Here we want rather to point out the positive aspects of the prejudice concept.

In this study we intend to explain and experimentally show the advantage of trust evaluation based on classes' reputation with respect to the reputation and opinion on single potential agents (partners). In an open world or in a broad population how can we have sufficient direct or reported experience on everybody? The quantity of potential agents in that population or net that might be excellent partners but that nobody knows enough can be high.

Our claim is that: the larger the population and the ignorance about the trustworthiness of each individual the more precious the role of trust in categories. If I know (through signals, marks, declaration, ...) the class of a given guy/agent I can have a reliable opinion of its trustworthiness derived from its class-membership.

It is clear that the advantages of such cognitive power provided by categories and prejudices does not only depend on recommendation and reputation about categories. We can personally build, by generalization, our evaluation of a category from our direct experience with its members (this happens in our experiments for the agents that later have to propagate their recommendation about). However, in this simulation we have in the trustor (which has to decide whom rely on) only a prejudice based on recommendations about that

category and not its personal experience.
After a certain degree of direct experiences and circulation of recommendations, the performance of the evaluation based on classes will be better; and in certain cases there will be no alternative at all: we do not have any evaluation on that individual, a part from its category; either we work on inferential instantiation of trustworthiness or we loose a lot of potential partners. This powerful inferential device has to be strongly present in WEB societies supported by MAS. We simplify here the problem of the generalization process, of how to form judgement about groups, classes, etc. by putting aside for example inference from other classes (higher or sub); we build opinion (and then its transmission) about classes on the bases of experience with a number of subjects of a given class.
First of all, we want to clarify that here we are not interested in steretypes, but in categories. We define steretypes as the set of features that, in a given culture/opinion, characterize and distinguish that specific group of people.
Knowing the stereotype of an agent could be expensive and time consuming. Here we are just interested in the fact that an agent belongs to a category: it has not to be a costly process and the recognition must be well discriminative and not-cheating. There should be visible and reliable "signals" of that membership. In fact, the usefulness of categories, groups, roles, etc. makes fundamental the role of the *signs* for recognizing or inferring the category of a given agent. That's why in social life are so important coats, uniforms, titles, badges, diplomas, etc. and it is crucial their exhibition and the assurance of their authenticity (and, on the other side, the ability to falsify and deceive). In this preliminary model and simulation let us put aside this crucial issue of indirect competence and reliability *signaling*; let us assume that the membership to a given class or category is true and transparent: the category of a given agent is public, common knowledge.
Differently from [2][10][17] in this work we do not address the problem of learning categorical knowledge and we assum that the categorizzation process is objective.
Similarly to [3], we give agents the possibility to recommend categories and this is the key point of this paper.
In the majority of the cases available in the literature, the concept of recommendation is used concerning recommender systems [1]. These ones can be realized using both past experience (content-based RS)[13] or collaborative filtering, in which the contribute of single agents/users is used to provide group recommendations to other agents/users.
Focusing on collaborative filtering, the concepts of similarity and trust are often exploited (together or separately) to determine which contributes are more important in the aggregation phase [14][18]For instance, in [7] authors provide a system able to recommend to users group that they could join in Online Social Network. Here it is introduced the concepts of *compactness* of a social group, defined as the weighted mean of the two dimensions of similarity and trust.
Even in [11] authors present a clustering-based recommender system that exploits both similarity and trust, generating two different cluster views and combining them to obtain better results.
Another example is [6] where authors use information regarding social friendships in order to provide users with more accurate suggestions and rankings on items of their interest.
A classical decentralized approach is referral systems [20], where agents adaptively give referrals to one another.
Information sources come into play in FIRE [12], a trust and reputation model that use them to produce a comprehensive assessment of an agent's likely performance. Here authors take into account open MAS, where agents continuously enter and leave the system. Specifically, FIRE exploits interaction trust, role-based trust, witness reputation, and certified reputation to provide trust metrics.
The described solutions are quite similar to our work, although we contextualized this problem to information sources. However we do not investigate recommendations with just the aim of suggesting a particular trustee, but also for inquiring categories' recommendations.

II. RECOMMENDATION AND REPUTATION: DEFINITIONS

Let us consider a set of agents $Ag_1, ..., Ag_n$ in a given world (for example a social network). We consider that each agent in this world could have trust relationships with anyone else. On the basis of these interactions the agents can evaluate the trust degree of their partners, so building their judgments about the trustworthiness of the agents with whom they interacted in the past.
The possibility to access to these judgements, through recommendations, is one of the main sources for trusting agents outside the circle of closer friends. Exactly for this reason recommendation and reputation are the more studied and diffused tools in the trust domain [15].
We introduce

$$\mathrm{Rec}_{x,y,z}(t) \qquad (1)$$

where $x,y,z \in \{Ag_1, Ag_2, ..., Ag_n\}$, we call *D* the specific set of agents: $D \circ \{Ag_1, Ag_2, ..., Ag_n\}$
and $0 \le \mathrm{Rec}_{x,y,z}(t) \le 1$
τ, as established in the trust model of [4], is the task on which the recommender *x* expresses the evaluation about *y*.
In words: $\mathrm{Rec}_{x,y,z}(t)$ is the value of *x*'s recommendation about *y* performing the task τ, where *z* is the agent receiving this recommendation. In this paper, for sake of simplicity, we do not introduce any correlation/influence between the value of the recommendations and the kind of the agent receiving it: the value of the recommendation does not depend from the agent to whom it is communicated.
So (1) represents the basic expression for recommendation.
We can also define a more complex expression of recommendation, a sort of *average recommendation*:

$$\sum_{x=Ag_1}^{Ag_n} \mathrm{Rec}_{x,y,z}(t)/n \qquad (2)$$

in which all the agents in the defined set of agents express their individual recommendation on the agent *y* with respect

the task τ and the total value is divided by the number of agents.

We consider the expression (2) as the reputation of the agent y with respect to the task τ in the set D.

Of course the reputation concept is more complex than the simplified version here introduced [5][16].

It is in fact the value that would emerge in the case in which we receive from each agent in the world its recommendation about y (considering each agent as equally reliable).

In the case in which an agent has to be recommended not only on one task but on a set of tasks (τ_1 , ..., τ_k), we could define instead of (1) and (2) the following expressions:

$$\sum_{i=1}^{k} \mathrm{Rec}_{x,y,z}(t_i)/k \qquad (3)$$

that represents the *x*'s recommendation about *y* performing the set of tasks (τ_1,..., τ_k), where *z* is the agent receiving this recommendation.

Imagine having to assign a meta-task (composed of a set of tasks) to just one of several agents. In this case the information given from the formula (3) could be useful for selecting (given the *x*'s point of view) on average (with respect to the tasks) the more performative agent *y*.

$$\sum_{x=Ag_1}^{Agn} \sum_{i=1}^{k} \mathrm{Rec}_{x,y,z}(t_i)/nk \qquad (4)$$

that represents a sort *of average recommendation* from the set of agents in D, about *y* performing the set of tasks (τ_1 , ..., τ_k).

We consider the expression (4) as the reputation of the agent y with respect the set of tasks (τ_1 , ..., τ_k), in the set D.

Having to assign the meta-task proposed above, the information given from the formula (4) could be useful for selecting on average (with respect to both the tasks and the agents) the more performative agent *y*.

A. Using Categories

As described above, an interesting approach for evaluating agents is to classify them in specific categories already pre-judged/rated and as a consequence to do inherit to the agents the properties of their own categories.

So we can introduce also the *recommendations about categories*, not just about agents (we discuss elsewhere how these recommendations are formed). In this sense we define:

$$\mathrm{Rec}_{x,Cy,z}(t) \qquad (5)$$

where $x \in \{Ag_1, Ag_2, ..., Ag_n\}$ as usual, and we characterize the categories $\{C_1, ..., C_l\}$ through a set of features $\{f_{y1}, ..., f_{ym}\}$:

$\forall y \in \{Ag_1, ..., Ag_n\} \exists c_y \in \{C_1, ..., C_l\} \mid (C_y \supseteq \{f_{y1}, ..., f_{ym}\}) \wedge (\{f_{y1}, ..., f_{ym}\} \in y)$

it is clear that there is a relationship between task τ, and the features $\{f_{y1}, ..., f_{ym}\}$ of the C_y category. In words we can say that each agent in *D* is classified in one of the categories $\{C_1, ..., C_l\}$ that are characterized from a set of features $\{f_1, ..., f_m\}$; as a consequence each agent belonging to a category owns the features of that category.

$0 \le \mathrm{Rec}_{x,Cy,z}(t) \le 1$

In words: $\mathrm{Rec}_{x,Cy,z}(t)$ *is the value of x's recommendation about the agents included in category C_y when they perform the task τ,* (as usual *z* is the agent receiving this recommendation).

We again define a more complex expression of recommendation, a sort of *average recommendation*:

$$\sum_{x=Ag_1}^{Ag_n} \mathrm{Rec}_{x,Cy,z}(t)/n \qquad (6)$$

in which all the agents in the domain express their individual recommendation on the category C_y with respect the task τ and the total value is divided by the number of the recommenders.

We consider the expression (6) as the reputation of the category C_y with respect the task τ in the set D.

Now we extend to the categories, in particular to C_y, the recommendations on a set of tasks (τ_1, ..., τ_k):

$$\sum_{i=1}^{k} \mathrm{Rec}_{x,Cy,z}(t_i)/k \qquad (7)$$

that represents *the recommendation value of the x's agent about the agents belonging to the category C_y when they perform the set of tasks (τ_1,..., τ_k).*

Finally, we define:

$$\sum_{x=Ag_1}^{Agn} \sum_{i=1}^{k} \mathrm{Rec}_{x,Cy,z}(t_i)/nk \qquad (8)$$

that represents *the value of the reputation of the category C_y (of all the agents y included in C_y) with respect the set of tasks (τ_1,..., τ_k), in the set D.*

B. Definition of Interest for this Work

In this paper we are in particular interested in the case in which *z* (a new agent introduced in the world) asks for recommendation to *x* ($x \in D$) about an agent belonging to its domain D_x for performing the task τ (D_x is a subset of *D*, it is composed by the agents that *x* knows). *x* will select the best evaluated *y*, with $y \in D_x$ on the basis of formula:

$$\max_{y \in D_x}(\mathrm{Rec}_{x,y,z}(t)) \qquad (9)$$

where $D_x \supseteq \{Ag_1, Ag_2, ..., Ag_m\}$, D_x includes all the agents evaluated by *x*. They are a subset of *D*: $D_x \subseteq D$.

In general *D* and D_x are different because *x* does not necessarily know (has interacted with) all the agents in *D*.

z asks for recommendations not only to one agent, but to a set of different agents: $x \in D_z$ (D_z is a subset of *D*, to which z asks for reputation), and selects the best one on the basis of the value given from the formula:

$$\max_{x \in D_z}(\max_{y \in D_x}(\mathrm{Rec}_{x,y,z}(t))) \qquad (10)$$

$D_z \subseteq D$, z could ask to all the agents in the world or to a defined subset of it (see later).

We are also interested to the case in which *z* ask for recommendations to *x* about a specific *agents' category* for performing the task τ. *x* has to select the best evaluated C_y among the different $C_y \in \{C_1, ..., C_l\}$ x has interacted with (we are supposing that each agent in the world *D*, belongs to a category in the set $\{C_1, ..., C_l\}$).

In this case we have the following formulas:

$$\max_{Cy \in D_x}(\mathrm{Rec}_{x,Cy,z}(t)) \quad (11)$$

that returns the category best evaluated from the point of view of an agent (x). And

$$\max_{x \in D_z}(\max_{Cy \in D_x}(\mathrm{Rec}_{x,Cy,z}(t))) \quad (12)$$

that returns the category best evaluated from the point of view of all the agents included in D_z.

III. Computational Model

A. General Setup

In order to realize our simulations, we exploited the software NetLogo [19].

In every scenario there are four general categories, called Cat1, Cat2, Cat3 and Cat4, composed by 100 agents per category.

Each category is characterized by:

1. an **average value of trustworthiness**, in range [0,100];
2. an **uncertainty value**, in range [0,100]; this value represents the interval of trustworthiness in which the agents can be considered as belonging to that category.

These two values are exploited to generate the **objective trustworthiness** of each agent, defined as *the probability that, concerning a specific kind of required information, the agent will communicate the right information*.

Of course the trustworthiness of categories and agents is strongly related to the kind of requested information/task. Nevertheless, for the purpose of our it is enough to use just one kind of information (defined by τ) in the simulations. The categories' trustworthiness of Cat1, Cat2, Cat3 and Cat4 are fixed respectively to 80, 60, 40 and 20% for τ. What changes through scenarios is the uncertainty value of the categories: 1, 20, 50, and 80%.

B. How the simulations work

Simulations are mainly composed by two main steps that are repeated continuously. In the first step, called **exploration phase**, agents without any knowledge about the world start experiencing other agents, asking to a random 3% of the population for the information P. Then they memorize the performance of each queried agent both as individual element and as a member of its own category.

The performance of a agent can assume just the two values 1 or 0, with 1 meaning that the agent is supporting the information P and 0 meaning that it is opposing to P. For sake of simplicity, we assume that P is always true.

The exploration phase has a variable duration, going from 100 ticks to 1 tick. Depending on this value, agents will have a better or worse knowledge of the other agents.

Then, in a second step (**querying phase**) we introduce in the world a trustor (a new agent with no knowlegde about the trustworthiness of other agents and categories, and that has the necessity to trust someone reliable for a given informative task: in our case τ). It will select a given subset of the population, going from 100% to 5%, and it will query them. In particular, the trustor will ask them for the best category and the best trustee they have experienced.

In this way, the trustor is able to collect information about both the best recommended category and agent.

It is worth noting that the trustor collects information from the agents considering them equally trustworthy with respect to the task of "providing recommendations". Otherwise it should weigh differently these recommendations. In practice the agents are sincere.

Then it will select a randomly chosen agent belonging to the best recommended category and it will compare it, in terms of objective trustworthiness, with the best recommended individual agent (trustee).

The possible **outcomes** are:

- **trustee wins (t_win)**: the trustee selected with individual recommendation is better than the one selected by the means of category; then this method gets one point;
- **category wins (c_win)**: the trustee selected by the means of category is better than the one selected with individual recommendation; then this method gets one point;
- **equivalent result**: if the difference between the two trustworthiness values is not enough (it is under a threshold), we consider it as indistinguishable result. In particular, we considered the threshold of 3% as, on the basis of previous test simulations, it has resulted a resonable value.

These two phases are repeated 500 times for each setting.

IV. Simulations Results

In these simulations we present a series of scenarios with different settings to show when it is more convenient to exploit recommendations about categories rather than recommendations about individuals, and vice versa.

We also present the "all-in-one" scenario, whose peculiarity is that the exploration lasts just 1 tick and in that tick every agent experiences all the others. Although this is a limit case, very unlikely in the real world, it is really interesting as each agent has not a good knowledge of the other agent as individual elements (it experienced them just one time), but it is able to get a really good knowledge of their categories, as it has experienced them as many times as the number of agents for each category. This is an explicit case in which agents' recommendations about categories are surely more informative than the ones about individuals.

In particular, we will represent this value:

$$\frac{c\_win}{c\_win + t\_win} \quad (13)$$

In words, this ratio shows how much categories' recommendation is useful if compared to individual recommendation.

Simulations' results are presented in a graphical way, exploiting 3D shapes to represent all the outcomes. These shapes are divided into two area and represented with two different colors:

- the part over 0.5, in which prevails the category recommendation;
- the one below 0.5, in which prevails the individual recommendation.

These graphs represent an useful view about the utility of the categorial role in the different interactional and social contexts.
For each value of uncertainty, we explored 40 different settings, considering all the possible couple of **exploration phase** and **queried trustee percentage**, where:

- exploration phase ∈ {all-in,1,3,5,10,25,50,100};
- queried trustee percentage ∈ {5,10,25,50,100}.

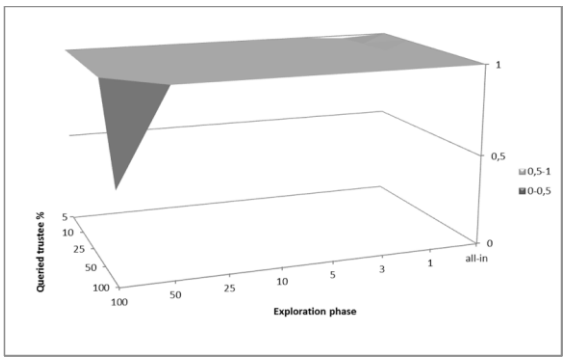


Figure 1. Outcomes for 1% of categories' uncertainty

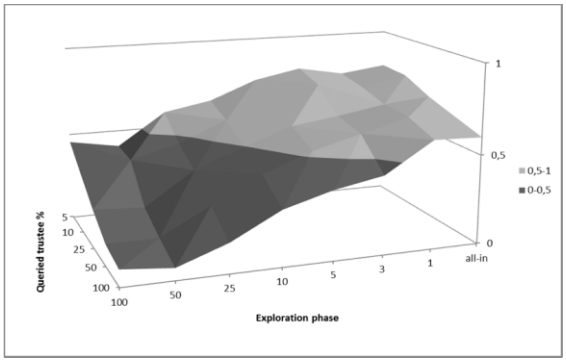


Figure 2. Outcomes for: 20% of categories' uncertainty

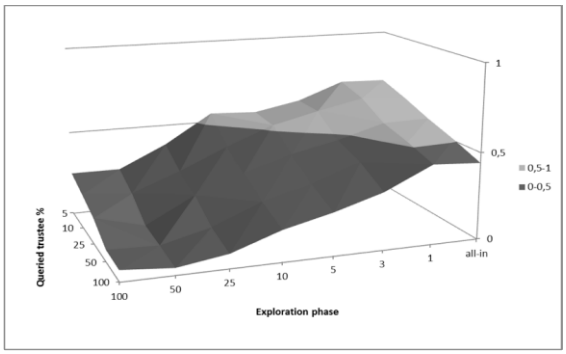


Figure 3. Outcomes for: 50% of categories' uncertainty

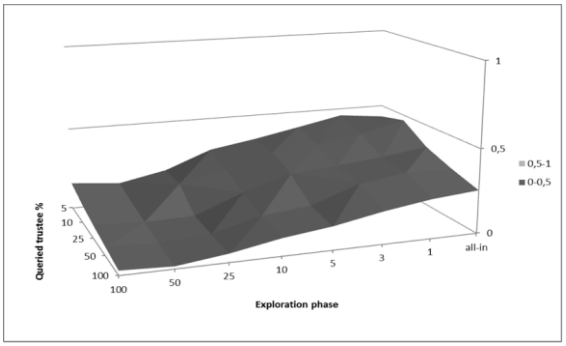


Figure 4. Outcomes for 80% of categories' uncertainty

The part in which category recommendation wins over individual recommendation is represented in light grey. Conversely, the part in which individual recommendation wins is represented in dark grey.
Through these graphs we identify three effects that influence the outcome. The **first effect** is due to categories' uncertainty: the less it is, the more is the utility of using categories; the more it is, the less categories will be useful. It is not possible to notice this effect just looking at one picture. On the contrary, looking at the overal picture one can notice that the curves of the graphs lower, going from a maximal value in **Figure 1** to a minimal value in **Figure 4**.
The **second effect** is due to exploration phase. The longer it is the more individual recommendations are useful; the less it lasts the more category recommendations are useful.
The **third effect** is introduced by the queried trustee percentage, that acts exactly as the exploration phase: the higher the percentage of queried agents, the more individual's recommendations are useful; the less it is, the more categories' recommendations are useful.
The exploration phase's length and the queried agents' percentage occur in all the four graphs and cooperate in determining respectevely the degree of knowledge (or ignorance) in the world and the level of inquire about this knowledge. In particular, with "the knowledge in the world" we intend how the agents can witness the trustworthiness of the other agents or their aggregate, given the constraints defined from the external circumstances (number and kind of interactions, kind of categories, and so on).
In practice, both these elements seem to suggest how the role of categories becomes relevant when either decreases and degrades the knowledge within the analyzed system (before the interaction with the trustor) or is reduced the transferred knowledge (to the trustor).
Let us explain better. The *first effect* shows how the reliability of category's trustworthiness (that will be inherited by its members) depends, of course, from the variability of the behavior among the class members. There may be classes where all the members are very correct and competent, other classes where there is a very high variance: in this last case our betting on a member of that class is quite risky.
The *second effect* can be described with the fact that each agent, reducing the number of interactions with the other agents in the explorative phase, will have relevantly less information with respect to the individual agents. At the same

time its knowledge with respect to categories does not undergo a significant decline given that categories' performances derive from several different agents.

The *third effect* can be explained with the fact that reducing the number of queried trustees, the trustor will receive with decreasing probability information about the more trustworthy individual agents in the domain, while information on categories, maintains a good level of stability also reducing the number of queried agents, thanks to greater robustness of these structures.

Resuming, the above pictures clearly show how, when the quantity of information (about the agents' trustworthiness exchanged in the system) decreases, it is better to rely on the categorial recommendations rather than individual recommendations.

This result reaches the point of highest criticality in the "all-in-one" case in which, as expected, the relevance of categories reach its maximal value.

V. CONCLUSION

Other works [9][2] show the advantages of using categorization to select trustworthy agents. In particular, how it were possible to attribute to a certain unknown agent, a value of trustworthiness with respect to a specific task, on the basis of its classification in, and membership to, one (/or more) category/ies. In practice, the role of generalized knowledge and prejudice (in the sense of pre-established judgment on the agents belonging to that category) has proven to determine the possibility to anticipate the value of unknown agents.

In this paper we have investigated the different roles that recommendations can play about individual agents and about categories of agents.

In this case the new agent introduced (called trustror) has a whole world of agents completely unknown to it, and ask for recommendations to a (variable) subset of agents for selecting an agent to whom delegate a task. The information received regards both individual agents and agents' categories. The informative power of these two kinds of recommendations depends on the previous interactions among the agents and also on the number agents queried by the trustor. However, there are cases in which information about categories is more useful that information towards individual agents. In some sense this result complements the results achieved in [9][2] because here we have a more strict match between information on individual agents and information about categories of agents: We are measuring the quantity of information, about individual agents and categories, for evaluating when is better using *direct information* rather than *generalized information* or, vice versa, when is better using the positive power of prejudice. Our results show how in certain cases becomes essential the use of categorial knowledge for selecting qualified partners.

In this work we have in fact considered a closed world, with a fixed set of agents. This choice was based on the fact that we were interested to evaluate the relationships between knowledge about individual and knowledge about categories, for calibrating their roles and reciprocal influences. In future works we have to consider how, starting from the analysis of this study, could change the role of knowledge about categories in a situation of open world. We have also to consider the cases in which the recommendations are not so transparent but influenced by specific goals of the agents.

ACKNOWLEDGMENTS

This work is partially supported both by the Project PRISMA (PiattafoRme cloud Interoperabili per SMArt-government; Cod. PON04a2 A) funded by the Italian Program for Research and Innovation (Programma Operativo Nazionale Ricerca e Competitività 2007-2013) and by the project CLARA—CLoud plAtform and smart underground imaging for natural Risk Assessment, funded by the Italian Ministry of Education, University and Research (MIUR-PON).

REFERENCES

[1] Adomavicius, G., Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering (TKDE) 17, 734–749, 2005

[2] Burnett, C., Norman, T., and Sycara, K. 2010. Bootstrapping trust evaluations through stereotypes. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10). 241248.

[3] C. Burnett, T. J. Norman, and K. Sycara. Stereotypical trust and bias in dynamic multiagent systems. ACM Transactions on Intelligent Systems and Technology (TIST), 4(2):26, 2013.

[4] Castelfranchi C., Falcone R., Trust Theory: A Socio-Cognitive and Computational Model, John Wiley and Sons, April 2010.

[5] Conte R., and Paolucci M., 2002, Reputation in artificial societies. Social beliefs for social order. Boston: Kluwer Academic Publishers.

[6] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti. Improving Recommendation Quality by Merging Collaborative Filtering and Social Relationships. In Proc. of the International Conference on Intelligent Systems Design and Applications (ISDA 2011) , Córdoba, Spain, IEEE Computer Society Press, 2011

[7] P De Meo, E Ferrara, D Rosaci, and G Sarné. Trust and Compactness of Social Network Groups. IEEE Transactions on Cybernetics, PP:99, 2014

[8] Falcone R, Castelfranchi C, Generalizing Trust: Inferencing Trustworthiness from Categories. In: TRUST 2008 - Trust in Agent Societies, 11th International Workshop, TRUST 2008. Revised Selected and Invited Papers (Estoril, Portugal, 12-13 May 2008). Proceedings, pp. 65 - 80. R. Falcone, S. K. Barber, J. Sabater-Mir, M. P. Singh (eds.). (Lecture Notes in Artificial Intelligence, vol. 5396). Springer, 2008.

[9] Falcone R., Piunti, M., Venanzi, M., Castelfranchi C., (2013), From Manifesta to Krypta: The Relevance of Categories for Trusting Others, in R. Falcone and M. Singh (Eds.) Trust in Multiagent Systems, ACM Transaction on Intelligent Systems and Technology, Volume 4 Issue 2, March 2013

[10] H. Fang, J. Zhang, M. Sensoy, and N. M. Thalmann. A generalized stereotypical trust model. In Proceedings of the 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pages 698–705, 2012.

[11] G. Guo, J. Zhang and N. Yorke-Smith, Leveraging Multiviews of Trust and Similarity to Enhance Clustering-based Recommender Systems, Knowledge-Based Systems, accepted, 2014

[12] Huynh, T.D., Jennings, N. R. and Shadbolt, N.R. An integrated trust and reputation model for open multi-agent systems. Journal of Autonomous Agents and Multi-Agent Systems, 13, (2), 119-154., 2006

[13] P. Lops, M. Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in Recommender Systems Handbook. Springer, pp. 73–105, 2011.

[14] P. Massa, P. Avesani, Trust-aware recommender systems, RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems, 2007

[15] S. Ramchurn, N. Jennings, Carles Sierra, and Lluis Godo. Devising a trust model for multi-agent interactions using confidence and reputation. Applied Artificial Intelligence, 18(9-10):833-852, 2004.

[16] Sabater-Mir, J. 2003. Trust and reputation for agent societies. Ph.D. thesis, Universitat Autonoma de Barcelona.

[17] M. Sensoy, B. Yilmaz, and T. J. Norman. STAGE: Stereotypical trust assessment through graph extraction. Computational Intelligence, 2014.

[18] C. Than and S. Han, Improving Recommender Systems by Incorporating Similarity, Trust and Reputation, Journal of Internet Services and Information Security (JISIS), volume: 4, number: 1, pp. 64-76, 2014

[19] Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

[20] Yolum, P. and Singh, M. P. 2003. Emergent properties of referral systems. In Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'03).

Using AOP neural networks to infer user behaviours and interests

Andrea Fornaia, Christian Napoli, Giuseppe Pappalardo, and Emiliano Tramontana
Department of Mathematics and Informatics
University of Catania
Viale A. Doria 6, 95125 Catania, Italy
{fornaia, napoli, pappalardo, tramontana}@dmi.unict.it

***Abstract*—Generally, users of an 'ego' social network provide personal information and actively participate in groups to discuss some topic. We propose a multi-agent driven system to analyse user behaviour and interests by gathering data related to different activities and show that a more comprehensive identity can be built from sparse data, while possibly reveal the tentative of some users to deceive other people. In our approach, user profiles are given to a profiling agent that retains relevant data by using ANN technologies that find categories for users. Even new users, whose profile is still mostly undefined, are given a 'most-likely' category, therefore the traits of such a category are inferred for new users. Since a group in a social network such as Facebook can be seen as a category, the agent driven system is also able to classify user profiles and recommend new groups users can subscribe to, according to their interests and preferences.**

***Index Terms*—Neural Networks, Social Networks, Artificial Intelligence, Security, Multi-agent Systems.**

I. Introduction

Trust and reliability of data available on social networks are important concerns for both service providers and subscribers. Given the large size of a social network, in terms of subscribers, data exchanged, and number of links (such as friendship, following, membership to groups, endorsements, etc.), it is desirable to have an automatic way to efficiently process data to ensure security and validate at least some contents. User *feature* and *behavioural* analysis are two interesting and important means upon which a solution can be build.

The first step in this direction is to group users into *categories*. Interesting performances have been achieved by systems analysing user interests, however, in general, such systems are only intended for a small context, or for analysing selected users. Even though statistical methods make it possible to characterise features and interests for a single user [1], it is difficult to build a proper analytical model for user interactions due to the vastness of data available in a social network, i.e. number of links, undetermined number of subscriber features, etc. A huge amount of features characterise subscribers, however a relevant portion of values for such features is missing for many subscribers in a real environment, hence a complete formulation of a comprehensive analytical model would be unfeasible [2], [3], [4]. Moreover, the large amount of data and the frequency of changes make the numerous reiterations needed to formulate the analytical model very computationally costly.

Still it is highly desirable to have an automatic analysis that can dynamically incorporate data available on the social network over time. This is fundamental for building advanced services such as e.g. the timely identification of autogenous threats. An automated mechanism should take advantage of soft computing approach such as soft artificial intelligence [5], particle swarm optimisation and positioning [6], [7], [8], evolutionary methods [9], swarm intelligence [10], neural networks [11], etc.... Neural networks has been proven effective for a large number of problems which cannot be solved in terms of a priori mathematical models [12], [13], especially when used with hybrid architectures [14]. We propose an agent driven artificial intelligence system based upon a Radial Basis Probabilistic Neural Network (RBPNN), which is well known for its capability to classify and generalise datasets and can be continuously trained to recognise novel features, hence can easily cope with changing data. The proposed neural network has been embedded into a *Classification Agent* that builds a model out of data coming from user profiles, and handled by other agents, such as a *Profiling Agent* and a *Crawler Agent*, which retain useful data from different parts of an 'ego' social network [15], such as Facebook(R). When analysing a social network, as Facebook, the main difficulties are due to: the unknown number of subscribers, friendship relations, groups, followers, etc.; and the unknown size of data and features for each subscriber. We overcome such difficulties thanks to several agents, which handle data and retain a representation (in our previous analyser version, a big amount of data has been properly processed using a GPU based solution [16], [17], [18], [19]). Specifically, our *Classification Agent*, according to the proposed RBPNN solution, can handle partial data, acting as a modeller for dynamically changing user' s profiles. With our classification approach, we are able to perform early identification of autogenous threats: firstly, an incoherent user profile could be identified when the RBPNN *prediction* of user behaviour, obtained by assigning the user to a category, differs from the actual behaviour; secondly, deception can be revealed by matching user features with others of undesirable categories of users. Moreover, the agent system can use the same classification approach to recommend new groups that fit user interests: this is achieved using group subscriptions as categories, instead of the ones specifically designed by the administrator to classify user behaviour.

Our solution comprises different collaborating agents that make the social network administrators able to classify and monitor the user behaviour for security enforcement, other than enhancing their experience suggesting new groups they can subscribe to, according to their interests.

The rest of this paper is structured as follows. Section II gives the background on the dynamics of a social network. Section III reports about analytical models for classification. Section IV describes the proposed multi-agent system based on RBPNNs. Section V describes the Classification Agent. Section VI and Section VII reports respectively the performed experiments and results. Finally, Section VIII draws our conclusions.

II. Social network dynamics

This work analyses *'ego'* networks and Facebook is considered as a significant representing example. In 'ego' social networks, the *small-world* properties are an important characteristic for the actual social dynamic of the network [20]. Moreover, social networks follow a *scale-free* behaviour [21], i.e. a few nodes (i.e. users) act as important hubs centralising a large number of links, hence data passing through such hubs are widely spread on the network.

A. Clusters of users in a social network

For social networks, such as Facebook, we identify two different kinds of relationships among users. A bidirectional interaction between a pair of users occurs when such a pair exchanges a *friendship*. Additionally, Facebook provides *groups*, i.e. a user is given means to broadcast contents to all the members of the same group where s/he belongs to. We define the mutual exchange of *friendship* between a pair of users as a *strong* connection between the pair, whereas for a pair of users that are *members* of the same group, the membership provides a *weak* connection between such a pair. When a user posts a content into a group, then the resulting one-to-all interaction provides a *weak*, and sometimes *random*, connection with members of the group, who generally share a limited number of interests.

We define the *distance* between a pair of users as follows. When a pair has exchanged a friendship, then the distance is simply 1, otherwise the distance is the minimum count of hops between the pair by following friendship or group connections. Hence, weak connections (available to users belonging to the same group) provide means for information to rapidly flow across users belonging to portions of the social network that have no direct friendship relationship. I.e., weak connections act as *bridges* between users having no friendship, by allowing their *distance* to become equal to 1.

From the friend list of each subscriber we identify *clusters* of users. Clusters consist of users having a higher number of friendships toward users within the same cluster rather than toward users not belonging to the cluster. Analogous to distance between users, we define distance between a pair of clusters as the minimum count of hops between one user on the first cluster from one in the second cluster. Distant clusters can be considered independent parts of the social network that still satisfy the scale-free properties. Clusters generally consist of users sharing a set of interests and activities, and users of the same cluster form a sort of social neighbourhood [22].

Let us suppose that two users belong to different clusters, while being on the same group. When considering the relationship of users and groups, we can see that a group acts as a bridge for the contents to flow from a cluster to another (the clusters of the correspondent users). Hence, different parts of the network become mutually capable of exchanging contents, fostering the small-world behaviour of the social network [23]. In this way, clusters of users, representing parts of the social network, communicate by using weak connections rather than strong ones. Thanks to the said properties of groups we can focus our analysis on a partition of the social network (where a partition is one or several clusters of users), without loosing consistence and pertinence with the social network in its entireness.

B. Existing online social networks

The main difference between a formal scale-free graph and an online social network is given by the percolation of links [24], i.e. in real life, how worth a certain friend is tends to decrease if there is no good reason to maintain the relationship. This decrease of interest is still true even in a social network, however it has no corresponding support in practice. Such a difficulty on the classification of links results into hard to grip data when performing an automatic analysis. Moreover, in a social network user features change steadily, thus it is difficult to determine the correlation between a user and his/her specific field of interests. Generally, for social networks that let users participate in a group, an average subscriber tends to sign into a large number of groups, while only a small amount of such groups are really interesting for the user.

The said wide-spread user behaviour would be difficult to generalise using traditional approaches, which are not noise robust. In turn, automatic selections and suggestions of posts provided by friends or groups become less useful, because of such inaccuracies. Moreover, it is difficult to distinguish between trustworthy users and dishonest or unreliable ones. Even though the user profile can be potentially genuine, differently from social networks, human networks evolve following a homophily law [25] leading a person to connect with others having similar 'real' interests. Hence, the homophily law lets us detect and reason with small, though relevant, differences between social networks and theoretical scale-free networks. Because of such differences, an existing online social network cannot adhere to a simple mathematical model, instead, since the stochastic behaviour typical of human beings is exhibited, an advanced nonlinear model is needed.

Due to the said untrustworthy, erratic, inconstant and unreliable behaviour of users, we maintain that it is paramount to uncover hidden or un-explicit interests, giving a representation of the effective relationships among users. Such (hidden) relationships are significant to find *categories* of users exhibiting

some common traits. Such an identified category would unveil features that can not be directly detected from the user profile.

III. Analytical models

Several generative models can be used to characterise datasets that determine properties and allow grouping data into *classes*. Generative models are based on stochastic block structures [26], on 'Infinite Hidden Relational Models' [27], etc. The main issue of class-based models is the type of relational structure that such solutions describe. Since the definition of a class is attribute-dependent, generally the reported models risk to replicate the existing classes for each new attribute added.

E.g. such models would be unable to efficiently organise (inherit) similarities between (from) the classes 'cats' and 'dogs' as child classes of the more general class 'mammals'. Such attribute-dependent classes would have to be replicated as the classification generates two different classes of 'mammals': the class 'mammals as cats' and the class 'mammals as dogs'. Consequently, in order to distinguish between the different races of cats and dogs, it would be necessary to further multiply the 'mammals' class for each one of the identified race. As a consequence, such models quickly lead to an explosion of classes. In addition, we would either have to add another class to handle each specific use or a mixed membership model, as for crossbred species.

Another paradigm concerns the Non-Parametric Latent Feature Relational Model, i.e. a Bayesian nonparametric model in which each entity has boolean valued latent features that influence the model's relations. Such relations depend on well-known covariant sets, which are neither explicit or known in the case of a social network during the initial analysis.

IV. The multi-agent system

Our aim is to provide to social network administrators a practical and effective tool to predict and monitor user behaviour and interests, both for security purposes and user experience enhancing. Figure 1 shows the agents for our designed system: a *Crawler Agent* periodically and autonomously gathers user information from their social network profiles, other than the list of their group subscriptions. After some preprocessing tasks, data are given to the *Classification Agent* that using the inner RBPNN assigns user profiles to known categories, according to the statistical model built on user information during training phases. Due to the intrinsic dynamics that the social network imposes, this model is constantly and incrementally updated.

The classification results, i.e. the associations between user profiles and categories, are given to the *Verification Agent*, that asks the *Category Agent* to provide the categories already assigned to a specific user (if any), comparing them with the ones just given from the Classification Agent results. If a specific user had no category assigned, the Verification Agent will notify the Category Agent with the newly one found; if instead the user already had a category assigned, but differing from the one just discovered, we could think at this as a clue for an autogenous threat (see Section IV-B) that should

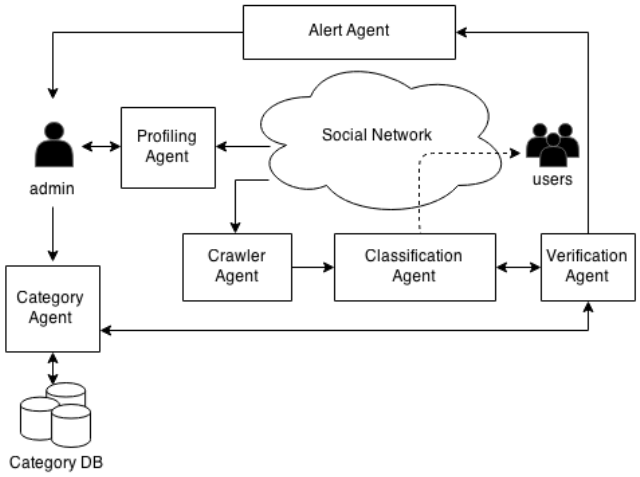


Fig. 1. Schema of the data flow through the agents of the proposed system

be reported to the administrator for further surveys on the user behaviour. This is achieved giving the profile of the threatening user to the *Alert Agent*, that constantly handles all the received notifications, timely warning the administrator with the potentially threats intercepted.

The administrator has also the ability to manually define categories built over the activity information of misbehaving users (see Section IV-C); if one of these categories is assigned to a user profile during classification, the Verification Agent will ask the Alert Agent to notify the administrator with the potential threat detected.

The administrator is then able to gather deeper information on the user activities using the functionalities provided by another agent, the *Profiling Agent*. Using this information, s/he can decide what to do according to social network policies. If the user behaviour is considered not compliant with such policies, the administrator has the ability to use the classification information to automatically identify other users in the network with the same behaviour, asking the Classification Agent to update the inner RBPNN model. On the other hand, if the behaviour of the user can be considered trustworthy, then the new classification label can be simply passed to the Category Agent.

Since we can see a *group* of a social network as a category of users, that gets together people with common interests, we can use the same approach just seen to classify user profiles with the groups that better suit their interests. This type of classification results could be directly provided by the Classification Agent as recommendations for groups that user can subscribe to (see Section IV-D).

A. Computing comprehensive identities

User categories can be chosen by the Classification Agent alone, which is statistically driven, and such categories have a probabilistic meaning that contributes to identify the most appropriate conceivable model for users. The 'model' should

be intended as a kind of representation of the behaviour of a user on the social network. The identified category, provided by the inner RBPNN classifier, can complement and integrate the online identity provided by each subscriber.

Such a comprehensive identity, assigned automatically, can help further understanding user behaviours. To make this solution as independent as possible from the social network data infrastructure, we store and manage additional data with a further agent, that is the Category Agent, but where a more integrated solution is desirable (and conceivable), we can imagine to add such data directly inside the social network user profiles. Once a user belongs to a given category, the administrator can be warned by the Alert Agent to check whether the subscribers linked to a category of misbehaving users are performing activities that conflict with social network policies. When a user posts a content or subscribes to a group, the social network administration is aware of the implicit or explicit choices made beforehand by that user. This 'history' helps understanding whether the current user activity is coherent or appropriate.

Data are continuously sent to the Classification Agent, hence tentative categories identified for a new user are either confirmed or changed according to the recent activities. Hence, more refined alert are given over time.

B. Preventing deception and threats

Theoretically, the RBPNN used by the Classification Agent unveils behavioural patterns that the user is expected to follow. If a user begins to act according to a different behavioural pattern with respect to those for which s/he has been classified, then this variation can be used as an alert that let an administrator monitor him/her and possibly apply some restrictions after a deeper check has occurred. Such an alert is meant to reveal a compromised account that has been stolen.

Once a user account has been confirmed as compromised, either manually or automatically, the supporting system can be set to rise a warning toward all the users that are the target of the activities of the perpetrator, in order to possibly avoid tentative deceptions.

Therefore, the proposed Classification Agent can be used to avoid autogenous threats, such as a misbehaving user or a thief, as much as a wide range of other online frauds and several violations. The more online behaviours are modelled, by training the RBPNN model with existing user data, the more positive and negative activities can be identified by the Classification Agent.

C. Security enforcement

Suppose that a user is disposed toward a bad behaviour on the network, then the Classification Agent would associate such a user with a category previously built by administrators, consisting of other misbehaving users. For building such a category, administrators would simply need to manually flag some selected users, interacting with the Category Agent to store these expert supervised associations.

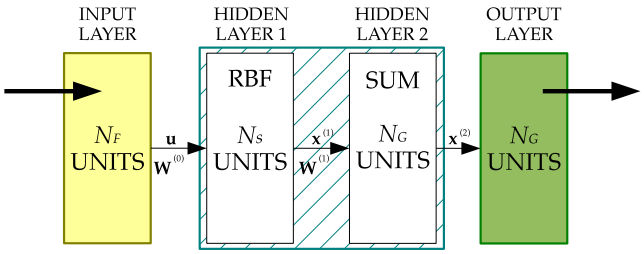


Fig. 2. RBPNN setup values: N_F is the number of considered features, N_S number of analysed subscribers, and N_G desired number of categories.

Although in some moments it would depend on human activities (i.e. administrators), such a control system can then be used to automatically restrict deeper surveys on a small number of possibly dangerous users, so that situations when urgent actions are needed can be timely handled. This automatic selection of users would avert the risk of having to restrain the entirety of subscribers.

D. Group recommendation

Although in this work the RBPNN model has been used to assign categories, which correspond to groups, the term 'categories' has been used on purpose for its more general meaning. The Classification Agent, with its RBPNN model, is able to find and propose non explicit groups, i.e. groups that have not yet been chosen by a subscriber, but which are very likely to be eventually chosen since they match the preferences of the subscriber. In a similar way, this RBPNN can be arranged to select users having an high affinity toward a group. I.e. the RBPNN can be asked to unveil the affinity of a user with a certain category of users, which can be intended not only as a group, but also as a behavioural category.

V. Proposed RBPNN based Classification Agent

Classical models suffer of the incompleteness of the initial input dataset (see Section III). On the other hand, neural networks have been largely used to uncover data classification and find probabilistic categories for data. Therefore, we use Radial Basis Probabilistic Neural Networks (RBPNN), managed by an independent agent, to automatically find *categories* of users, whereby a category reveals common traits for users. Note that *group* of 'ego' networks and social networks, such as Facebook, can be seen as categories, which the RBPNN finds. Our neural network, after being correctly trained, generates a model for the latent user features, and finds users having such features. This is usually considered both an interesting and difficult task [28]. However, the activation functions used for RBPNNs have to meet some important properties required to preserve generalisation abilities and the decision boundaries of Probabilistic Neural Networks (PNN) [29]. The selected RBPNN architecture takes advantage from both PNN topology and Radial Basis Neural Networks (RBNN) used in [30].

In a RBPNN both the input and the first hidden layer exactly match the PNN architecture. In a PNN, each hidden layer neuron performs the dot product of the input vector $\mathbf{u}$ by

a weight vector $\mathbf{W}^{(0)}$, and then gives output $\mathbf{x}^{(1)}$ that is provided to the following summation layer. While preserving the PNN topology, to obtain the RBPNN capabilities, the activation function is a radial basis function (RBF). We name f the chosen RBF, so the output of the first hidden layer for the j-esime neuron is

$$\mathbf{x}_j^{(1)} \triangleq f\left(\frac{||\mathbf{u} - \mathbf{W}^{(0)}||}{\beta}\right)$$

where β is a parameter that controls the distribution shape.

The second hidden layer in a RBPNN is identical to that of a PNN, it just computes weighted sums of the values received from the preceding neurons. The training for the output layer is performed as in a classic RBNN, however since the number of summation units is very small and in general remarkably less than in usual RBNNs, training becomes simplified and speed greatly increased.

The devised topology enable us to distribute different parts of the classification task to different layers (see Figure 2). The first hidden layer of the RBPNN is responsible to perform the fundamental task expected from a neural network, i.e. generalise and build an implicit model. The second hidden layer selectively sums the output of the first hidden layer. The output layer fulfils the nonlinear mapping, such as classification, approximation and prediction.

In order to have a proper classification of the input dataset, i.e. of users into categories, the size of the input layer matches the number N_F of *features*, labelled elements of the dataset (see Section VI), given to the RBPNN, whereas the size of the RBF units matches the number of examined subscribers N_S. The number of units in the second hidden layer is equal to the number of output units, these match the number of categories N_G to be found for the subscribers.

VI. Experimental setup

Since the paramount importance of the classification component in the proposed multi-agent solution, we have deeply tested the performance of the conceived RBPNN classifier used by the Classification Agent. We used a dataset consisting of features, i.e. a trace of the user activities and their preferences, coming from real Facebook profiles. Data for the features that we have been given have a label which is a numerical ID, i.e. the feature itself can not be recognised, however this does not affect the scope of this work nor the analysis performed.

As far as the feature lists is concerned, data provide boolean values. The presence or absence of a specific value is expressed as a boolean flag, e.g. 1 if the user has declared his job or 0 if no job information is given in the profile. Among such boolean values there are mutually exclusive values such as the gender, e.g. 1 if male or 0 if female.

The intrinsic structure of the dataset prevents us from considering only a reduced portion of the feature list for a user. A piece of information is usually largely spread over a certain number of features, e.g. a boolean variable could express if the gender is stated or not, and only if stated another variable could report if the user is male or female; then in case the profile does not state the gender, the latter feature has no meaning and should not be considered. However, since our dataset gives no labels, we can not exclude any feature.

Although data are anonymised, users are identified with a unique ID. Moreover, the memberships of users to groups is indirectly identified from the list of subscribers to each group.

Data intended to be input for our RBPNN have been passed to a preprocessing stage, whereby for each user the corresponding feature list has been paired with the list of group memberships. This enables us to build a statistically driven classifier that identifies the correspondence between user features and their groups.

VII. RBPNN findings

Both user profiles, consisting of features, and user memberships to groups were provided to our RBPNN classifier during the training phase. Therefore, the RBPNN classifier has learnt how to reproduce the correct paths that associate lists of profile features with groups.

Initially, we have asked our RBPNN to reconstruct the groups for 250 users. The RBPNN was able to correctly assign users to the proper groups with only a 5.67% of missing assignments: as a remarkable side effect while a few groups were not found, no false positive was given (see Figure 3). Moreover, if we compare the features for such unclassified users and the average features of their groups, relevant differences can be uncovered with respect to the average (and correctly classified) user. Just for validation purposes, we have performed the same comparison for users with an almost empty profile that the RBPNN could not insert into any category.

Then, we have asked our RBPNN to identify categories for new users. In Figure 3 new users are reported in black or green and are assigned to a group they have not expressed preferences in. For an appreciable percentage of users, i.e. about 20%, the proposed RBPNN has indicated a group that (unknown to the RBPNN) users had membership to. Indeed, a relevant number of the other 80% of user profiles is (almost) empty, therefore no classifier, not only our RBPNN, would manage. On the other hand, how many and which features suffice for a user to be classified depend on the model built by the RBPNN (simply counting the number of empty features is ineffective since they are not equally meaningful).

VIII. Concluding Remarks

With the recent growth of social networks usage, a keen interest for privacy and deception has arisen. In [31], authors describe the results of an extensive comparison between two important social networks such as Facebook and MySpace, showing that the interaction of trust and privacy concerns in social networking sites is not yet understood to a sufficient degree. In [32], authors explore the preservation of privacy and propose a novel method to avoid *neighbourhood attacks*. The authors show that anonymised data can be used to answer aggregate queries accurately.

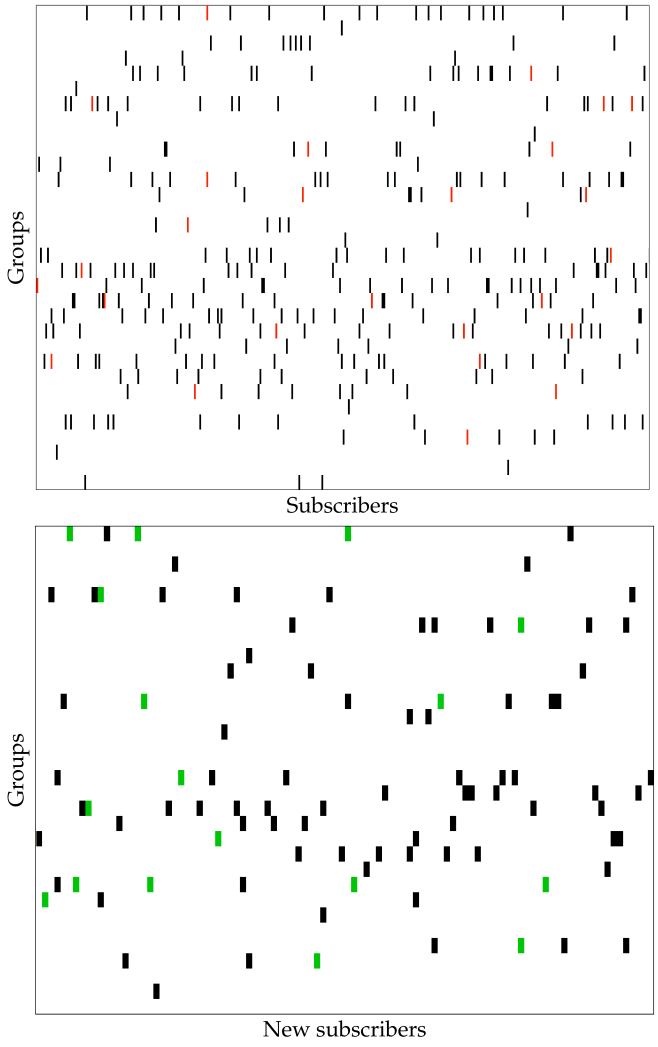


Fig. 3. Top: RBPNN assigned group memberships: correct findings are shown in black, whereas unfound ones are shown in red. Bottom: BPNN assignment to groups for new users: legitimate memberships are in green.

Other previous analyses of data concerning user profiling have taken into account the category of words appearing in texts [33], as well as the user behaviour on-line. The latter solution has been oriented towards an improvement of replica spreading by considering user bandwidth and availability [34]. Moreover, in [35], users have been profiled by observing their interactions with a workflow management system, in a working scenario based on a public administration. All the above profiling strategies can be taken into account and applied into an on-line social network environment for further enriching the classification proposed above.

We have proposed a multi-agent system for automatic analysis of data on a social network and have shown that interesting results can be obtained in terms of the knowledge on the behaviour of users. The proposed solution is based on RBPNN and finds for a user the most similar category (or social network group) s/he could belong to. Once the above solution would possibly be integrated with the servers handling user data, higher security levels could be achieved and the safety of the subscribers would be preserved, e.g. by timely warning administrator to intervene to check and stop autogenous threats.

Acknowledgements

This work has been partially supported by project PRISMA PON04a2 A/F funded by the Italian Ministry of University within PON 2007-2013 framework, and by project PRIME funded by the Italian Ministry of University and Research within POR FESR Sicilia 2007-2013 framework.

References

[1] C. Kiss, A. Scholz, and M. Bichler, "Evaluating centrality measures in large call graphs," in *Proceedings of IEEE Enterprise Computing, E-Commerce, and E-Services*, 2006.

[2] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[3] G. Capizzi, F. Bonanno, and C. Napoli, "A new approach for lead-acid batteries modeling by local cosine," in *Power Electronics Electrical Drives Automation and Motion (SPEEDAM), 2010 International Symposium on*, pp. 1074–1079, IEEE, 2010.

[4] M. T. Hagan, H. B. Demuth, M. H. Beale, *et al.*, *Neural network design*. Pws Pub. Boston, 1996.

[5] F. Bonanno, G. Capizzi, A. Gagliano, and C. Napoli, "Optimal management of various renewable energy sources by a new forecasting method," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2012 International Symposium on*, pp. 934–940, IEEE, 2012.

[6] M. Woźniak, "On positioning traffic in nosql database systems by the use of particle swarm algorithm," in *Proceedings of XV Workshop DAGLI OGGETTI AGLI AGENTI - WOA'2014*, vol. 1260, (25-26 September, Catania, Italy), p. paper 5, CEUR Workshop Proceedings (CEUR-WS.org), RWTH Aachen University, 2014.

[7] C. Napoli, G. Pappalardo, E. Tramontana, Z. Marszałek, D. Połap, and M. Woźniak, "Simplified firefly algorithm for 2d image key-points search," in *2014 IEEE Symposium on Computational Intelligence for Human-like Intelligence*, pp. 118–125, IEEE, 2014.

[8] M. Woźniak and D. Połap, "On some aspects of genetic and evolutionary methods for optimization purposes," *International Journal of Electronics and Telecommunications*, vol. 61, no. 1, pp. 7–16, 2015. DOI: 10.1515/eletel-2015-0001.

[9] M. Gabryel, M. Woźniak, and R. Damaševičius, "An application of differential evolution to positioning queueing systems," *Lecture Notes in Artificial Intelligence - ICAISC'2015*, vol. 9120, pp. 379–390, 2015. DOI: 10.1007/978-3-319-19369-4_34.

[10] M. Woźniak, D. Połap, M. Gabryel, R. K. Nowicki, C. Napoli, and E. Tramontana, "Can we preprocess 2d images using artificial bee colony?," *Lecture Notes in Artificial Intelligence - ICAISC'2015*, vol. 9119, pp. 660–671, 2015. DOI: 10.1007/978-3-319-19324-3_59.

[11] F. Bonanno, G. Capizzi, and C. Napoli, "Some remarks on the application of rnn and prnn for the charge-discharge simulation of advanced lithium-ions battery energy storage," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2012 International Symposium on*, pp. 941–945, IEEE, 2012.

[12] F. Bonanno, G. Capizzi, S. Coco, C. Napoli, A. Laudani, and G. Lo Sciuto, "Optimal thicknesses determination in a multilayer structure to improve the spp efficiency for photovoltaic devices by an hybrid fem—cascade neural network based approach," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2014 International Symposium on*, pp. 355–362, IEEE, 2014.

[13] M. Woźniak, "Fitness function for evolutionary computation applied in dynamic object simulation and positioning," in *IEEE SSCI 2014: 2014 IEEE Symposium Series on Computational Intelligence - CIVTS 2014: 2014 IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems, Proceedings*, (9-12 December, Orlando, Florida, USA), pp. 108–114, IEEE, 2014. DOI: 10.1109/CIVTS.2014.7009485.

[14] F. Bonanno, G. Capizzi, and C. Napoli, "Hybrid neural networks architectures for soc and voltage prediction of new generation batteries storage," in *IEEE international conference on clean electrical power (ICCEP)*, pp. 341–344, 2011.
[15] C. Jones and E. H. Volpe, "Organizational identification: Extending our understanding of social identities through social networks," *Journal of Organizational Behavior*, vol. 32, no. 3, pp. 413–434, 2011.
[16] C. Napoli, G. Pappalardo, and E. Tramontana, "A hybrid neuro-wavelet predictor for qos control and stability," in *Proceedings of AIxIA*, vol. 8249 of *LNCS*, pp. 527–538, Springer, 2013.
[17] C. Napoli, G. Pappalardo, and E. Tramontana, "Using modularity metrics to assist move method refactoring of large systems," in *Proceedings of International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, pp. 529–534, IEEE, 2013.
[18] C. Napoli, G. Pappalardo, E. Tramontana, and G. Zappalà, "A cloud-distributed gpu architecture for pattern identification in segmented detectors big-data surveys," *The Computer Journal*, p. bxu147, 2014.
[19] F. Bonanno, G. Capizzi, G. Lo Sciuto, C. Napoli, G. Pappalardo, and E. Tramontana, "A novel cloud-distributed toolbox for optimal energy dispatch management from renewables in igss by using wrnn predictors and gpu parallel solutions," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2014 International Symposium on*, pp. 1077–1084, IEEE, 2014.
[20] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong, "Analysis of topological characteristics of huge online social networking services," in *Proceedings of World Wide Web*, pp. 835–844, ACM, 2007.
[21] A.-L. Barabási, "Scale-free networks: a decade and beyond," *Science*, vol. 325, no. 5939, pp. 412–413, 2009.
[22] M. Granovetter, "The Strength of Weak Ties," *The American Journal of Sociology*, vol. 78, no. 6, pp. 1360–1380, 1973.
[23] S. Schnettler, "A structured overview of 50 years of small-world research," *Social Networks*, vol. 31, pp. 165–178, July 2009.
[24] N. Schwartz, R. Cohen, D. ben Avraham, A.-L. Barabási, and S. Havlin, "Percolation in directed scale-free networks," *Phys. Rev. E*, vol. 66, p. 015104, Jul 2002.
[25] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, vol. 27, no. 1, pp. 415–444, 2001.
[26] K. Nowicki and T. A. B. Snijders, "Estimation and prediction for stochastic blockstructures," *Journal of the American Statistical Association*, vol. 96, no. 455, pp. 1077–1087, 2001.
[27] Z. Xu, V. Tresp, K. Yu, and H. peter Kriegel, "Infinite hidden relational models," in *Proceedings of Uncertainity in Artificial Intelligence (UAI)*, 2006.
[28] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.
[29] S. O. Haykin, *Neural networks and learning machines (3rd Edition)*, vol. 3. Prentice Hall, 2009.
[30] F. Bonanno, G. Capizzi, G. Graditi, C. Napoli, and G. Tina, "A radial basis function neural network based approach for the electrical characteristics estimation of a photovoltaic module," *Applied Energy*, vol. 97, pp. 956–961, 2012.
[31] C. Dwyer, S. Hiltz, and K. Passerini, "Trust and privacy concern within social networking sites: A comparison of facebook and myspace," in *Proceedings of Americas Conference on Information Systems*, pp. 339–351, 2007.
[32] B. Zhou and J. Pei, "Preserving privacy in social networks against neighborhood attacks," in *Proceedings of Data Engineering*, IEEE, 2008.
[33] C. Napoli, G. Pappalardo, and E. Tramontana, "An agent-driven semantical identifier using radial basis neural networks and reinforcement learning," in *Proceedings of XV Workshop "Dagli Oggetti agli Agenti"*, vol. 1260, CEUR-WS, 2014.
[34] C. Napoli, G. Pappalardo, and E. Tramontana, "Improving files availability for bittorrent using a diffusion model," in *Proceedings of International WETICE Conference*, pp. 191–196, IEEE, 2014.
[35] G. Borowik, M. Wozniak, A. Fornaia, R. Giunta, C. Napoli, G. Pappalardo, and E. Tramontana, "A software architecture assisting workflow executions on cloud resources," *International Journal of Electronics and Telecommunications*, vol. 61, no. 1, pp. 17–23, 2015.

A Case-Study for Sentiment Analysis on Twitter

Paolo Fornacciari, Monica Mordonini and Michele Tomaiuolo
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma
Parma, Italy

e-mail: paolo.fornacciari@studenti.unipr.it, {monica.mordonini, michele.tomaiuolo}@unipr.it

***Abstract* — Microblogging platforms like Twitter can convey short messages to direct contacts, but also to other potentially interested users. They are actively exploited either by individual users or whole organizations and companies. This paper describes some results we obtained from the Social Network and Sentiment Analysis of a Twitter channel, related to a pop music event. Apart from the particular results, a methodology and some guidelines for the automatic classification of Twitter content are discussed.**

Keywords—Social Network; Sentiment Analysis; Hierarchical Classification

I. Introduction

In the common meaning of the term, an online community (or virtual community) is a group of people interested in a particular topic, or that share some ways of thinking, or that in general have some kind of link that brings them together, with the peculiarity that they interface and connect to each other through a data communication network (such as Internet). In this way, they form a social network with unique characteristics: in fact this combination is not necessarily bound to a physical place and anyone can participate wherever he is, with a simple access to networks.

The social networking sites (SNSs), as defined by Boyd and Ellison in [9], are a collection of web-based services that allow users to build a profile within the system and define a list of other users with whom they have some kind of connection. According to Sunden profiles are unique pages where one can "type oneself into being" [32], as the creation of a profile is the minimum condition for joining an SNSs. What makes the SNSs unique is that their purpose is not, in most cases, to allow users to make new friends but the emphasis is on making visible their existing social networks and on the chance to describe them. On the other hand, the specific features of each social network site may depend also on the possible target (social, linguistic or geographic) to which the service is directed. The architecture of social networking platforms is very differentiated. While the most popular platforms are build as essentially centralized systems, other platforms have a distributed architecture [14][15]. The decentralized systems, in particular, often use some notion of trust and cryptography to address the risks of online social networks, which are perceived as serious by many users and have led to incidents [13][35].

Ethnicity, religion, sexual orientation, political beliefs are other factors that have led to the establishment of dedicated social network services, but probably they are also playing an active role in creating and aggregating online communities leveraging the bigger and most popular social networks. This suggests the possibility of new ways to spread information and to influence public opinion. These new scenarios can be better evaluated by a combined observation of the structure and the actual content of the network. This kind of analysis could highlight emerging social behaviors. In [6], for example, the possible differences in the sentiment polarity of female and male users, towards the discussed topic, are examined.

To investigate on the content and on the relations among the actors of a network, it could be useful to contextualize the network itself. In particular, it could be important to consider and inquiry the content of the messages that guide the relationships of the community. It is only through this kind of investigation that we can analyze the semantic meaning of a link, from which we could infer the kind of relationship. This sharpens our description of the social network in many of its facets. A useful tool for such surveys is Sentiment Analysis (SA). SA is a branch of Opinion Mining, that aims to listen and process the data that users post on social media. It is an interdisciplinary field that in recent years has had a significant growth and that makes an extensive use of machine learning techniques. A survey of the main techniques and approaches can be found in [26][7][8]. In [33], it is showed how the information about social relationships can be used to improve user-level sentiment analysis. In [25] Sentiment Analysis is mapped on social media with observations and measurable data; the results highlight the importance of SNSs (i.e. Facebook) as a platform for online marketing.

II. Background

Anthropologist John Barnes was the first to introduce the concept of social network. In 1954 in [5] he described the results of over two years of studies on the composition of classes and social groups in the town of Bremnes (today Bomlo) in Norway. James Mitchel in [22] gave a more sociological and analytical interpretation, describing a social

network as "*a specific set of linkages among a defined set of actors, with the additional property that the characteristics of these linkages as a whole may be used to interpret the social behavior of the actors involved*". Mitchel is a representative of the anthropological school of Manchester, formed in the late 40s, whose founders were the first to use the concept of network in a systematic way.

More simply and more generally, in [34] Wasserman and Faust defined a social network as a finite set of actors and the relation or relations defined on them. This approach is characterized by the priority interest turned to the shape of the networks, rather than their content. According to the exponents of this line of research, the form of social relations largely determines their content. This theory (developed since the 70s at Harvard) lays the foundation for social network analysis (SNA). SNA has the objective to model social structures with different properties, starting from the mathematical theory of graphs and the use of matrix algebra [12]. All these definitions could be summarized by arguing that a social network is a group of individuals (actors) which are connected to each other through different types of social links (relationships), such as family ties, employment relationships, superficial knowledge, common interests. With the development of communication technologies and the growth of online communities, the importance of social networks has increased. The research in SNA finds application in analytical and predictive models used in sociology, anthropology, psychology, computer science and economics [29].

One of the most popular social networks is Twitter (https://twitter.com/). At the end of 2012, the company declared in a tweet: «*There are now more than 200M monthly active @twitter users. You are the pulse of the planet. We're grateful for your ongoing support*.» In this short message, the company announced what many researchers in different domains had already noticed: the information and opinions in our society go through a social network where everyone can sign up and participate. So the analysis of this large amount of data is an exciting challenge for researchers, but it is also crucial for all those who work at different levels in the current information society.

Twitter has been the subject of attention from researchers as early as 2009, for example in [18]. In [24], the authors describe a recent important application for understanding how public sentiment is shaped, how it could be tracked and its polarization with respect to candidates and issues. Another kind of research in the Twitter social network is to combine data source and sentiment analysis. In [2] geo-spatial information related to tweets is used for estimating happiness in the Italian cities. Twitter is also a microblogging platform, so the techniques used generally in Sentiment Analysis and Text Classification must be adapted to the famous 140-character tweet and this opens the way for new issues. Some example of work in this sector are described in [1][21][20][36]. One of the major problems is how to automatically collect a corpus for Sentiment Analysis and Opinion Mining purposes; see, for example, [28][19].

Sentiment Analysis is traditionally focused on the classification of web comments into positive, neutral, and negative categories. But an intelligent and flexible opinion-mining system has to incorporate a deeper analysis of affective knowledge, and detecting emotions [11]. In [10] the correlation among topics and the positive or negative opinions are investigated, to automatically classify the topics themselves. An ontology driven approach is used in [4] to extract rich emotional semantics of tagged texts, by combining available computational and sentiment lexicons with an ontology of emotional categories. A similar approach can be taken into consideration for the detection of feelings in tweets: for example, a taxonomy of feelings can drive the selection of hashtags for the automatic search of tweets with a prevalent sentiment. Such tweets can be used in the training phase of an automatic classifier.

III. Sentiment Analysis on Twitter

In this research work, we built a system for social network and sentiment analysis, which can operate on Twitter data. Twitter is a popular platform for social networking and microblogging, counting hundreds of millions of active users and daily published messages. As a social networking platform, Twitter is structured as a directed graph, in which each user can choose to follow a number of other users (followees), and can be similarly followed by other users (followers). Thus, the "follow" relationship is asymmetrical, it does not require mandatory acknowledgement, and it is essentially used to receive all public messages published by any followee user. As a microblogging service, Twitter is used to publish short messages counting a maximum of 140 characters (tweets), which may contain opinions, thoughts, facts, references to images and other media. Moreover, through the @ symbol it is possible to introduce mentions, i.e. references to other users, and through the # symbol it is possible to introduce hashtags, i.e. references to discussion topics.

Consequently, in our analysis we collected three types of data. The User type represents users' profiles; from Twitter we obtain the following fields: user_id, name, location, num_followers, num_tweets. The Tweet type represents posted messages; from Twitter we obtain the following fields: tweet_id, user_id, message, date. Finally, the Friend type represents the "follow" relationships among users. Apart from data obtained directly from Twitter, we added a field to both tweets and users, to associate a sentiment with them, according to the result of our analysis.

As a communication medium, tweets have a quite peculiar nature. Some distinguishing features of communication on Twitter are related to technical aspects; those include length of text, tags, urls, etc.. Other features may be classified as idiomatic use of the medium, and create a sort of Twitter culture; those features include typical content and most discussed topics, idiomatic expressions, abbreviated forms, etc. For example, a tweet may have the following form:

«*RT @richman wow this is the #happiest day of my life. #happy #glad #icantbelievit :) :D http://t.co/4VEH827bG7*»

The peculiar nature of tweets requires specialized analysis techniques. As a start, a tweet may contain many elements which are not significant for our classification, and can thus be dropped though a filtering process. To polish the message, we defined various filters, which we have applied in a customizable sequence.

A first filter eliminates useless tokens. Removed tokens include: the starting "*RT*" sequence, which indicates a republished messages from a different user (i.e. a retweet); the @ character and the whole following user name; the # symbol, but not the following topic name, which is kept in the message. The topic name is also removed, though, when it coincides with the name of the channel where tweets are collected from.

A second filter applies the language specific rules. It includes an orthographic correction of the message, which is used to remove unknown words, which may not appear in any other tweet (in the example: "*icantbelievit*"). Ideally, the filter at this level should also support stemming and removal of stopwords. However, those operations can be easily performed by Weka, which we used for analysis.

Finally, another filter separates all punctuation symbols from the text, and organizes them as single-character words. However, some typical patterns are kept as aggregates, including smiles sequences, repeated question and exclamation marks.

The final result of the filtering process is a word vector, which is then submitted to the classifier agents. As we have mentioned, our analysis aims at identifying the following classes of messages: undiscriminated, objective, subjective, positive, negative.

The system is organized as a simple hierarchy of agents, mimicking the hierarchy of sentiment classes. In fact, since objective messages have no polarity by definition, the classifier for positive and negative sentiments is only applied to subjective messages. If a message fails to be classified at the first stage, then it simply remains undiscriminated. If it fails to be classified at the second stage, then it is marked as generically subjective.

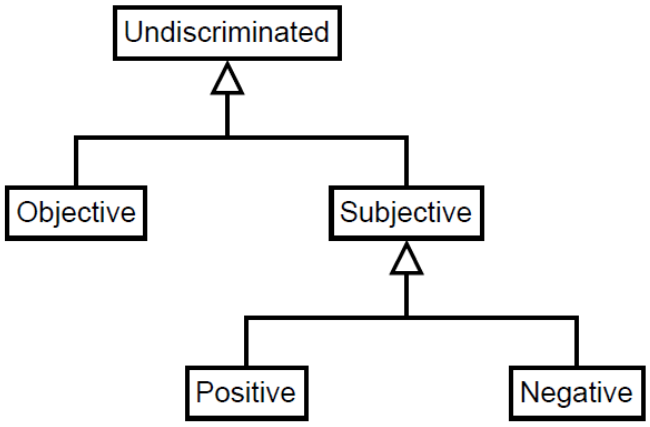


Fig. 1. Hierarchy of basic sentiment classes.

Currently, the classifier agents apply the Multinomial Naive Bayes algorithm, but other methods can be used and different agents can be plugged in the system. However, instead of generating a training set by hand, we aimed at realizing an automated (or at least semiautomated) process for obtaining good training sets.

About the objectivity/subjectivity classifier, we adopted a similar strategy to [27]. In fact, to obtain objective content, we gathered messages generated from popular news agencies. In our tests, we used the following list: *@ABC, @BBCNews, @BBCSport, @business, @BW, @cnnbrk, @CNNMoney, @fox32news, @latimes, @nytimes, @TIME*. To obtain subjective content, instead, we gathered comments directed to the same list of users.

About the polarity classifier, we decided to search for sources of mostly positive or negative messages, respectively. On the one hand, those sources should fit the particular setting of Twitter (short messages, idiomatic expressions, smiles, etc.). On the other hand, they should not be specific to a particular topic or context (sport, music, etc.). Thus, we dropped the idea of collecting messages about particular events, mostly generating either positive or negative sentiments. Instead, we collected messages, using generic yet polar terms as queried hashtags. In particular, we used the following channels to gather positive content: *#adorable, #awesome, #beautiful, #beauty, #cool, #excellent, #great*. We used the following channels to gather negative content: *#angry, #awful, #bad, #corrupt, #pathetic, #sadness, #shame*. Actually, such terms have been chosen quite empirically, taking into account the quality of training sets they generated. But they could be selected from WordNet-Affect [31], SentiWordNet [3], and other affective lexicons, in a more systematic way.

This way, the training set is generated in an automated fashion, as a list of tweets. Each tweet is associated with its supposed class, in accordance to its source. In fact, the training set is not perfect, as it contains messages gathered from public channels. However, a training set of this kind can be generated easily and in a methodical way, from real and updated Twitter messages. In the next section, we will also discuss the quality of results that can be obtained, using it as a basis for sentiment analysis.

The training set can be provided directly to the classifier agents. In the present form, the system is based on Weka, and can thus be configured for performing additional preprocessing steps on the messages, including common TF-IDF transformations, stemming, elimination of stopwords, exclusion of infrequent words, etc.

Currently, we analyze tweets for discriminating the basic classes of objectivity and polarity, at two levels. However, we designed the system for more complex hierarchical classification, with the application of various types of classifiers, as an alternative to current Naive Bayes.

In fact, hierarchical classification has been applied successfully in a number of studies, for information retrieval [30]. It has been proven effective especially in the case of classification over hierarchical taxonomies. Moreover, it has the advantage of being modular and customizable, with respect to the classifiers used at different levels. Using the same probabilistic classifier and a maximum likelihood estimator, instead, does not provide advantages for the hierarchical approach over the flat approach. Mitchell [23] has proved that the same feature sets represent documents in both approaches. Consequently, the whole hierarchical classifier system is equivalent to the corresponding flat system.

Also in the case of sentiment analysis, a hierarchy of classes can be defined [16][4]. Accordingly, hierarchical classification has already been applied to sentiment analysis, too [17].

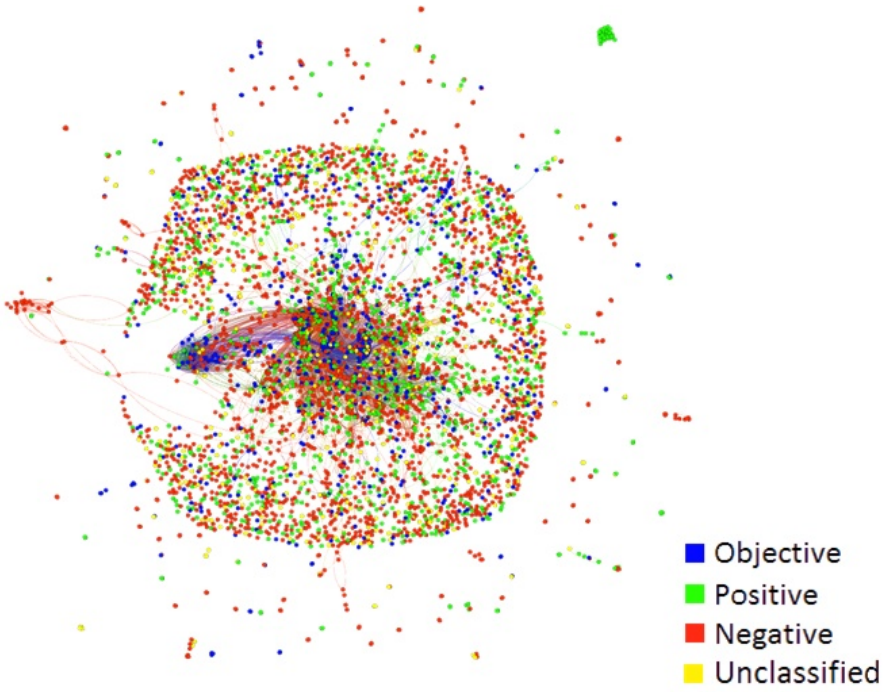


Fig. 2. Communities participating in the *#SamSmith* channel.

IV. A Case-Study: the *#SamSmith* Channel

This section will show the results of the classifiers and the analysis carried out on a case study.

With the above described software, it is possible to obtain some training sets for the classifiers. In our case study, they consist of:

- 86000 instances (polarity)
- 32000 instances (subjectivity)

These instances have been obtained by exploring more than 60 channels on the social network.

In the generated models, the selected features are consistent with our expectations: the typical expressions of a certain feeling (such as smileys, or some words that express appreciation or disgust) show a higher probability of belonging to the class of that feeling, rather than to the class of the opposite sentiment.

The obtained results by the classifiers using cross-validation (with folds = 10) on the training sets showed an accuracy of:

- 77,45% (polarity classifier)
- 79,50% (subjectivity classifier)

These results show that the model of the classifiers contains effective features for the recognition of the sentiment of a message.

The case study which was considered in this work is the social network of the *#SamSmith* channel (the singer who won four awards at the Grammy Awards 2015). The choice of this channel is justified by the strong similarities found between the type of the published tweets and the instances used for training the classifiers. All data were downloaded between 2015-02-02 and 2015-02-10. The awarding of the Grammy took place on 2015-02-08. The network (shown in Fig. 2) consists of a total of 5570 nodes and 6886 arcs.

Looking at the figure, it is possible to notice that the network topology is consistent with the nature of the considered case. In fact, most of the channel consists of independent users (or small groups of users) that express their opinion about the artist; however, in the central part of the network there are some major communities.

As shown in Fig. 3, the prevailing sentiment detected from the classifier is the negative one. Performing an analysis on a

NEG	49,84%
POS	26,52%
OBJ	13,41%
NC	10,23%

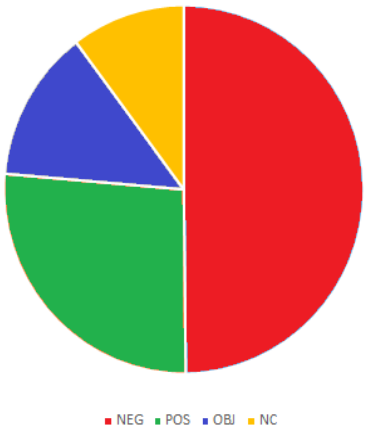


Fig. 3. Sentiment analysis on the *#SamSmith* channel.

Fig. 4. A small community, showing positive sentiment.

sample of tweets in the network, we noticed that many sentences are actually quotes of songs. These messages contain melancholic and sad phrases, and are therefore classified as negative. Considering that a quote is generally an appreciation for the artist, most users classified as negative are actually positive users. This is a typical example of a classic problem of misunderstanding of the SA: the system, while classifying correctly the tweet, misses the assessment of the feeling because it can not evaluate the tweet together with its context.

For evaluating the performances of our system, we conducted a simple survey through a group of persons in our department. In this way, we selected and classified 100 messages that show a clear opinion on the singer. Then, we used those messages as a test. The results of the classifiers showed an accuracy of 84% for the polarity and 88% for subjectivity.

In the network periphery, it is possible to notice a small group of users whose feeling is completely positive (Fig. 4). After a careful analysis of users' tweets in this small group, it was found that these posts are mainly retweets and the original messages are only two. Of these two messages, the first is actually positive, while the other one is objective. This episode shows how some errors of assessment can have important impact on larger communities.

Another kind of analysis we made concerns with the grade of the users. Fig. 5 shows that two nodes have a key role within the social network:

- *@samsmithworld*
- *@TheGRAMMYs*

These users are the main sources of news about the singer Sam Smith and the event Grammy Awards 2015. This explains their importance within the social network which we considered.

V. Conclusion

In this article, we describe some results obtained from the synthesis of Social Network Analysis and Sentiment Analysis applied to the channel *#SamSmith* during the Grammy Awards in 2015. Apart from the particular results, a methodology and some guidelines for the automatic classification of Twitter content have been discussed.

The implemented software allows: (*i*) to get a training set for the classifiers that deal with Sentiment Analysis, and (*ii*) to make a thorough study of the topology of the networks.

The study of the global sentiment within the network has highlighted the typical problems of Sentiment Analysis (irony, sarcasm, lack of information, etc.). Additionally, some peculiar problems of the considered channel were also detected (such as the quotes of songs).

The performances obtained by the classifiers during tests conducted on the training set and the analysis of the case studies have shown good and promising results.

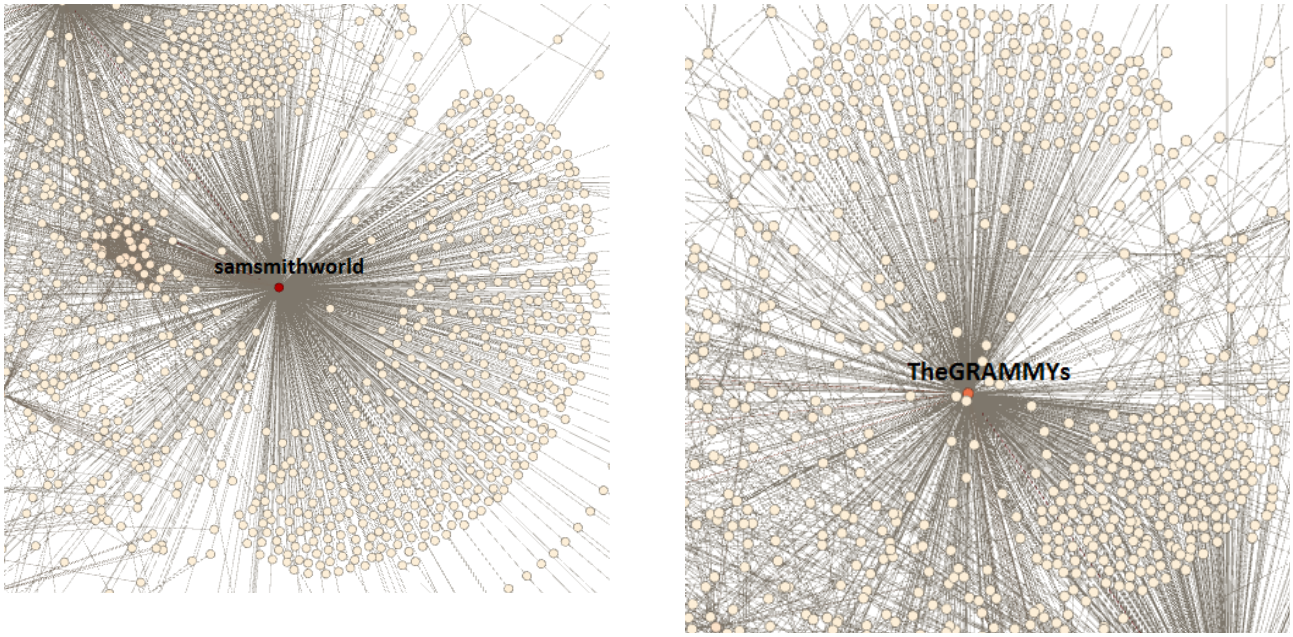


Fig. 5. The most followed nodes in the *#SamSmith* channel.

References

[1] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau. "Sentiment analysis of Twitter data", Procs of the Workshop on Languages in Social Media (LSM '11), pp. 30-38, 2011.

[2] L. Allisio, V. Mussa, C. Bosco, V. Patti, and G. Ruffo, "Felicittà: Visualizing and Estimating Happiness in Italian Cities from Geotagged Tweets," Proc. of ESSEM 2013, Turin, Italy, 2013.

[3] A. E. S. Baccianella and F. Sebastiani, "Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," Proc. of the 7th Conf. on Inter. Language Resources and Evaluation (LREC'10), ELRA, 2010.

[4] M. Baldoni, C. Baroglio, V. Patti, and P. Rena, P. "From tags to emotions: Ontology-driven sentiment analysis in the social semantic web," Intelligenza Artificiale, vol. 6(1), pp. 41-54, 2012.

[5] J. A. Barnes, "Class and Committees in a Norwegian Island Parish, Human Relations", Human Relations, vol 7(1), pp. 39-58, 1954.

[6] A. Bermingham, M. Conway, L. McInerney., N. O'Hare, and F. Smeaton, "Combining Social Network Analysis and Sentiment Analysis to Explore the Potential for Online Radicalisation", in Procs of the 2009 Int. Conf. on Advances in Social Network Analysis and Mining (ASONAM '09). IEEE Computer Society, pp 231-236, 2009.

[7] L. Bing, "Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data," Data-Centric Systems and Applications, 2nd ed., Springer, pp. 1-603, 2011.

[8] L. Bing. Sentiment Analysis and Opinion Mining, Morgan & Claypool Publishers, 2012.

[9] D. Boyd, N. Ellison, "Social Network Sites: Definition, History and Scholarship," Journal of Computed-Mediated Communication, vol. 13 (1), pp. 210-230, 2008.

[10] K. Ca, S. Spangler. Y. Chen; L. Zhang, "Leveraging Sentiment Analysis for Topic Detection," Web Intelligence and Intelligent Agent Technology (WI-IAT '08), IEEE/WIC/ACM Int. Conf. on, vol.1, pp. 265-271, 2008.

[11] E. Cambria, B. Schuller, Y. Xia, C. Havasi , "New Avenues in Opinion Mining and Sentiment Analysis," IEEE Intelligent Systems, vol.28, no. 2, 2013.

[12] P. Carrington, J. Scott, S. Wasserman, Models and Methods in Social Network Analysis, Cambridge University Press, 2005.

[13] E. Franchi, A. Poggi, and M. Tomaiuolo. "Information and password attacks on social networks: An argument for cryptography," Journal of Information Technology Research (JITR), 8(1), 25-42, 2015. doi:10.4018/jitr.2014070104.

[14] E. Franchi, A. Poggi, and M. Tomaiuolo, "Open Social Networking for Online Collaboration," International Journal of e-Collaboration (IJeC), 9(3), 50-68, 2013. doi:10.4018/JITR.2015010103.

[15] E. Franchi, and M. Tomaiuolo, "Distributed social platforms for confidentiality and resilience," Social Network Engineering for Secure Web Data and Services, IGI Global Publisher, pp. 114-136, 2013. doi:10.4018/978-1-4666-3926-3.ch006.

[16] V. Francisco, P. Gervas, and F. Peinado., "Ontological reasoning to configure emotional voice synthesis," Procs of Web Reasoning and Rule Systems, vol. 4524 of LNCS, pp. 88–102. Springer, 2007.

[17] D. Ghazi, D. Inkpen, and S. Szpakowicz, "Hierarchical approach to emotion recognition and classification in texts," Advances in Artificial Intelligence, vol. 6085 of LNCS, Springer Berlin, pp 40-50 2010.

[18] A. Go, L. Huang, and R. Bhayani, "Twitter sentiment analysis," Final Projects from CS224N for Spring 2008/2009 at The Stanford Natural Language Processing Group, 2009. [Online].

[19] S. Hassan, F. Miriam, H. Yulan, and A. Harith, "Evaluation datasets for Twitter sentiment analysis: a survey and a new dataset, the STS-Gold," Proc. of ESSEM 2013, Turin, Italy, 2013.

[20] S. Kiritchenko, X. Zhu, and S. Mohammad, "Sentiment analysis of short informal texts," J. Artif. Int. Res., vol. 50(1), 2014.

[21] A Kowcika, A. Gupta, K. Sondhi, N. Shivhre, R. Kumar, "Sentiment Analysis for Social Media", Int. J. of Advanced Research in Computer Science and Software Engineering, vol. 3(7), 2013.

[22] J.C. Mitchell, Social Networks in Urban Situations: Analysis of Personal Relationships in Central African Towns, Manchester Univ. Press, 1969.

[23] T. Mitchell, "Conditions for the Equivalence of Hierarchical and Flat Bayesian Classifiers". Technical report, Center for Automated Learning and Discovery, Carnegie- Mellon University, 1998.

[24] S. Mohammad, X. Zhu, S. Kiritchenko, J. Martin, "Sentiment, emotion, purpose, and style in electoral tweets", Information Processing & Management, Elsevier, vol 50 (1), 2014.

[25] F. Neri, C. Aliprandi, F. Capeci, M. Cuadros., and T. By, "Sentiment Analysis on Social Media", Proces of the 2012 Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM '12), IEEE Computer Society, pp: 919-926, 2012.

[26] B. Pang, and L. Lee, "Opinion Mining and Sentiment Analysis," Found Trends Inf. Retr., vol. 2 (1-2), pp. 1-135, 2008.

[27] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques". Procs of the ACL-02 conf. on Empirical methods in natural language processing (EMNLP '02), vol. 10, USA, pp. 79-86, 2002.

[28] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," in Proc. of the 7th conf. on International Language Resources and Evaluation (LREC'10). Valletta, Malta, May 2010.

[29] J. Scott, Social Network Analysis, 3rd ed., London and Beverley Hills Sage Publications, 2012.

[30] Jr. Silla, and A. Freitas, "A survey of hierarchical classification across different application domains," Data Mining and Knowledge Discovery, vol. 22(1-2), pp. 31-72, 2011.

[31] C. Strapparava and A. Valitutti. "WordNet-Affect: an affective extension of WordNet," Procs of 4th Int. Conf. on Language Resources and Evaluation (LREC'04), vol. 4, pp 1083–1086, 2004.

[32] J. Sunden, Material Virtualities. Approaching Online Textual Embodiement, New York: Peter Lang., 2003.

[33] C. Tan, L. Lee, J. Tang, L. Jiang, M. Zhou, and P. Li, "User-level sentiment analysis incorporating social networks", Procs of the 17th ACM SIGKDD Int.. Conf. on Knowledge Discovery and Data Mining (KDD '11), ACM, pp 1397-1405. 2011.

[34] S. Wasserman, K. Faust, Social Network Analysis: Methods and Applications, Cambridge University Press, 1994.

[35] M. Tomaiuolo, "dDelega: Trust Management for Web Services," International Journal of Information Security and Privacy (IJISP), 7(3), 53-67, 2013. doi:10.4018/jisp.2013070104

[36] X. Zhu, S. Kiritchenko, and S. Mohammad, "NRC-Canada-2014: Recent improvements in sentiment analysis of tweets," Procs of the Int. Workshop on Semantic Evaluation, Dublin, Ireland, 2014.

Rule-based location extraction from Italian unstructured text

Daniele Caruso, Rosario Giunta, Dario Messina, Giuseppe Pappalardo, Emiliano Tramontana
Department of Mathematics and Computer Science
University of Catania, Italy

Email: {giunta, pappalardo, tramontana}@dmi.unict.it

***Abstract*—Named entity recognition is a wide research topic concerned with the extraction of information from unlabelled texts. Existing approaches mainly deal with the English language, in this paper we present the results of a novel approach specifically tailored to the Italian language. The approach is directed at recognising location names in unstructured texts by several agents based on rules devised for the Italian grammar. Preliminary results show an F1 score up to 0.67.**

***Keywords*-Information Extraction, Named entity recognition, Free Text, Natural Language Processing, Italian Language.**

I. Introduction

Huge amounts of text data are easily available on the World Wide Web. Unfortunately, the great majority of such texts is in the form of unstructured or semi-structured text. Such a reality makes it difficult for both human beings and machines to make a good use of the content of such texts. Information Extraction is concerned with the process of structuring existing texts (both semi-structured and free) so as to single out some parts of text and have them accessed directly by some existing postprocessors [4].

A comprehensive survey of existing approaches [22] show how the Information Extraction community evolved from the seminal approaches since the early '90s, e.g. automatic learning of rules to extract entities [1], maximum entropy models [17], Conditional Random Fields [11], etc. Many, if not all, of these approaches are tested, or developed, on the English language. Moreover, specific analysers have been developed to embed security checks on software programs [10], discover structural properties [3], [12], [14], [18], [19], [23], [24], and perform automatic transformation of programs [2].

We are especially concerned with the problem of named entities extraction from free texts in the Italian language, in particular we are interested in the extraction of location names, i.e. proper nouns of places. Free texts can be of any kind, ranging from dialogues in a movie to fiction prose, thus enacting different constraints, however, in general, a location name is assumed to be written with a capital letter, and common names can be thus considered location names, especially in casual speech, e.g. in *Vediamoci in Dipartimento* (Let's meet at the Department)[1], where the said Department is a shared knowledge between speakers.

Unlike machine learning approaches, both unsupervised and supervised, we proposed a rule-based approach built from simple grammar rules of the Italian language complemented by a dictionary. The process of location names extraction is pursued by means of several specialised agents, each performing an elaboration step and connected in the pipe and filter style (see Figure 1), i.e. the result of the application of a rule removes the bulk of the candidate words, which later have to pass a further screening based, essentially, on a variant of a dictionary comparison.

The text is pre-filtered to remove punctuations symbols and then split into sentences. Each sentence is analysed by up to three rules (See Section III) so to identify word candidates, finally combined to remove false positives. Devised rules are typical Italian language patterns, identifying general contexts where a location can be found, thus the rules are not a simple filtering of words from an existing dictionary.

Preliminary results of the algorithm are encouraging: precision goes up to 0.82 and recall up to 0.92, while the comprehensive F1 score goes up to 0.67.

II. Phase 1: preliminaries

The approach and corresponding tool we have developed works on simple text files, i.e. a web page can be pre-processed beforehand by one of the many converters available to remove HTML tags.

For the devised rules, we make use of an especially compiled Italian lexicon, containing the following classes of words:

- **Articles**. A list of definite articles, e.g. *il* (the).
- **Prepositions**. Both kinds (*semplice* (simple) and *articolata* (composite)) but excluding *con* (with), as it is not used when naming places.
- **Verbs**. A subset of verbs related with places, such as *andare* (to go), *mandare* (to send), *partire* (to leave), *passeggiare* (to (take a) walk).
- **Descriptors**. A list of adverbs frequently related to a place, such as *dentro* (inside), *vicino* (near).
- **Non-places**. Words of various kinds (verbs, adverbs, nouns, etc.) not related to places, but that can appear in grammar structures (defined by the rules we set) as if they were places. E.g. *acido* (sour), *dormire* (to sleep). As

[1]We have decided to use both the original Italian and the translated version of any processed text we show, in order to allow a better appreciation of the proposed approach.

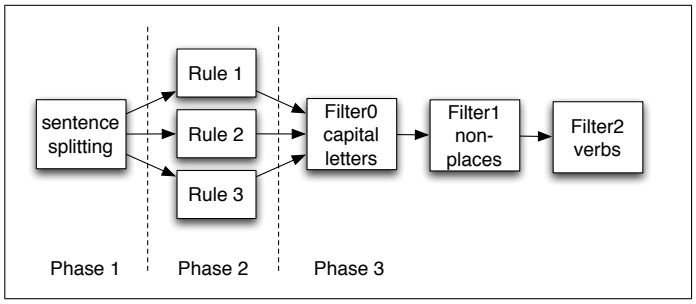


Fig. 1: The agents implementing the pipe and filter model

Verbs	Descriptors	Non-places
abitare (to dwell)	*avanti* (in front of)	*altrimenti* (else)
camminare (to walk)	*dietro* (rear)	*decimo* (tenth)
entrare (to get/come in)	*fianco* (side)	*allora* (then)
uscire (to get out)	*dentro* (inside)	*filosofo* (philosopher)
salire (to go up)	*fuori* (outside)	*molto* (much)
viaggiare (to travel)	*vicino* (near)	*scrivere* (to write)
partire (to leave)	*direzione* (direction)	*camminare* (to walk)
andare (to go)	*esterno* (outer)	*distrarre* (to distract)
indirizzare (to address)	*interno* (inner)	*florido* (prosperous)
raggiungere (to reach)	*lontano* (away)	*bere* (to drink)
risiedere (to inhabit)	*sinistra* (left)	*visitare* (to visit)
visitare (to visit)	*destra* (right)	*ognuno* (everyone)
imboccare (to access)	*adiacente* (adjacent)	*cremisi* (crimson)
arrivare (to arrive)	*vicinanza* (proximity)	*nostro* (ours)
svoltare (to turn)	*ingresso* (entrance)	*riempire* (to fill)
tornare (to go back)	*uscita* (exit)	*durante* (while)
fermare (to stop)	*dirimpetto* (opposite)	*esso* (it)
giungere (to arrive)	*attiguo* (adjacent)	*piatto* (flat)
parcheggiare (to park)		*lucente* (shining)
emigrare (to emigrate)		*spostare* (to move)
decollare (to take off)		*capace* (capable)

TABLE I
SAMPLE VERBS, DESCRIPTORS AND NON-PLACES WORDS

different words may be incorrectly identified as places, the approach assists users in the customisation of the set, by incorporating additional words, so as to exclude (refine) future results.

Table I shows the sample lists of words used in these categories.

In the following sections, such sets will be named after their initial, e.g. we will talk of V as the set of verbs.

Given a text T (read from an input file), the first step is to separate sentences, based on standard Italian grammar rules. T is split at occurrences of one of the symbols in the set of sentence-end punctuation marks, i.e. $\{\text{full-stop}, \text{ellipsis}, \text{exclamation-mark}, \text{question-mark}\}$, all the other punctuation types are removed in order to be processed by the next agents, obtaining a list of sentences. Any other non-letter symbol is ignored, e.g. dollar sign, percent sign, etc.

Each sentence in the input text is further segmented in order to find words, this is accomplished by using the space character as word separator, this applies to any rule we describe. The words found within a sentence are then compared with the entries in the lexicons, according to the different rules described in the next sections.

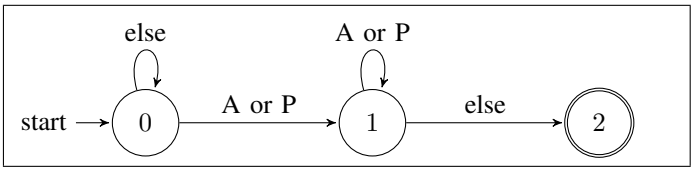


Fig. 2: The FSM implementing Rule 1

III. PHASE 2: RULE-BASED EXTRACTION

We defined three finite state automata, to implement three grammar cases possibly implying the use of a place in the accepting state of the automaton. Each rule identifies a different sentence pattern. The rules are applied at the sentence level, i.e. on a list of words terminated by a punctuation symbol, obtained in phase 1. The tokens (words) are fed to the automaton and, if an accepting state is reached, the current token is marked as a location candidate. If no accepting state is reached no candidate is produced. When a candidate is found, and a sentence still contains some more words, then the automaton restarts from its initial state using the token after the candidate, proceeding until the sentence ends.

The devised rules are independent from one another, so they can be parallelised by running as different agents e.g. on a multicore machine or in different machines coordinated in a Cloud fashion.

The result of each rule application is a list of candidate words, such words are used as input for the next phase (see Section IV) for the definitive labelling. Different rules possibly yields different candidates. Then, a way to use all the said rules is to combine them, hence the candidate words passing the rule filter(s) will be the union of the candidate words determined by each applied rule (see Section V).

A. Rule 1: Da Roma

The first rule, translating "from Rome", is used to identify possible candidate words as a location, and is named, as the other rules, after a typical example of a (part of a) sentence in which a place can be identified.

The automaton (see Figure 2) scans words (tokens) of a given sentence and remains in state 0 until a preposition (P) or an article (A) is found, this condition makes the automaton changes its current state from 0 to 1, and the state remains unchanged unless a different kind of word is found in the next token. Other articles or prepositions do not enable a state change, which is instead triggered by any other kind of word. The final state is reached when a candidate word for a place is found, however many candidates will be ignored afterwards, as described in Section IV.

As a single rule, this yields the highest number of false positives, as the use of an article or a preposition is very common in the Italian language.

B. Rule 2: Vicino a Roma

The second rule accommodates the mentioning of a place name in sentences such as the name of the rule suggests, "Near Rome", as the presence of a Descriptor (see Section II)

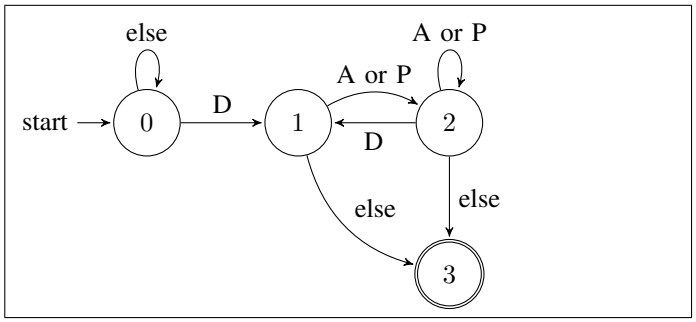


Fig. 3: The FSM implementing Rule 2

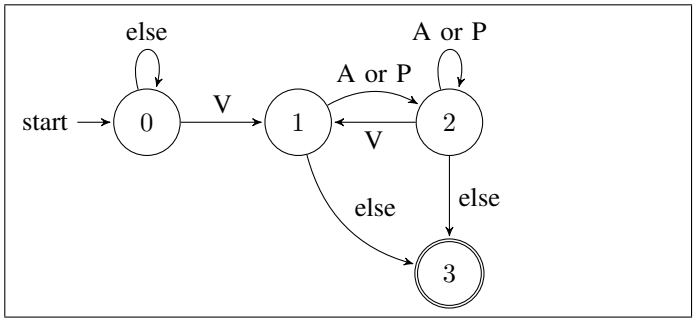


Fig. 4: The FSM implementing Rule 3

is a strong indication that a place will be mentioned in the following text in the same sentence.

Figure 3 shows the Finite State Machine (FSM) to find a candidate place. The automaton starts by reading the words one by one and does not change its initial state (0) until a descriptor is found, then it changes the current state to 1. From state 1 a transition can take place to the state 2, when an article or a preposition is found, or directly to the accepting state 3 in any other case. From state 2 it is possible to return to state 1, if another descriptor is found, or stay in the same state, if more articles or prepositions are found. Finally, the accepting state can be reached by reading any other kind of word.

The accepting state identifies a candidate word as a place, as *Roma* in the rule name.

C. Rule 3: Andando a Roma

While the previous rule uses descriptors as a way to identify a possible place name, this rule is concerned with verbs, such as in the rule name, "Going to Rome".

The FSM implementing the rule is shown in Figure 4. The behaviour of the automaton is the same of Rule 2, where instead of a descriptor a, possibly conjugated, verb is used. The verbs included are only verbs related to movement, and thus usually related to places, such as staying in a place or moving to and from a place. Since a verb can be found in a conjugated form, the check is performed using the Italian version of the stemming algorithm Snowball [21].

The automaton scans the tokens and remains in the state 0 until a verb is found and the current state is changed to state 1. The automaton may change from state 1 to state 2 (and vice versa), by reading a preposition or an article (or reading a verb). Any other words will make the automaton to change into the accepting state 3, i.e. pointing at the current word as a possible candidate for a place name.

IV. Phase 3: Non-place words removal

The candidate words yielded by the application of a rule are further filtered before being labelled as a place name. A candidate, to be considered a place name and thus evaluated as a positive result, has to pass the following filters.

- Filter1: The candidate word is checked against the Non-places lexicon (N). If it exists in N, then it is regarded a False Postitive (FP) and hence discarded. E.g. in the sentence *Andare alla capitale* (To go to the capital city), Rule 2 will suggest *capitale* as a possible place, however it is a common name and thus it will be discarded.
- Filter2: After passing the previous filter (Filter1), all the remaining candidate words are filtered to avoid identifying (conjugated) verbs as places. Once again, the check is performed using a stemming algorithm [21]. To check if a candidate word is a verb, it is stemmed and then concatenated with the three possible suffixes used in Italian verbs (*-are*, *-ere* and *-ire*) so to get the infinitive form of the verb, which is then searched for in the Non-places lexicon. E.g. if the word (a verb) appears in the lexicon, it is discarded as it is not a place name. E.g. in the sentence *Ella esce camminando* (She gets out walking), Rule 3 stays in the state 0 for *Ella*, then goes into state 1 reading the verb *uscire*, however the next token is not an article nor a preposition, thus *camminando* is proposed as a place candidate. However, in this filtering, such a place candidate is recognised as a conjugation of the verb *camminare* and finally discarded. The remaining candidate words are promoted as results.

Any word passing the said filters are labelled as a place name, however such a result may be a True Positive (TP) or a False Positive (FP).

Unstructured text may not be reliably using orthographic conventions, as a text could be a professionally proof-read book or an informal automatic transcription, thus it may or may not use capital letters to address location names. As far as we described our approach, we did not make any assumption on such an orthographic convention, however, experimentally, we found better results when such a convention is satisfied, thus we also provide a further filter:

- Filter0: If the candidate begins with a lower case character, it is not deemed a location name, while it is output as a result if it starts with a capital letter.

As the name suggests, this filter has to be applied before Filter1 and Filter2, as the user sees fit, based on the text to be processed.

V. Discussion

In the next subsections we review the rules and how they relate to the actual grammar writings we examined, and then show the results of the labelling experiments made on different texts.

A. Rule's Assessment

Real case examples. Table II shows several fragments of sentences recognised by our approach, for both TP and FP results, as specified in column 3. The words in italics are the tokens consumed by the automaton for the current rule, while the bold word is the one identified as a place.

Lexicons. As the rules are based on several lexicons, the completeness of such lexicons is essential for a good recognition. In the sentence *"dentro la stanza"* (inside the room), the rule 2 will candidate "stanza", which should not be proposed as a result, as it is not a proper location name. It is a responsibility of the Non-places filter (Filter 1, see Section IV) to recognise that the word is not a location, however, if "stanza" is not in the N lexicon, it will be selected and thus proposed as TP while being a FP.

Repetitions. A simple observation of the FSMs shown in Figures 2 to 4 may lead to a traversal of the states recognising illicit sentences for the Italian grammar. E.g. *"Andando camminare per per Roma"* (Going to walk to to Rome) in which the application of Rule 3 would candidate "Roma" as a place name. Our preliminary studies show that ungrammatical sentences, such as the previous example, are not so frequent unless we factor in informal languages, such as instant messaging or poetic prose/verses.

However, the same rules are capable of recognising ungrammatical sentences appearing in both formal and informal speech. A phrase such as *"Andando a... a... a Roma"* (Going to... to... to Rome) would make the FSM in Figure 4 pointing at "Roma" as an accepting state, even if the sentence is not grammatically correct. As such a sentence can be typical in speeches, e.g. when one speaks while recalling something, an automatic transcription may report such sentences and thus we left the loops in the rules.

Sentence patterns. The rules we are proposing can be considered arbitrary, even if intuitively correct. Thus, before making the actual experiments in labelling, we studied the result of the application of the rules alone on a set of unstructured texts so as to check if such grammar structures had the needed responsiveness degree. I.e. we are interested in the possible paths any automaton may take, given real written texts and not just simple cases (such as the ones in the titles of subsections in Section III, which are correct but also very basic).

Rules have been tested on different kinds of textfiles, both prose and dialogue transcriptions, for a total of 1.2 million of characters. The results are shown in Table III. In each line, the first column is the rule, the second is the sentence pattern found by the rule, the third column is the number of instances of the pattern found in the test corpus. The sentence pattern is identified by the transition in the automaton, e.g. VPA (Verb, Preposition, Article) identifies a sentence such as *Viaggiare per l'Italia* (To travel in Italy), which is decomposed as *Viaggiare*$_V$ *per*$_P$ *l'*$_A$ *Italia*, where the words before *Italia* are being catalogued respectively as [V]erb, [P]reposition and [A]rticle.

Rule	Sentence	Result
1	Il processo che si svolge *a* **Milano**	TP
	(The trial taking place in Milano)	
	I treni a lunga percorrenza *per la* **Sicilia**	TP
	(Long distance trains to Sicily)	
	Il vertice che si terr oggi *a* **Bruxelles**	TP
	(The meeting taking place in Bruxelles)	
	Se *il* **Ministro** in indirizzo non intenda intervenire	FP
	(If the addressed Minister does not mean to intervene)	
2	Scappa *verso il* **Canale**	TP
	(Runs away towards the Channel)	
	All'*interno della* **Basilica** Palladiana	TP
	(Inside the Basilica Palladiana)	
	Mi fanno sedere *accanto a* **Carlo**	FP
	(They let me sit beside Carlo)	
	Sono operative *presso le* **DIGOS** di tutto lo Stato	FP
	(They are operational in the DIGOS (offices) of the State)	
3	*Passando per* **Piazza** Del Popolo	TP
	(Proceed through Piazza Del Popolo)	
	La prima volta che *vedo* **Palermo**	TP
	(The first time I see Palermo)	
	Non ho più *visto* **Carlo**	FP
	(I have not seen Carlo)	
	La soglia richiesta per *entrare in* **Parlamento**	FP
	(The threshold required to get into the Parliament)	

TABLE II
SAMPLE RESULTS USING DIFFERENT RULES

Rule	Sentence Pattern	Occurrences
Rule 1	A	1076
	P	4174
	AA	1
	AP	1
	PA	56
	PP	4
	APA	1
Rule 2	D	58
	DA	41
	DP	73
	DADP	1
	DPDP	2
Rule 3	V	82
	VA	26
	VP	190
	VPA	1
	VPV	1
	VAVP	2
	VPVP	1

TABLE III
DIFFERENT SENTENCE KINDS

Given a rule, a Sentence Pattern such as A is more general than any pattern having A as a suffix e.g. PA. Thus, all the occurences of PA form a subset of the occurences of A. For the experiments (Section V-B) the automata are set to found the longest match.

The preliminary study reported in Table III shows just the number of occurrences for each sentence pattern, it does not show the percentage of TPs or FPs, as this is just a way to check the different transitions in the proposed automata.

B. Experiments

The rules detailed in Section III have been developed in a tool and have been tested on different kinds of unstructured texts: (i) theatrical dialogue transcriptions (texts T2, T3, T4),

(ii) official stenographic transcriptions of political debates (T5) and (iii) news articles (T1). In the two latter cases, the transcriptions are properly capitalized, and thus the Filter0 (see Section IV) has been used in the experiments, while the other texts were all in lower cases and thus only Filter1 and Filter2 have been used in phase 3 (see Section IV).

All the texts used for the experiments have been manually labelled for the location names. In the experiments, all the combinations of the rules have been tested, as shown in the second column. E.g. Rule "2&3" means to put together as a mathematical union the set of candidates gathered by Rule 2 with the set of candidates gathered by Rule 3, using such an union for the filtering agents in phase 3.

The precision metric is computed as a correctness measure, using also the number of False Negatives (FN), as $\frac{TP}{TP+FP}$, while the recall is computed as a completeness metric as $\frac{TP}{TP+FN}$. The F1 score gives the harmonic mean of precision and recall.

There are cases where a rule fails to identify any TP, however this is expected. When an input text does reference a place name by e.g. a motion verb, then only Rule 3 can be able to recognise such places, while Rule 2, concerned with the usage of descriptors, will never be applied.

The results show an interesting F1 score, going up to 0.67 with an average of 0.38. The precision metric goes up to 0.82 in the best case, with a minimum value of 0.27 and an average of 0.45. The recall shows also good results, having a maximum value of 0.92 and an average of 0.51.

While there are cases where very few location names are identified, we deem such preliminar experiments worth expanding, as one of the limitations is the small number of labelled text which we have dealt with.

Text	Rules	TP	FP	FN	F1	precision	recall
T1	1	39	49	9	0,57	0,44	0,81
	2	1	1	47	0,04	0,50	0,02
	3	0	3	48	n/a	n/a	n/a
	1&2	39	50	9	0,57	0,44	0,81
	1&3	39	52	9	0,56	0,43	0,81
	2&3	1	4	47	0,04	0,20	0,02
	1&2&3	39	53	9	0,56	0,42	0,81
T2	1	33	49	6	0,55	0,40	0,85
	2	2	1	37	0,10	0,67	0,05
	3	9	2	30	0,36	0,82	0,23
	1&2	33	50	6	0,54	0,40	0,85
	1&3	34	51	5	0,55	0,40	0,87
	2&3	11	3	28	0,42	0,79	0,28
	1&2&3	34	52	5	0,54	0,40	0,87
T3	1	13	7	6	0,67	0,65	0,68
	2	0	0	19	n/a	n/a	n/a
	3	3	2	16	0,25	0,60	0,16
	1&2	13	7	6	0,67	0,65	0,68
	1&3	14	9	5	0,67	0,61	0,74
	2&3	3	2	16	0,25	0,60	0,16
	1&2&3	14	9	5	0,67	0,61	0,74
T4	1	56	136	5	0,44	0,29	0,92
	2	0	4	61	n/a	n/a	n/a
	3	9	15	52	0,21	0,38	0,15
	1&2	56	140	5	0,44	0,29	0,92
	1&3	56	151	5	0,42	0,27	0,92
	2&3	9	19	52	0,20	0,32	0,15
	1&2&3	56	155	5	0,41	0,27	0,92
T5	1	74	190	84	0,35	0,28	0,47
	2	5	4	153	0,06	0,56	0,03
	3	4	8	154	0,05	0,33	0,03
	1&2	76	194	82	0,36	0,28	0,48
	1&3	75	198	83	0,35	0,27	0,47
	2&3	9	12	149	0,10	0,43	0,06
	1&2&3	77	202	81	0,35	0,28	0,49

TABLE IV
EXPERIMENTAL RESULTS

VI. RELATED WORK

Information extraction has come to be a hot research topic, especially since the availability of huge amounts of data publicly available. An excellent survey on Information Extraction is [22], where the author reviews all the significant existing approaches with a great amount of details. While many different approaches have been proposed, however to the best of our knowledge, little to no effort has been put towards the Italian language.

In [20] named entities are extracted and related to classified newspaper advertisements (in French), using different techniques. They make use of a lexicon to store already known entities, thus once a word is found in an advertisement and in the lexicon it can be automatically tagged as the lexicon suggests. They also use regular expressions for entities such as telephone numbers. Finally, a word spotting algorithm is used to compute a score for unrecognised words, based on the context (i.e. other specialised lexicons). While we also make use of a lexicon, we use it to exclude a candidate, after a rule has yielded one. It would be a trivial and brute force approach to recognise a location name using a lexicon with all existing location names (apart from homonymy), instead the rules we propose allow the discovery of names not already inserted in a lexicon.

The approach presented in [1] shows some similarities with ours. The authors start with sample patterns containing named entities, then identify actual instances of named entities, found names are searched for to automatically identify new patterns and reiterate the process.

A different approach has been proposed in [5], and try to identify named entities by short sequences of words, analysing n-grams statistics obtained on Internet documents. Their Lex method is a semi-supervised learning algorithm based on the assumption that a sequence of capitalised words compound the same name when such a n-gram appears to be statistically more frequent than simple chance.

A data mining approach is presented in [25], especially crafted for geographical names. The algorithm searches for specific keywords and patterns manually constructed and related to geographical names, such as *island of* or *archipelago*. The results are used to train a classifier with respect to the found instances of a pattern.

VII. CONCLUSIONS

We have presented an algorithm devised specifically for the Italian language, based on rules built upon its grammar. The

rules represent grammar pattern, implemented by finite state machines, typically used in both written and spoken language, thus several agents can be coordinated in a pipe and filter style to get an unstructured input text to be filtered by the rules to get candidate places. Preliminary results are promising, as the F1 score reaches a maximum of 0.67, whereas the highest precision and recall are 0.82 and 0.92, respectively.

As possible future work, we aim to connect with our previous research in which we have proposed to improve the modularity of a software system by letting classes assume roles on some design patterns [6]–[9]. The work presented here can foster an approach whereby the automatic processing of the italian language used for program comments can assist in the selection of roles for classes. Moreover, semantic analysis of text can take advantage of neural networks [15] and as a further work a possible approach would aim to recognise text fragments using a soft computing approach [13], [16].

ACKNOWLEDGEMENT

This work has been supported by project PRIME funded within POR FESR Sicilia 2007-2013 framework.

REFERENCES

[1] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of ACM Conference on Digital Libraries (DL)*, pages 85–94, New York, NY, USA, 2000. ACM.

[2] F. Bannò, D. Marletta, G. Pappalardo, and E. Tramontana. Tackling consistency issues for runtime updating distributed systems. In *Proceedings of International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pages 1–8. IEEE, 2010.

[3] A. Calvagna and E. Tramontana. Delivering dependable reusable components by expressing and enforcing design decisions. In *Proceedings of Computer Software and Applications Conference (COMPSAC) Workshop QUORS*, pages 493–498. IEEE, July 2013.

[4] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *IEEE Trans. on Knowl. and Data Eng.*, 18(10):1411–1428, Oct. 2006.

[5] D. Downey, M. Broadhead, and O. Etzioni. Locating complex named entities in web text. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2733–2739. Morgan Kaufmann Publishers Inc., 2007.

[6] R. Giunta, G. Pappalardo, and E. Tramontana. Using Aspects and Annotations to Separate Application Code from Design Patterns. In *Proceedings of Symposium on Applied Computing (SAC)*. ACM, 2010.

[7] R. Giunta, G. Pappalardo, and E. Tramontana. Aspects and annotations for controlling the roles application classes play for design patterns. In *Proceedings of Asia Pacific Software Engineering Conference (APSEC)*. IEEE, 2011.

[8] R. Giunta, G. Pappalardo, and E. Tramontana. AODP: refactoring code to provide advanced aspect-oriented modularization of design patterns. In *Proceedings of Symposium on Applied Computing (SAC)*. ACM, 2012.

[9] R. Giunta, G. Pappalardo, and E. Tramontana. Superimposing roles for design patterns into application classes by means of aspects. In *Proceedings of Symposium on Applied Computing (SAC)*. ACM, 2012.

[10] R. Giunta, G. Pappalardo, and E. Tramontana. A redundancy-based attack detection technique for java card bytecode. In *Proceedings of International WETICE Conference*, pages 384–389. IEEE, 2014.

[11] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[12] M. Mongiovi, G. Giannone, A. Fornaia, G. Pappalardo, and E. Tramontana. Combining static and dynamic data flow analysis: a hybrid approach for detecting data leaks in Java applications. In *Proceedings of Symposium on Applied Computing (SAC)*. ACM, 2015.

[13] C. Napoli, G. Pappalardo, and E. Tramontana. A hybrid neuro-wavelet predictor for qos control and stability. In *Proceedings of AIxIA*, volume 8249 of *LNCS*, pages 527–538. Springer, 2013.

[14] C. Napoli, G. Pappalardo, and E. Tramontana. Using modularity metrics to assist move method refactoring of large systems. In *Proceedings of Complex, Intelligent and Software Intensive Systems (CISIS)*. IEEE, 2013.

[15] C. Napoli, G. Pappalardo, and E. Tramontana. An agent-driven semantical identifier using radial basis neural networks and reinforcement learning. In *Proceedings of XV Workshop "Dagli Oggetti agli Agenti"*, volume 1260. CEUR-WS, 2014.

[16] C. Napoli, G. Pappalardo, and E. Tramontana. Improving files availability for bittorrent using a diffusion model. In *Proceedings of International WETICE Conference*, pages 191–196. IEEE, 2014.

[17] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *IJCAI workshop on machine learning for information filtering*, volume 1, pages 61–67, 1999.

[18] G. Pappalardo and E. Tramontana. Automatically discovering design patterns and assessing concern separations for applications. In *Proceedings of Symposium on Applied Computing (SAC)*. ACM, 2006.

[19] G. Pappalardo and E. Tramontana. Suggesting extract class refactoring opportunities by measuring strength of method interactions. In *Proceedings of Asia Pacific Software Engineering Conference (APSEC)*, pages 105–110. IEEE, December 2013.

[20] R. A. Peleato, J.-C. Chappelier, and M. Rajman. Automated information extraction out of classified advertisements. In *Natural Language Processing and Information Systems*, pages 203–214. Springer, 2001.

[21] M. F. Porter. Snowball: A language for stemming algorithms, 2001. *URL http://snowball. tartarus. org/texts/introduction. html*, 2009.

[22] S. Sarawagi. Information extraction. *Found. Trends databases*, 1(3):261–377, Mar. 2008.

[23] E. Tramontana. Automatically characterising components with concerns and reducing tangling. In *Proceedings of Computer Software and Applications Conference (COMPSAC) workshop QUORS*. IEEE, 2013.

[24] E. Tramontana. Detecting extra relationships for design patterns roles. In *Proceedings of AsianPlop*. March 2014.

[25] O. Uryupina. Semi-supervised learning of geographical gazetteers from the internet. In *Proceedings of the HLT-NAACL 2003 Workshop on Analysis of Geographic References - Volume 1*, pages 18–25. Association for Computational Linguistics, 2003.

Protocols with Exceptions, Timeouts, and Handlers: A Uniform Framework for Monitoring Fail-Uncontrolled and Ambient Intelligence Systems

Davide Ancona, Daniela Briola and Viviana Mascardi
Department of Informatics, Bioengineering, Robotics and System Engineering
University of Genova, Via Dodecaneso 35, 16146, Genova, ITALY
Email: {davide.ancona,daniela.briola,viviana.mascardi}@unige.it

***Abstract*—This paper describes an approach for designing, formalizing and implementing *sentinels* that detect errors in *fail-uncontrolled multiagent systems*, and *controllers* that identify particular situations in *ambient intelligence (AmI) systems*. The formalism we use for representing the expected patterns of actions along with exceptions, timeouts, and their handlers, is that of *constrained global types* extended with features for dealing with these new constructs. We provide the syntax and semantics of the extended constrained global types and examples of their use, in the different contexts of fail-uncontrolled and AmI systems.**

I. Introduction

Multiagent protocols are usually intended as a means to regulate communicative interactions among agents. The literature on agent interaction protocols is huge and despite its long tradition dating back to the dawn of the research on agents and multiagent systems [1], [2], it is still a hot research topic [3], [4], [5], [6], [7].

In this paper we adhere to a general definition of protocols as a means to define *legal sequences or concurrent combinations of actions without regard to the meanings of those actions* [8]: our protocols define patterns of actions in the environment, rather then just conversations.

The purpose of our research is to provide the multiagent system developer with a framework for designing, formalizing and implementing the following monitoring agents on top of existing agent environments:

1) *sentinels* that detect errors in *fail-uncontrolled multiagent systems* and
2) *controllers* that identify particular situations in *ambient intelligence (AmI) systems*.

Despite the different contexts where sentinels and controllers are expected to operate, both must be able to monitor the system's behavior and to properly deal with failures and exceptional situations that may arise during the multiagent system (MAS for short) functioning.

The language we propose for formalizing the expected patterns of actions in the environment is that of *constrained global types* [9], [10], [11] extended with features for dealing with exceptions, timeouts, and their handlers. These extensions are used to describe specific situations that must be managed before they lead to an error or to a dangerous or unwanted situation.

The basic mechanism for runtime verification that an event which took place in the environment is compliant with the protocol has already been described in our previous works [9], [6], [12], where we presented a working framework for Jason [13], chosen as a paradigmatic example of logic-based MAS language and framework, and for JADE [14], chosen because of its wide adoption both in academic and industrial environments.

In this paper we describe the extensions to *constrained global types* to cope with the newly introduced concepts such as events, timeouts etc, as described in Section III. Although the updating of the existing and working prototypes for Jason [9] and JADE [6] to include these new features is still under way, we are confident on the positive outcomes of our efforts. In fact, we already implemented an SWI Prolog extended interpreter that properly deals with these new features and which is presented in Section III, and we tested it as a standalone application (Section IV). The missing final step is to substitute the interpreter for monitoring MASs that is already integrated on top of Jason (resp. JADE) with this new, enhanced interpreter. This requires to take care of minor syntactic issues so it does not come completely for free, but is should not even raise relevant technical problems.

We consider two different types of systems, fail-uncontrolled MASs and AmI systems, to prove the usability of constrained global types for representing patterns of events (that we also name situations when referred to AmI systems, or protocols, coherently with the terminology used in the literature about security) that should drive the runtime monitoring activity of sentinels and controllers.

In *fail-uncontrolled multiagent systems*, the protocol modeling the normal expected behavior of the MAS is *explicitly represented* and involves events of any type (not limited to communicative ones); sentinels observe the system under monitoring and execute user-defined, domain-dependent code when a deviation from the expected behavior takes place. Deviations are modeled as *exceptions* to the normal flow of actions that the protocol models, and the user code must implement the exception handlers.

In *AmI systems*, where usually there is no explicit protocol driving the agents' behavior, but rather agents are free to act as they wish in any ambient, we only model particular situations that must be identified as soon as they arise: in this way we can respect the autonomy of agents and deal with unforeseen

events, but we are still able to check that specific situations are identified as "wrong" ones and their consequences are avoided.

By exploiting the abstraction of "event type" supported by constrained global types, we can model events which are observed in the monitored system and that are relevant for the monitored application, but also events which are not interesting and hence should be discarded (*uninteresting events*). *Unexpected exceptions* can be captured and managed as well.

Since sentinels and controllers are agents like any other one, they can be "part of the protocol" as well, hence being both promoters and targets of the monitoring activities[1]. For example, if the protocol states that – when an exception has been detected – the controller must always send a message to the agent that caused it, and this does not happen, the controller can realize that it is violating the protocol and should repair its own code.

In order to avoid bottlenecks, *the monitoring activities may be distributed among many different and independent sentinels (resp. controllers)*, each observing a subset of the events based on their type, the location where they take place, or other criteria. An approach for distributing the monitoring activity by projecting the protocol onto subsets of agents involved in it has been explored in [15].

The proposed approach is general enough to be implemented on top of any MAS or AmI framework where events can be observed and translated into a suitable representation, amenable for formal reasoning.

The paper is structured in the following way: Section II provides the background to this work. Section III describes the extension of constrained global types with exceptions, timeouts, and their handlers. Section IV provides examples in different contexts. Section V concludes.

II. Background and related work

The classification of system failures and the definition of fail-uncontrolled system we refer to, is that provided in [16], which in turn refers to [17]:

> *The most generally accepted failure classification can be found in [17]:*
>
> 1) *A crash failure means a component stops producing output; it is the simplest failure to contend with.*
> 2) *An omission failure is a transient crash failure: the faulty component will eventually resume its output production.*
> 3) *A timing failure occurs when output is produced outside its specified time frame.*
> 4) *An arbitrary (or byzantine) failure equates to the production of arbitrary output values at arbitrary times.*
>
> *Given this classification, two types of failure models are usually considered in distributed environments:*
>
> - *fail-silent, where the considered system allows only crash failures, and*
> - *fail-uncontrolled, where any type of failure may occur.*

In such a context,

> *a sentinel is an agent, and its mission is to guard specific functions or to guard against specific states in the society of agents. The sentinel does not partake in domain problem solving, but it can intervene if necessary, choosing alternative problem solving methods for agents, excluding faulty agents, altering parameters for agents, and reporting to human operators. Being an agent, the sentinel interacts with other agents using semantic addressing. Thereby it can, by monitoring agent communication and by interaction (asking), build models of other agents. It can also use timers to detect crashed agents (or a faulty communication link)* [18].

Exceptions can be informally defined as:

> *[...] abnormal conditions that arise during the execution of a process. The importance of exceptions stems from the simple fact that they are an essential feature of real-life processes. Businesses, for example, entertain exceptional requests from customers in the interest of better customer service. Conversely, exceptions that occur in a process may lead to poor user satisfaction. Therefore, businesses must accommodate exceptions in their underlying systems and their interactions with other businesses. For concreteness, let us review a classification of exceptions proposed by Eder and Liebhart [19]:*
>
> 1) *Basic failures, which are system-level failures such as network failures.*
> 2) *Application failures, such as database transaction failures.*
> 3) *Expected exceptions, which are deviations from the normal flow that occur infrequently but often enough to be incorporated into the process model.*
> 4) *Unexpected exceptions, which are not modeled and hence require a change in the design of the process when they are discovered.*
>
> *An alternate classification of exceptions distinguishes among system level exceptions, programming language exceptions, and pragmatic exceptions. Among these, pragmatic exceptions are the most acute and the most difficult to handle* [20].

According to our analysis of the state of the art in fault tolerant multi-agent systems, we can distinguish two broad classes of approaches:

1) approaches where the multiagent protocol is represented in an explicit way, and fault tolerance amounts to handling exceptions due to a non-compliant behavior w.r.t. the protocol [8], [20]; in such approaches, multiagent protocols are usually limited to express constraints on communicative actions;
2) approaches where no protocol exists, and fault tolerance amounts to detecting crashed agents and replicate them in a transparent way [21], [16].

[1]This "introspective" model can work only if sentinels and controllers perform the very same monitoring activity whatever the agent being monitored, including themselves.

Our approach falls in the first category.

With respect to *AmI systems*, the idea of implementing them using an agent-based approach dates back to the late nineties. Among the very first projects in this area we may mention the Intelligent Room project at MIT [22] that concentrated on making the room responsive to the occupant by adding intelligent sensors to the user interface, the ACHE system [23] which also aimed at energy saving and increased personal comfort by learning the user preferences automatically by observing the behavior of the persons in the building, and the work by Paul Davidsson and Magnus Boman [24] who used multiagent principles to control building services, with agents that decompose systems by function rather than behavior. The synergy between ambient intelligence and intelligent software agents has become stronger and stronger, and nowadays there are tens of papers and projects in this area. Among the most relevant ones, we may mention [25], [26], [27]. The impressive growth in the last years [28], [29], [30], [31], [32] is also witnessed by the Multi-Agent Systems and Ambient Intelligence (MASAI) special track at the Practical Applications of Agents and Multi-Agent Systems (PAAMS) 2014 and 2015 conferences.

Among the most recent works in the MAS & AmI area, the closer to our proposal is [33]. That paper presents a concrete software architecture dedicated to AmI features and requirements. The proposed behavioral model, called Higher-order Agent (HoA) captures the evolution of the mental representation of the agent and the one of its plan simultaneously. Plan expressions are written and composed using a formal algebraic language named AgLOTOS which is very similar to our formalism based on constrained global types although less expressive because of the lack of concatenation and recursion. In [33], the planning process is improved with two new services which provide a guidance for the mental process and a model checking approach to analyze some temporal properties over the agent plan. Both services are based on a transition system called Contextual Planning Systems that, similarly to the traces of our constrained global types, can represent the different execution traces the agent could perform from a given HoA configuration, assuming the information context of the HoA configuration holds. With AgLOTOS it is possible to handle actions and plans failures, as we do. The main differences between the approaches, besides the lower expressive power of AgLOTOS w.r.t. the formalism that we adopt, are that our approach is more general than the one discussed in [33] and it can be implemented on top of any existing MAS framework, as the only requirement is that the sentinels and controllers can observe events taking place in the environment, but the approach based on AgLOTOS is in a more mature development stage and has been experimented on the field in a Smart-Campus Project.

Finally, we may observe that our approach of supervising the ongoing activities of the MAS, being it an *AmI system* or a more generic distributed open system, is similar to the "Monitoring by overhearing" approach presented in [34], where an overhearing agent monitors the exchanged communications between the system's agents and uses the observed communications to independently assemble and infer the needed monitoring information.

III. Multiagent Protocols with Timeouts, Exceptions, and Handlers

Our proposal is general enough to be applied to many languages and environments for MASs. The only needed requirement is that some agents (the sentinels or controllers) can be given the power and the capability to observe (some of) the actions taking place in the environment.

As far as sentinels are concerned, the four types of failure identified in [17] are all resorted to a behavior that does not comply with the multiagent protocol:

- a timing failure can be detected thanks to implicit and explicit timeouts on the actions: if an expected action does not take place within the given timeout, then a time failure takes place. More successive timing failures may mean either an omission or a crash failure;
- a byzantine failure can be detected by verifying that the occurred action was (or was not) expected by the protocol in the current state and time.

Sentinels implementation verifies at run-time that perceived events are compliant with the protocol. This verification activity is necessary when the multiagent protocol is designed and the MAS is tested, but cannot guarantee any fault tolerance, as the output of the verification in any given time instant is just "yes" or "no". In order to capture deviations from the expected flow and manage them, so to possibly recover to some acceptable state when the verification fails, the sentinel must pro-actively intervene by executing some recovery plan. We have designed the mechanism that allows a sentinel to capture the expected failures of the protocol by means of *exceptions*, and the recovery intervention by means of *exception handlers*.

In this section we present the formalism of constrained global types introduced by Ancona, Mascardi et al. [9], [10], [11], extended to cope with any kind of observable event instead of only communicative actions and to model exceptions and timeouts. Although never exploited, dealing with uninteresting events was already possible with the original versions of the formalism and hence does not represent a real extension to it.

In the sequel, we will take the perspective of constrained global types as a powerful means to specify multiagent protocols, but they can be used to specify AmI situations as well, and for the same reasons. For readability, we will avoid repeating "or AmI situations", but we emphasize that our assertions hold for them as well.

Intuitively, a constrained global type represents the state of an multiagent protocol from which several transition steps to other states (that is, to other constrained global types) are possible, with a resulting event.

Exceptions are the mechanism we have introduced in order to cope with byzantine failures, whereas timeouts are used to detect and cope with omission, timing and crash failures.

A. Syntax

Event. An event e is any observable event taking place in the MAS environment, including communicative actions,

actions performed by agents, agent location and move, and actions performed by artifacts. We do not face the transduction problem and assume that events are translated into symbols that agents can manipulate by some mediator between the agents and the environment. For the purposes of our work, events are symbolic expressions.

Example 1:

```
transport(policeman(marcus), prisoner(alice),
from(jail), to(room1), by(car)).
```

Example 2:

```
collect_parcel(agent2, parcel3, parcelweight(4, kg)).
```

The event "transport" is characterized by the involved entities (the policeman and the prisoner, identified by their names), the departure and arrival places and the means of transport.

Event type. A "normal" event type η is a predicate on events. Its interpretation is the set of events that verify η; we write $e \in \eta$ to mean that η is true on e, and we also say that e has type η.

Example 1:

```
transport(policeman(marcus), prisoner(alice), from(jail),
to(room1), by(car)) ∈ move(alice, jail, room1).
```

Example 2:

```
collect_parcel(agent2, parcel3, parcelweight(4, kg))
∈ collect_parcel.
```

With respect to the actual event that took place in the environment and that was transduced into a symbolic form, the event type may abstract some details which are not relevant for the sentinel monitoring activities (like in Example 2), and can be identified by a different predicate with different arguments (like in Example 1, where "move(...)" is the event type of "transport(...)" event).

Exception type. An exception type $\mathcal{E}$ is – from a logical point of view – a quadruple <type of the event that will fire the exception, type of the exception, pointer to the exception handler, arguments of the exception handler>. We represent exceptions in a more compact way, as a couple whose second element has the type of the exception as functor, and the pointer to the exception handler along with its arguments as argument.

An exception type is a way for stating that perceiving a specific event in the current state of the protocol represents an exceptional situation and must be handled as such, calling an ad-hoc piece of code.

Example:

```
(exception(move(A, key_room, treasure_room),
     illegal_move_exc(
     entering_treasure_room_without_permission(A)))).
```

The events whose type is `move(A, key_room, treasure_room)` may raise an exception (if they take place in the state of the protocol where the exception type may be reached) whose type is `illegal_move_exc`, and that must be handled by the code pointed to by `entering_treasure_room_without_permission`, with argument `A`.

Set timeout type. From a logical point of view, a set timeout type $\mathcal{ST}$ is a couple <type of the event that will cause the alarm to be set, list of triples (timeout label l , delay timeout d and its omission handler, crash timeout c and its crash handler)>.

A timeout is set on occurrence of one event whose type is specified as first element of the couple and causes two alarms to be set and, later on, raised in an automatic way by the system. The first alarm will be automatically raised by the system after the delay timeout, unless an event tagged with label l took place before, and will be handled using the associated "omission handler". The second alarm will be automatically raised by the system after the crash timeout, unless an event tagged with label l took place before. The crash timeout must be greater than the delay timeout and will be handled using the associated "crash handler".

The link between the two alarms set and the condition under which they should have no consequences on the system behavior is given by the label l. In fact, if – after having set the alarms – an event is perceived and it matches a check timeout (explained in the sequel) identified by the same label l, then it means that the condition associated with the alarms has been satisfied and they can be switched off.

From an implementation viewpoint, a `set_timeout` event should save the information in the associated list for successive retrieval using the label as a key. In our SWI-Prolog prototype described in Section III-B, this is done by calling the `set_alarm` predicate which, for each element in the list, asserts it, computes the time when the two alarms should be raised based on the current time and the delay and crash timeouts, and generates the two awake events which will actually take place in the system at due time.

Example:

```
set_timeout( (first_move(A, _SomeRoom, key_room),0),
[timeout_setting(t1(A),
d(1000, handlerOmission(A)),
c(2000, handlerCrash(A)) ) ] ).
```

When one event whose type is `first_move(A, _SomeRoom, key_room)` takes place[2], two timeouts labeled with `t1(A)` together with their handlers are set:

- one whose expiration may be due to omission failures `d(1000, handlerOmission(A))`, meaning that if some check timeout event labeled with `t1(A)` will not take place within `1000` time units from the current time, then the agent or component that should make the condition associated with label `t1(A)` become true will be considered delayed;

[2]The fact that the event type is represented as a couple `(first_move(A, _SomeRoom, key_room),0)` with a `0` as second element means that the type is a "producer event type". This notion is introduced in the sequel.

- and one whose expiration may be due to crashes `c(2000, handlerCrash(A))`, meaning that if the check timeout events labeled with `t1(A)` will not take place within `2000` time units from the current time, then the agent or component that should make the condition associated with label `t1(A)` become true will be considered crashed.

When the timeout will expire, the corresponding event of type "awake" (explained later) will be generated by the sentinel, and the handler will be used to manage the failure.

In this specific example, since the agent identity is part of the label itself, the agent whose delay or crash is under monitoring is `A`. As soon as `A` makes its `first_move` into the `key_room`, two countdowns start. If `A` does not ask the `key` to the `key_keeper` (see example below) within 1000 time units, he will be considered delayed. If he does not ask the `key` in 2000 time units, he will be considered crashed.

Check timeout type. A check timeout type $\mathcal{CT}$ is, from an abstract point of view, a quadruple <type of the event that should take place within the timeout, label of the timeout, pointer to the timeout exception handler, arguments of the timeout exception handler>. As for the exception type, we use a more compact representation as a term with its functor and arguments.

Example:

```
check_timeout((ask(A, key_keeper, key),0),timeout_exc(t1(A),
handlerEventWithDelay(ask(A, key_keeper, key))).
```

When one event of type `ask(A, key_keeper, key)` takes place, a check if it is on time or not is performed, by retrieving the information associated with the label `t1(A)`. In our implementation, this means looking for a fact previously asserted by `set_alarm` whose label unifies with `t1(A)`. If the event is on time, the protocol is respected and alarms associated with label `t1(A)` are switched off (namely, it is tagged as "done" and the awake events that have been already generated by set_alarm will have no effect). If the event is not on time, then one or both alarms associated with `t1(A)` have expired and the corresponding handlers have been executed. Note that the crash handler is supposed to be executed when there is a high degree of confidence that the expected event will not take place anymore. Since handlers could be generic ones, we leave the developer the possibility to specify some further code, `handlerEventWithDelay(ask(A, key_keeper, key))` in this case, to be executed when the delayed event takes place.

Awake type. An awake event type $\mathcal{AW}$ is a couple <awake type (delay or crash), label of the timeout>.

It is automatically generated when the countdown associated with a delay or crash alarm reaches 0.

Example:

```
awake_delay(t1(A)) (resp. awake_crash(t1(A))).
```

When one event typed with these awake types takes place, then the delay (resp. crash) handler defined when the timeout for `t1(A)` was set will be called in order to cope with the omission (resp. crash) failure.

Uninteresting events. Events which are not interesting for the monitoring purposes can be tagged as uninteresting ones just by exploiting the event type. For example, the following definition of the `has_type` predicate states that all the actual events observed in the system which do not unify either with `truck_at_dock(_, _, _)`, or with `drop_parcel(_, _)`, etc, have type `uninteresting_event`.

```
has_type(Event, uninteresting_event) :-
   /* must list explicitly all the interesting events, as
        different (\=) from the current one */
   Event \= truck_at_dock(_, _, _),
   Event \= drop_parcel(_, _),
   Event \= move_to_free_shelf(_, _, _, _),
   Event \= collect_parcel(_, _),
   Event \= move_to_truck(_, _, _, _).
```

Any multiagent protocol can be defined as the interesting part of the protocol, interleaved with the uninteresting one. Interleaving is modeled by the fork operator "`|`" (see more details below) and the branch of the protocol modeling the uninteresting events can be defined in a standard way as

```
DISCARD = (uninteresting_event, 0): DISCARD
```

meaning that a `DISCARD` constrained global type is defined as an infinite sequence (sequence operator "`:`") of uninteresting events.

Producers and consumers. In order to model constraints across different branches of a constrained fork (explained later in this section), we introduce two different kinds of event types, called *producers* and *consumers*, respectively. Each occurrence of a producer event type must correspond to the occurrence of a new event; in contrast, consumer event types correspond to the same event specified by a certain producer event type. The purpose of consumer event types is to impose constraints on event sequences, *without introducing new events*. A consumer is an event type η, whereas a producer is an event type η equipped with a natural superscript n. In the Prolog concrete syntax that we use, η^n is represented by the couple (η, n).

Constrained global types. A constrained global type τ represents a set of possibly infinite sequences of events, and is defined on top of the following type constructors:

- λ (empty sequence): the singleton set $\{\epsilon\}$ containing the empty sequence ϵ.
- $\eta^n{:}\tau$ (*seq-prod*): the set of all sequences whose first element is an event e matching type η ($e \in \eta$), and the remaining part is a sequence in the set represented by τ. The superscript n specifies the number n of corresponding consumers that coincide with the same event type η; hence, n is the required number of times $e \in \eta$ has to be "consumed" to allow a transition labeled by e.
- $\eta{:}\tau$ (*seq-cons*): a consumer of event e matching type η ($e \in \eta$), and followed by any sequence in the set represented by τ.

- $\tau_1 + \tau_2$ (*choice*): the union of the sequences of τ_1 and τ_2.
- $\tau_1|\tau_2$ (*fork*): the set obtained by shuffling the sequences in τ_1 with those in τ_2.
- $\tau_1 \cdot \tau_2$ (*concat*): the concatenation of the sequences of τ_1 and τ_2.
- The "meta-construct" *fc* (for *finite composition*) takes τ, the constructor cn, and the positive natural number n as inputs and generates the "normal" constrained global type (τ cn τ cn ... cn τ) (n times, hence $fc(\tau, cn, 1)$=τ).

Constrained global types are regular terms, that is, can be cyclic (recursive), and hence they can be represented by a finite set of syntactic equations. To make the treatment simpler, we limit our investigation to *contractive* (a.k.a. *guarded*) and *deterministic* constrained global types. A constrained global type τ is *contractive* if all infinite paths[3] in τ contain an occurrence of the ":" constructor. Determinism ensures that dynamic checking can be performed efficiently without backtracking. Intuitively, a constrained global type is deterministic if, in case more transition rules can be applied when event e takes place, they lead to equivalent constrained global types. The formal definition is given in the next section.

B. Semantics

The state of a constrained global type τ can be represented by τ itself. In this section, when talking about constrained global types we will refer to their current state. Also, we will use "constrained global type" and "protocol" interchangeably.

The interpretation of a constrained global type is based on the notion of transition, a total function δ:$\mathbb{N} \times \mathcal{CT} \times \mathcal{A} \rightarrow \mathcal{P}_{fin}(\mathcal{CT} \times \mathbb{N})$, where $\mathcal{CT}$ and $\mathcal{A}$ denote the set of contractive and constrained global type and of events, respectively. The `next/5` Prolog predicate discussed below implements the δ transition function that we do not show in this paper for space constraints. If τ_1 represents the current state of the protocol and the event e takes place, then the protocol can move to τ_2 iff $\delta(0, \tau_1, e) = (\tau_2, 0)$, that we write as $\tau_1 \overset{e}{\rightarrowtail} \tau_2$.

Moving from the definition of δ to its Prolog implementation, the following relationship holds:

$\delta(N, T1, Ev) = (T2, M)$ iff `next(N,T1,Ev,T2,M).`

The auxiliary function ϵ(_), implemented by the `empty/1` Prolog predicate, specifies the constrained global types whose interpretation contains the empty sequence ϵ. Intuitively, a constrained global type τ s.t. $\epsilon(\tau)$ holds specifies a protocol that is allowed to successfully terminate.

Let τ_0 be a contractive and constrained global type. A *run* ρ for τ_0 is a sequence $\tau_0 \overset{e_0}{\rightarrowtail} \tau_1 \overset{e_1}{\rightarrowtail} \ldots \overset{e_{n-1}}{\rightarrowtail} \tau_n \overset{e_n}{\rightarrowtail} \tau_{n+1} \overset{e_{n+1}}{\rightarrowtail} \ldots$ such that (1) either the sequence is infinite, or it has finite length $k \geq 0$ and the last constrained global type τ_k verifies $\epsilon(\tau_k)$; and (2) for all τ_i, e_i, and τ_{i+1} in the sequence, $\tau_i \overset{e_i}{\rightarrowtail} \tau_{i+1}$ holds. We denote by $A(\rho)$ the possibly empty or infinite sequence of events $e_0 e_1 \ldots e_n \ldots$ contained in ρ. The interpretation $[\![\tau_0]\!]$ of τ_0 is the set $\{A(\rho) \mid \ \rho$ is a run for $\tau_0\ \}$. A contractive constrained global type τ is deterministic if for any possible run ρ of τ and any possible τ' in ρ, if $\tau' \overset{e}{\rightarrow} \tau''$, and $\tau' \overset{e}{\rightarrow} \tau'''$, then $[\![\tau'']\!] = [\![\tau''']\!]$.

[3]By "path of a constrained global type" we mean "path in the possibly infinite tree corresponding to the term that represents the constrained global type".

The next and empty predicates.

The `next` predicate implements the δ function defining the semantics of constrained global types extended with constructs to deal with exceptions, timeouts and their handlers. Besides the definition of transition rules for "normal" constrained global types, in the sequel we describe the additional transition rules for the new constructs introduced in this paper.

Basic cases for "normal" constrained global types.

If the constrained global type is a sequence (":" operator) consisting of a producer event type EvType with associated number N, followed by the constrained global type T (second argument), and the actual event perceived in the environment is Ev (third argument), and Ev has type EvType, then (EvType,N):T can move to T (fourth argument). If there were no events of type EvType to be consumed (first argument of next is zero), now N events of EvType are generated and must be consumed (fifth argument of next is N, which is the number associated with EvType in the second argument)

```
next(0,(EvType,N):T,Ev,T,N) :- has_type(Ev, EvType).
```

This rule is similar to the previous one, but the event type EvType is a consumer (in fact, it has no number associated) and, if the actual event Ev perceived from the environment has type EvType, then EvType:T1 moves to T1 and the events of type EvType still to be consumed are decreased by one.

```
next(M,EvType:T1,Ev,T1,N) :-
EvType \= (_,_), has_type(Ev, EvType),
M > 0, N is M - 1.
```

If T1 can move to T2, then the choice between T1 and any other constrained global type, T1+_, can move to T2 (and the converse for the second rule)

```
next(M,T1+_,Ev,T2,N) :- next(M,T1,Ev,T2,N).
next(M,_+T1,Ev,T2,N) :- !, next(M,T1,Ev,T2,N).
```

If T1 can move to T3, then the interleaving between T1 and T2, T1|T2, can move to T3|T2 (and the converse for the second rule)

```
next(N,T1|T2,Ev,T3|T2,M) :- next(N,T1,Ev,T3,M).
next(N,T1|T2,Ev,T1|T3,M) :- next(N,T2,Ev,T3,M).
```

If T1 can move to T3 and T2 can move to T4, and T1 produces event types which can be consumed by T2, then T1 and T2 can synchronize and T1|T2 can move to T3|T4 (and the converse for the second rule)

```
next(N,T1|T2,Ev,T3|T4,P) :-
   next(N,T1,Ev,T3,M), M > 0, next(M,T2,Ev,T4,P).
next(N,T1|T2,Ev,T4|T3,P) :-
   !,next(N,T2,Ev,T3,M), M > 0, next(M,T1,Ev,T4,P).
```

If T1 can move to T3, then the concatenation T1*T2 can move to the concatenation T3*T2; if T1 contains ϵ, that is, empty(T1) holds (second rule), then T1*T2 can move to T3 if T2 can move to T3

```
next(M,T1*T2,Ev,T3*T2,N) :-
   next(M,T1,Ev,T3,N).
next(M,T1*T2,Ev,T3,N) :-
   !,empty(T1), next(M,T2,Ev,T3,N).
```

Rule for finite composition of constrained global types.

```
fc(T, T1, C, N) :- N>1, copy_term(T1, Fresh1),
N1 is N-1, fc(T2, T1, C, N1),  T =.. [C, Fresh1, T2].
fc(Fresh1, T1, _C, 1) :- copy_term(T1, Fresh1).
```

Additional rule for dealing with exceptions.

If the constrained global type is a sequence consisting of an event of type EvType, which raises an exception to be handled with the Handler code, followed by T, and the actual event perceived in the environment is Ev, and Ev has type EvType, then the Handler is executed and exception(EvType,Code):T moves to T.

```
next(0, exception(EvType,Handler):T, Ev,T,0) :-
   has_type(Ev, EvType), call(Handler).
```

Additional rules for dealing with timeouts.

If the constrained global type is a sequence consisting of a set_timeout event with its parameters, followed by T, and the perceived event Ev has type EvType, then the alarm is set by calling the set_alarm code. We do not show the rule dealing with the case where EvType is a consumer event type rather than a producer.

```
next(0,set_timeout((EvType,N), List):T, Ev,T,N) :-
   has_type(Ev,EvType), set_alarm(List).
```

If the constrained global type is a sequence consisting of a check_timeout event with its parameters, including the label of the timeout handler to check and the Code to execute if the timeout is expired, followed by T, and the perceived event Ev has type EvType, then the actual time is retrieved together with the timeout settings associated with label L. If the timeout is expired, then Handler is executed, otherwise the two alarms are switched off by setting their handlers to "done". The constrained global time moves to T. We do not show the rule dealing with the case where EvType is a consumer event type rather than a producer.

```
next(0,check_timeout((EvType,N), timeout_exc(L, Handler)):T,
Ev,T,N) :-
   has_type(Ev, EvType), get_time(CurrentTime),
   timeout_setting(L, d(Delay, _HandlDTimeExp), _C),
   (CurrentTime > Delay ->
     call(Handler);
     (retract(timeout_setting(L, d(Delay, _), c(Crash, _))),
      assert(timeout_setting(L, d(Delay, done), c(Crash,
           done))))
    ).
```

If the constrained global type is a sequence consisting of an awake_delay event with a label L as argument, followed by T, and the perceived event is awake_delay(L) which is automatically generated when a timeout expires, then the code associated with that awake event is executed and this fact is recorded by setting the handler to "done" for avoiding successive re-executions. The constrained global time moves to T.

```
next(N,awake_delay(L):T,awake_delay(L),T,N) :-
   timeout_setting(L, d(Delay, HandlDTimeExp), C),
   HandlDTimeExp \== done,
   call(HandlDTimeExp),
   retract(timeout_setting(L, d(Delay, HandlDTimeExp), C)),
   assert(timeout_setting(L, d(Delay, done), C)), !.

next(N,awake_crash(L):T,awake_crash(L),T,N) :-
   timeout_setting(L, D, c(Crash, HandCTimeExp)),
   HandCTimeExp \== done,
   call(HandCTimeExp),
   retract(timeout_setting(L, D, c(Crash, HandCTimeExp))),
   assert(timeout_setting(L, D, c(Crash, done))), !.
```

Code for setting the alarms.

```
set_alarm([]).

set_alarm(
   [timeout_setting(L, d(D, HandlDTimeExp),
                       c(C, HandCTimeExp))|T]) :-
   asserta(timeout_setting(L, d(D, HandlDTimeExp),
                              c(C, HandCTimeExp))),
   get_time(CurrentTime), !,
   DT is CurrentTime + D,
   CT is CurrentTime + C,
   generate_event(DT, awake_delay(L)),
   generate_event(CT, awake_crash(L)),
   set_alarm(T).
```

The `empty` predicate is true on the empty constrained global type `lambda`. If $T1$ or $T2$ contains ϵ, then $T1 + T2$ contains ϵ. If both $T1$ and $T2$ contain ϵ, then $T1|T2$ and $T1 * T2$ contains ϵ.

```
empty(lambda) :- !.
empty(T1+T2) :- (empty(T1),!;empty(T2)).
empty(T1|T2) :- !,empty(T1),empty(T2).
empty(T1*T2) :- !,empty(T1),empty(T2).
```

IV. EXAMPLES

In all the examples of this paper, event types have the same syntax of the events they refer to, apart from the "uninteresting_event" type which must be explicitly defined as all the events which do not unify with an interesting event. This means that, apart from uninteresting ones, event e has event type e.

A. Example 1: illegal moves

This protocol states that, in order to enter the treasure room, an agent first moves to the key room, asks for the key of the treasure room to the key keeper within 1000 secs, waits for an acceptance, enters the treasure room and then exits it. If the key keeper refuses to give the key, the agent must exit the key room. One exception to this normal flow is that from an external room (we do not care which one) the agent enters the treasure room with some trick without asking the permission to the key keeper and without waiting for the key. In this case, an exception is raised and the exception handler associated with this "illegal move" exception must be executed. Among the actions implemented by the handler, some will be executed (and consequently monitored too) by the sentinel. In particular, the sentinel will sanction the agent, which will be transported out of the treasure room, and in jail. After some time, the agent will be set free again. Note that here, the sentinel monitors one of its actions, `sanction(sentinel, A)`, implementing a form of introspection.

The last line of the protocol models the case when the event that has been sensed is unknown. If the `unknown_event(A)` event type holds on all the events that the sentinel is not able

to manage, then this exception will be thrown whenever the sentinel is facing an unexpected exception. The handler can be some general code trying to ensure the vital functions of the system even if the normal flow has some fail.

```
T = ((set_timeout((move(A, _SomeRoom, key_room),0),
     [timeout_setting(
        t1(A),
        d(1000, handlerDTimeoutExpiration(A)),
        c(2000, handlerCTimeoutExpiration(A)))]):
  ((check_timeout((ask(A, key_keeper, key),0),
     timeout_exc(
     t1(A),
     handlerEventWithDelay(ask(A, key_keeper, key)))):
     (
       ((accept(key_keeper, A, key),0):(give(key_keeper, A,
           key),0):
       (move(A, key_room, treasure_room),0):(move(A,
           treasure_room, key_room),0):
       (give(A, key_keeper:key),0):lambda)
     +
       ((refuse(key_keeper, A, key),0):
       (move(A, key_room, _AnotherRoom),0):lambda)
    )*T)
+
   ((exception(move(A, _SomeRoom, treasure_room),
     illegal_move_exc(
          entering_treasure_room_without_permission(A)))):
    (sanction(sentinel, A),0):(transport(A, treasure_room,
         key_room),0):
    (transport(A, key_room, jail),0):(transport(A, jail,
         _AnyRoom),0):lambda)
  )
 )
+
 ((exception(unknown_event(A), unknown_event_exc(A))):lambda
      )).
```

Now we show some events traces compliant with the protocol, generated using SWI Prolog implementing the `next` predicate defined in Section III and the `accept` predicate that accepts (or generates, if the free variables can be grounded when `has_type` is called) sequences of events with a given length `N`, which respect the constrained global type `T`. By using the `findall` builtin predicate on `accept`, we were able to generate all the accepted traces of a given length.

```
accept(N,T,[*]) :- empty(T), N==0,!.
accept(0,_,[]) :- !.
accept(N,T1,[Ev|L]) :-
   next(0,T1,Ev,T2,0), M is N-1, accept(M,T2,L).
```

Correct traces with length 6 include:

```
move(alice, room1, key_room),
ask(alice, key_keeper, key),
accept(key_keeper, alice, key),
give(key_keeper, alice, key),
move(alice, key_room, treasure_room),
move(alice, treasure_room, key_room)

move(alice, room1, key_room),
ask(alice, key_keeper, key),
refuse(key_keeper, alice, key),
move(alice, key_room, room1),
move(alice, room1, key_room),
ask(alice, key_keeper, key)

move(alice, room1, key_room),
ask(alice, key_keeper, key),
refuse(key_keeper, alice, key),
move(alice, key_room, room2),
move(alice, room2, key_room),
ask(alice, key_keeper, key)
```

Some traces with length 6 where an exception is found and the sentinel sanctions alice are:

```
move(alice, room1, key_room),
move(alice, key_room, treasure_room),
sanction(sentinel, alice),
transport(alice, treasure_room, key_room),
transport(alice, key_room, jail),
transport(alice, jail, room1)

move(alice, room1, treasure_room),
sanction(sentinel, alice),
transport(alice, treasure_room, key_room),
transport(alice, key_room, jail),
transport(alice, jail, room1),
move(alice, room1, key_room)
```

B. Example 2: dock loading and unloading

This example is a simplified version of the scenario described in [35]:

> *In the loader dock several workers wander around or carry parcels between incoming trucks and the shelves. Their job is to unload parcels from a full truck, or to deliver parcels to an empty one, whenever trucks arrive at the store. The dock itself contains shelves where parcels can be placed temporally. The shelves are separated by corridor ways, which are used by workers to transport the parcels. ... When a worker makes an intention to move somewhere, it communicates this intention to other ones for the purpose of coordination. A trajectory of the movement, described by points and exact time-values, is sent to each worker, so it can update its beliefs about future changes in the environment. This information is stored in a domain time model representing the beliefs concerning prospective workers' positions. This information guides the planning process of each worker, so none of them intersects a path of another.*

In this example we assume that workers and trucks are represented as agents, which are free to execute many actions including those related to the work in the dock loader. The next code manages a simplification of the described example, without monitoring for example that agents will not crash while moving: its aim is to give a flavor of how we can use the uninteresting events to specify only the subset of the actions that are relevant for the controller.

The protocol starts when a `Truck` arrives at the dock (`UNLOAD_TRUCK`), carrying `NumberOfParcels` parcels, which must be unloaded (so, after this event, the protocol moves on with `UNLOAD_ALL_PARCELS`, that is created repeating the trace `UNLOAD_PARCEL` for each parcel in a parallel way using the | constructor). The protocol part identified by `UNLOAD_PARCEL` states that the correct sequence is that firstly an agent comes to the `Truck` position, then it collects a parcel, moves to a free shelf and leaves there the parcel.

We model the positions inside the dock as couples of points, and the time slots as couples of minutes.

To model the fact that we are only interested in monitoring the events specified in the previous traces, but not the others (that is, if one event that is not related to this protocol takes place during the protocol, it is not a violation of it!), we add in the trace `UNLOAD_TRUCK` a fork (| operator) with `DISCARD`, so that any event of type `uninteresting_event`, at any time, is caught by the trace `DISCARD` (that simply

skips that event and waits for the next one). The type `uninteresting_event` is defined differently from all the other events: an `uninteresting_event` is whatever event that is not `move_to_truck`, `collect_up_parcel`, and so on.

```
UNLOAD_PARCEL =
(move_to_truck(Agent, _FromPosition, TruckPosition,
     _TimeSlot),0):
(collect_parcel(Agent, P),0):
(move_to_free_shelf(Agent, TruckPosition, _FreeShelfPosition
     , _SuccTimeSlot),0):
(drop_parcel(Agent, P),0):
lambda,
fc(UNLOAD_ALL_PARCELS, UNLOAD_PARCEL, |, NumberOfParcels),
UNLOAD_TRUCK =
  (((truck_at_dock(_TruckId,TruckPosition,NumberOfParcels)
       ,0):
     UNLOAD_ALL_PARCELS)|
  DISCARD),
DISCARD = (uninteresting_event, 0): DISCARD.
```

An example of a trace (including events not interesting for the protocol) which is recognized as compliant with the protocol is:

```
snoring(a1),
truck_at_dock(truck1, (0, 0), NumberOfParcels),
move_to_truck(a1, (6, 21), (0, 0), (10, 15)),
collect_parcel(a1, p1),
move_to_truck(a2, (12, 3), (0, 0), (15, 18)),
snoring(a1),
smiling(a1),
move_to_free_shelf(a1, (0, 0), (54, 3), (15, 22)),
drop_parcel(a1, p1),
move_to_truck(a1, (54, 3), (0, 0), (10, 15)),
collect_parcel(a2, p2),
snoring(a2),
dancing_tiptap(a1),
move_to_free_shelf(a2, (0, 0), (43, 78), (18, 21)),
drop_parcel(a2, p2),
move_to_truck(a1, (4, 9), (0, 0), (30, 35)),
collect_parcel(a1, p3),
move_to_truck(a3, (44, 9), (0, 0), (31, 37)),
collect_parcel(a3, p4),
move_to_free_shelf(a3, (0, 0), (6, 24), (37, 49)),
drop_parcel(a3, p4),
move_to_free_shelf(a1, (0, 0), (1, 24), (35, 43)),
drop_parcel(a1, p3),
snoring(truck),
smiling(truck),
drinking_a_beer(truck)
```

whereas the following trace is not compliant because parcel `p1`, collected up by agent `a1` (third event), is dropped by agent `a4` (fourth event) instead of being dropped by `a1`.

```
truck_at_dock(truck1, (0, 0), NumberOfParcels),
move_to_truck(a1, (6, 21), (0, 0), (10, 15)),
collect_parcel(a1, p1),
drop_parcel(a4, p1).
```

C. Example 3: air conditioning

The last example relates to the AmI context, where we assume to manage a building for social events (concerts, workshops and so on): in this building there are many rooms, and we would add some intelligence to the system itself, specifying some rules that should be respected in the building. In particular, if there are more then 1000 visitors and the temperature inside the building is >30, then if the air conditioning is closed, it must be switched on, otherwise, no more visitors can enter the building.

This situation is (partially) described by the constrained global type AC:

```
SAFETY_CHECK =
(
  (
     ((counted_more_than_1000_visitors,0):
      lambda)
     |
     ((temperature_higher_than_30_celsius,0):
      lambda)
  )
  *
  (
     ((air_conditioning_off,0):
      (switch_air_conditioning_on,0):
      lambda
     )
     +
     ((air_conditioning_on,0):
      (do_not_admit_other_visitors,0):
      lambda
     )
   )
)
*
SAFETY_CHECK,
AC = (SAFETY_CHECK | DISCARD),
DISCARD = (uninteresting_event, 0): DISCARD.
```

Some examples of traces which are recognized as compliant with the protocol are:

```
counted_more_than_1000_visitors,
temperature_higher_than_30_celsius,
air_conditioning_on,
do_not_admit_other_visitors.

counted_more_than_1000_visitors,
temperature_higher_than_30_celsius,
air_conditioning_off,
switch_air_conditioning_on

temperature_higher_than_30_celsius,
concert_begins,
counted_more_than_1000_visitors,
air_conditioning_off,
switch_air_conditioning_on
```

In this case the constrained global type is used to control that if a specific situation happens (visitors $>$ 1000 and temperature $>$ 30, with events perceived in any order), some specific action (`do_not_admit_other_visitors` or `switch_air_conditioning_on`) is taken.

V. Conclusions

In this paper we described an approach for designing, formalizing and implementing *sentinels* that detect errors in *fail-uncontrolled multiagent systems*, and *controllers* that identify particular situations in *AmI systems*. Our approach – like all the approaches aimed at runtime verification of system properties – requires that a formal representation of the system property (a MAS protocol in our case) is available. We used *constrained global types extended with constructs for modeling exceptions, timeouts, and handlers* as representation formalism. We implemented the SWI Prolog interpreter for this formalism and we run different tests in different application domains.

Representing a protocol using our formalism may not be trivial, but the more expressive is the formalism, the more difficult is using it. When used for runtime verification, our formalism is more expressive than linear time temporal logic (LTL [36]) because it can represent a protocol like $a^n b^n$ which

cannot be expressed using LTL[4]. For this reason it should not be surprising that our formalism is more complex than, for example, LTL.

There are however many benefits in using such a formalism:

- having a protocol formalization can help testing the MAS itself, since our Prolog translation can be exploited to generate and test correct, an not correct, simulations of the system, and constrained global types have already been proven a powerful language, able to model complex protocols (as shown in [12]);
- in *AmI systems*, *controllers* can be added on top of the already existing agents, as an independent layer, improving the modularity of the overall system and simplifying the design, development and maintenance phases while, in *fail-uncontrolled MASs*, *sentinels* may help to identify and recover from error situations by separating the errors management from the normal protocol flow;
- the basic version of our approach, without the extensions discussed in this paper, works for JADE and Jason, two among the most used platforms for MAS development; the transition rules providing the semantics of the improved formalism presented in this paper have been already translated into Prolog, the language we used for implementing our approach on top of JADE and Jason, so we claim that extending the JADE and Jason prototypes with exceptions, timeouts, and handlers should raise no technical problems;
- the extensions presented in this paper enhance the "MAS control layer" with the possibility to manage both MASs where a predefined global protocol does not exist (*AmI systems*) and MASs where errors may arise and must be formalized and managed in a controlled way.

The formalism that we have discussed in this paper seems suitable for being integrated into a sensor network in order to capture what happens in the environment, as we did in [12]. Thanks to its modularity it could be used to monitor and supervise environments where physical agents (robots) can fail leading to exceptional situations and need to be either reconfigured or replaced at runtime.

The feasibility of the approach is limited to situations where a formal representation of entities, messages and in general events, exists: so, when considering dynamic and open large-scale MASs as those in AmI and wireless sensor networks, we assume that all the agents that can join the MAS (maybe developed by third parties), are foreseen in the formalization. Our work is meant to provide a set of tools and languages for monitoring distributed systems, without claiming to be able to solve all the related problems.

For example, although we can automatically project the protocol onto subsets of agents (it means that the protocol itself can be split into separated parts) and then allocate to different controllers the supervision of those parts of the MAS, we have not solved the problem of *how to* identify the best protocol distribution, which is not even always possible. In a similar way, the decision of what to do in case of a failure is demanded to the protocol formalization or to the actual agent code, and is out of the scope of our work.

As part of our future work, we are investigating how expressing more complex behaviors when specifying timeout, and how managing timeouts related to the same agent. From a more theoretical point of view, the relationships between our formalism and timed linear time temporal logic [37] should be deeply analyzed.

[4]The protocol where one agent sends n messages of type a to another agent (with n that can be any positive number) and the receiver answers with n messages of type b.

ACKNOWLEDGMENT

This work has been partially supported by the MIUR PRIN Project CINA: Compositionality, Interaction, Negotiation, Autonomicity for the future ICT society, prot. 2010LHT4KM.

REFERENCES

[1] Y. Demazeau, "From interactions to collective behaviour in agent-based systems," in *In Proceedings of the 1st European Conference on Cognitive Science. Saint-Malo*, 1995, pp. 117–132.

[2] M. Barbuceanu and M. S. Fox, "COOL: A language for describing coordination in multi agent systems," in *Proceedings of the First International Conference on Multiagent Systems, June 12-14, 1995, San Francisco, California, USA*, V. R. Lesser and L. Gasser, Eds. The MIT Press, 1995, pp. 17–24.

[3] Y. Abushark, J. Thangarajah, T. Miller, and J. Harland, "Checking consistency of agent designs against interaction protocols for early-phase defect location," in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 933–940.

[4] M. Baldoni, C. Baroglio, and F. Capuzzimati, "Typing multi-agent systems via commitments," in *Engineering Multi-Agent Systems*. Springer, 2014, pp. 388–405.

[5] F. Al-Saqqar, J. Bentahar, K. Sultan, and M. El Menshawy, "On the interaction between knowledge and social commitments in multi-agent systems," *Applied intelligence*, vol. 41, no. 1, pp. 235–259, 2014.

[6] D. Briola, V. Mascardi, and D. Ancona, "Distributed runtime verification of JADE multiagent systems," in *Intelligent Distributed Computing VIII - Proceedings of the 8th International Symposium on Intelligent Distributed Computing, IDC 2014, Madrid, Spain, September 3-5, 2014*, ser. Studies in Computational Intelligence, D. Camacho, L. Braubach, S. Venticinque, and C. Badica, Eds., vol. 570. Springer, 2014, pp. 81–91. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10422-5_10

[7] D. Ancona, D. Briola, A. Ferrando, and V. Mascardi, "Global protocols as first class entities for self-adaptive agents," in *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), May 4–8, 2015, Istanbul, Turkey*, R. Bordini, E. Elkind, G. Weiss, and P. Yolum, Eds. IFAAMAS/ACM, 2015.

[8] P. Yolum and M. P. Singh, "Reasoning about commitments in the event calculus: An approach for specifying and executing protocols," *Ann. Math. Artif. Intell.*, vol. 42, no. 1-3, pp. 227–253, 2004.

[9] D. Ancona, S. Drossopoulou, and V. Mascardi, "Automatic Generation of Self-Monitoring MASs from Multiparty Global Session Types in Jason," in *DALT X. Revised, Selected and Invited Papers*, ser. LNAI, vol. 7784. Springer, 2012.

[10] D. Ancona, M. Barbieri, and V. Mascardi, "Constrained global types for dynamic checking of protocol conformance in multi-agent systems," 2013, sAC 2013. ACM.

[11] V. Mascardi and D. Ancona, "Attribute global types for dynamic checking of protocols in logic-based multiagent systems (technical communication)," 2013, to appear in Theory and Practice of Logic Programming, On-line Supplement, as technical communication of the ICLP 2013 conference.

[12] V. Mascardi, D. Briola, and D. Ancona, "On the expressiveness of attribute global types: The formalization of a real multiagent system protocol," in *AI*IA 2013: Advances in Artificial Intelligence - XIIIth International Conference of the Italian Association for Artificial Intelligence, Turin, Italy, December 4-6, 2013. Proceedings*, ser. Lecture Notes in Computer Science, M. Baldoni, C. Baroglio, G. Boella, and R. Micalizio, Eds., vol. 8249. Springer, 2013, pp. 300–311. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-03524-6_26

[13] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak Using Jason*. John Wiley & Sons, 2007.

[14] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. Wiley, 2007.

[15] D. Ancona, D. Briola, A. El Fallah Seghrouchni, V. Mascardi, and P. Taillibert, "Efficient verification of MASs with projections," in *Engineering Multi-Agent Systems - Second International Workshop, EMAS 2014, Revised Selected Papers*, ser. Lecture Notes in Computer Science, F. Dalpiaz and J. D. M. B. van Riemsdijk, Eds., vol. 8758, 2014, pp. 246–270.

[16] N. Faci, Z. Guessoum, and O. Marin, "DimaX: A fault-tolerant multi-agent platform," in *EUMAS*, ser. CEUR Workshop Proceedings, B. Dunin-Keplicz, A. Omicini, and J. A. Padget, Eds., vol. 223. CEUR-WS.org, 2006.

[17] D. Powell, Ed., *Delta-4: A Generic Architecture for Dependable Distributed Computing (Research Reports Espirit Project 818/2252, Delta 4, Vol. 1)*, 1992.

[18] S. Hägg, "A sentinel approach to fault handling in multi-agent systems," *Second Australian Workshop on Distributed Artificial Intelligence, Cairns, QLD, Australia, August 27, 1996*, vol. s. 181-95, 1997.

[19] J. Eder and W. Liebhart, "The workflow activity model WAMO," in *Proc. 3rd Int. Conf. Cooperative Inf. Syst.*, 1995, pp. 87–98.

[20] A. U. Mallya and M. P. Singh, "Modeling exceptions via commitment protocols," in *AAMAS*, F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. P. Singh, and M. Wooldridge, Eds. ACM, 2005, pp. 122–129.

[21] Z. Guessoum, M. Ziane, and N. Faci, "Monitoring and organizational-level adaptation of multi-agent systems," in *AAMAS*. IEEE Computer Society, 2004, pp. 514–521.

[22] M. H. Coen, "Design principles for intelligent environments," in *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98, July 26-30, 1998, Madison, Wisconsin, USA.*, J. Mostow and C. Rich, Eds. AAAI Press / The MIT Press, 1998, pp. 547–554. [Online]. Available: http://www.aaai.org/Library/AAAI/1998/aaai98-077.php

[23] M. C. Mozer, "The neural network house: An environment that adapts to its inhabitants," in *Proc. Am. Assoc. Artificial Intelligence Spring Symp. Intelligent Environments*. AAAI Press, 1998, pp. 110–114.

[24] P. Davidsson and M. Boman, "Saving energy and providing value added services in intelligent buildings: A MAS approach," in *Agent Systems, Mobile Agents, and Applications, Second International Symposium on Agent Systems and Applications and Fourth International Symposium on Mobile Agents, ASA/MA 2000, Zürich, Switzerland, September 13-15, 2000, Proceedings*, ser. Lecture Notes in Computer Science, D. Kotz and F. Mattern, Eds., vol. 1882. Springer, 2000, pp. 166–177. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45347-5_14

[25] H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish, and H. Duman, "Creating an ambient-intelligence environment using embedded agents," *IEEE Intelligent Systems*, vol. 19, no. 6, pp. 12–20, 2004. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MIS.2004.61

[26] F. Doctor, H. Hagras, and V. Callaghan, "A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 35, no. 1, pp. 55–65, 2005. [Online]. Available: http://dx.doi.org/10.1109/TSMCA.2004.838488

[27] C. Ramos, J. C. Augusto, and D. Shapiro, "Ambient intelligence - the next step for artificial intelligence," *IEEE Intelligent Systems*, vol. 23, no. 2, pp. 15–18, 2008. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MIS.2008.19

[28] J. M. Gascueña, E. Navarro, P. Fernández-Sotos, A. Fernández-Caballero, and J. Pavón, "Idk and icaro to develop multi-agent systems in support of ambient intelligence," *Journal of Intelligent and Fuzzy Systems*, 2014.

[29] M. Villaverde, D. Perez, and F. Moreno, "Cooperative learning model based on multi-agent architecture for embedded intelligent systems," in *Industrial Electronics Society, IECON 2014-40th Annual Conference of the IEEE*. IEEE, 2014, pp. 2724–2730.

[30] A. Olaru, "Context-awareness in multi-agent systems for ambient intelligence," in *Context in Computing*. Springer, 2014, pp. 541–556.

[31] M. Ruta, F. Scioscia, G. Loseto, and E. Di Sciascio, "Semantic-based resource discovery and orchestration in home and building automation: a multi-agent approach," *Industrial Informatics, IEEE Transactions on*, vol. 10, no. 1, pp. 730–741, 2014.

[32] A. Freitas, D. Schmidt, A. Panisson, R. H. Bordini, F. Meneguzzi, and R. Vieira, "Applying ontologies and agent technologies to generate ambient intelligence applications," in *Agent Technology for Intelligent Mobile Services and Smart Societies*. Springer, 2015, pp. 22–33.

[33] A. Chaouche, A. E. Fallah-Seghrouchni, J. Ilié, and D. Saïdouni, "A higher-order agent model with contextual planning management for ambient systems," *T. Computational Collective Intelligence*, vol. 8780, pp. 146–169, 2014. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-44871-7_6

[34] G. Gutnik, "Monitoring large-scale multi-agent systems using overhearing," in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '05. New York, NY, USA: ACM, 2005, pp. 1377–1377. [Online]. Available: http://doi.acm.org/10.1145/1082473.1082785

[35] A. Walczak, L. Braubach, A. Pokahr, and W. Lamersdorf, "Augmenting bdi agents with deliberative planning techniques," in *Proceedings of the 4th International Conference on Programming Multi-agent Systems*, ser. ProMAS'06. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 113–127. [Online]. Available: http://dl.acm.org/citation.cfm?id=1759324.1759334

[36] A. Pnueli, "The temporal logic of programs," in *FOCS*, IEEE, Ed., 1977, pp. 46–57.

[37] J.-F. Raskin, "Logics, automata and classical theories for deciding real-time," Ph.D. dissertation, Namur, Belgium, 1999.

Coordination of Large-Scale Socio-Technical Systems: Challenges and Research Directions

Andrea Omicini
ALMA MATER STUDIORUM–Università di Bologna
via Sacchi 3, 47521 Cesena, Italy
Email: andrea.omicini@unibo.it

Franco Zambonelli
Università di Modena e Reggio Emilia
via G. Amendola 2, 42100 Reggio Emilia, Italy
Email: franco.zambonelli@unimore.it

***Abstract*—Most of the emerging software-intensive systems nowadays are very large-scale ones, and inherently socio-technical. In this position paper, we argue that the peculiar features of such emerging systems (up to millions of interacting components, lacking central control, mixing humans and artificial components) call for novel approaches to coordinate the overall activities and functionalities. Accordingly, we discuss the key challenges to be faced by research in coordination models and technologies, and try to sketch some promising research directions.**

I. INTRODUCTION

The massive diffusion of networked ICT devices increasingly entangled with our physical and social world, along with progresses in the area of robotics and AI, are making our everyday life and economic activities increasingly dependent on the functionalities of large-scale distributed software-intensive (i.e., heavily relying on software) systems [1]. In the near future, our streets will be populated by self-driving cars interacting with traffic control infrastructures, most physical work activities will be performed by robots, citizens will actively contribute to the automated activities of urban management, all health care activities and infrastructures will be managed by intelligent information infrastructure.

All the above examples, and most emerging ICT-centred scenarios, are characterised by the innovative services and functionalities they provide, which require coordination of the activities of distributed components. However, such systems exhibit peculiar traits that heavily challenge most of the coordination models and technologies currently available [2]:

- They require the capability of effectively coordinating the activities of a *huge number* (up to the millions) of *decentralised autonomous* components. This implies the impossibility of enacting some centralised scheme of coordination of the activities, as well as the impossibility of enforcing full control over the activities and interactions of some (if not most) of the components.
- Such systems are inherently *heterogeneous* and *socio-technical* [3], since they require orchestrating the activities of components as diverse as software agents, a variety of sensors and actuators, robots, and – last but not least – they involve the active contributions of humans with their peculiar capabilities and competences [4].
- They are *situated* in dynamic and unpredictable physical and social *environments*, where their components are required to be *context-aware* and *socially-aware* in their interaction, and where consequently any coordination scheme has to be adaptive to the context.

In the reminder of our position paper, we detail some of the above-mentioned emerging large-scale socio-technical scenarios, and discuss their peculiar coordination requirements, and how they call for novel models, languages, and infrastructure (Section II). Then, based on the potential impact of such systems on the area of coordination, we identify some of the new challenges and promising directions for theoretical research and technological development (Section III).

II. SCENARIOS

The scenarios we outline in this section are generally representative of some easy-to-envision future scenarios, in which millions of ICT sensors, actuators, and services will be called to operate in an orchestrated way, along with the active contribution of the sensing, actuating, and reasoning capabilities of humans [4].

A. Urban Traffic

The role of coordination for handling urban traffic has been widely acknowledged [5]. However, technological evolution mandates for new sorts of coordination models and technologies in the urban traffic scenarios. The drivers that will make urban traffic totally different from what it is now include *(i)* the rise of self-driving cars and *(ii)* the widespread diffusion of on-demand and social mobility services.

First, modern cars already include a number of features to alleviate driver activities (e.g., the capability of autonomously park), and fully autonomous self-driving car is going to hit the streets in a few years [6]. Such self-driving cars will have to coordinate their motion with each other in a cooperative way, and in a (mostly) norm-compliant way with urban computer-based infrastructures such as traffic lights. However, for quite some time, their will also have to interact in a mixed systems of millions of self-driving cars and human-driven ones.

Second, continuous connectivity to the Internet and the rise of social networks are enabling a variety of innovative social mobility models, such as unplanned on-the-fly ride-sharing, dynamic traffic slot allocation, dynamic on-demand schedule of public transport vehicles [7]. The basic idea is to dynamically match the mobility needs expressed by users by coordinating all the actors that can possibly satisfy such

needs, in a context-aware way and without neglecting the social aspects involved in the resulting coordination scheme.

B. Robotic and Human Teams

Critical missions such as armed conflicts and handling of natural disasters, which typically involve large teams of soldiers and volunteers, will increasingly involve autonomous weapon systems and robots as well [8], [9]. Autonomous unmanned aerial vehicles and – to some extents – unmanned ground vehicles, are already a reality, and an important support for human teams: many additional classes of robots, there included robotic soldier, are likely to be exploited in the near future.

Accordingly, there will be the need of coordinating the activities of possibly large-scale mixed teams of humans and robots in such critical missions. A key issue for coordination in these scenarios comes from the fact that the activities of the system components are situated in environments and situations that can hardly be known in advance, and therefore require the dynamical adaptation of both individual activities and the coordination schemes required to achieve their effective orchestration.

As a consequence, the scenario requires the adoption of coordination models (and of the corresponding coordination technologies) that could promote the dynamic and adaptive expression of different coordination patterns [10] depending on the situation of the environment.

C. Participatory Urban Management

Municipalities spend a lot of their money in trying to monitor and maintain our urban environments at the best. Activities include: garbage collection, maintenance of roads and public green, installation and maintenance of public lights, etc. Such activities involve a lot of human work – only a small portion of which calls for specialised skills or tools –, are very costly (due to the need of employ a lot of people), are and typically based on static planning.

Clearly, ICT technologies (such as cameras) make it possibile in principle to automate some of the sensing activities, and make them cheaper. However, a more radical evolution that can be promoted by smart phones and continuous connectivity is that of dynamically involving citizens in the above tasks, and make urban management a participatory activity. For instance, one could dynamically involve citizens to help mapping the noise level in a town, by having them supply the lack of appropriate sensors in some parts of the town [11], [12].

Such kind of participatory urban management activities clearly require bringing together the complimentary capabilities of humans and ICT devices, may involve a very large number of devices and humans, and require involving and coordinating humans in a context-aware way, depending on their current positions, goals, and activities.

D. Health Care

Pervasive health care is the new frontier according to a twofold perspective: the adoption of pervasive models and technologies to health care systems, and the ubiquitous availability (everywhere, to everybody) of healthcare [13]. Pervasive health care systems are then typically huge socio-technical systems, where millions of citizens, doctors, and operators need to coordinate through myriads number of interconnected devices, in order to organise health care activities around huge information sources and a wide range of health care hardware, often in critical conditions. Medical protocols need to be enforced, best procedures have to be promoted, whereas emergency operations should be supported in any moment, both on the local and on the global scale.

As a result, coordination in pervasive health care systems generally mandates for robustness, safety, and security: but, first of all, *dependability* and *efficiency* even on the large scale are essential to ensure that critical, possibly unplanned operations can be successfully brought to an end successfully. Also, the ability to deal with a huge and inordinately growing amount of knowledge, and to match it in real time with protocols and procedures is a fundamental requirement of pervasive health care systems, which coordination models and technologies are required to address.

III. Challenges

In this section we analyse some of the peculiar challenges that arise in large-scale socio-technical systems, such as the one exemplified above, and accordingly point out the main research issues that coordination models and technologies are demanded to address in the next years.

A. Human-ICT Coordination

The activities of forthcoming socio-technical systems will involve a variety of agents: humans with mobile devices, ICT sensors and actuators, cameras, self-driving cars, diverse sort of autonomous robots. The features and capabilities of such heterogeneous classes of entities are very different from each other: for instance, one may think at how differently (in terms of modality, timescales, accuracy) humans and artificial vision systems see and classify images, or at how differently robots and humans can assist people.

Accordingly, a coordination language should be expressive enough to enable the representation of coordination schemes among such different classes of components (and, of their associated services and capabilities), yet maintaining a uniform and clear model behind. Similarly, a coordination infrastructure should be able to implement and support such a model in an effective and scalable way, to support the orchestration of myriads of heterogeneous agents physically spread over an urban area. Such issues are so far largely unexplored, and worth being investigated.

B. Autonomy

One key feature of components and systems in nowadays application scenarios is *autonomy*, as clearly exemplified by the current transition from human-driven to self-driving cars. Many different acceptations of the term "autonomy", however, are adopted in different fields – computer science, philosophy, military, psychology, biology, among the many – which lead to different requirements and behaviours for systems and components: and, consequently, for their coordination.

In general, autonomy is typically bound to the existence of an inner goal, to be self-achieved—by the system as a whole, or by the component. Since effective coordination usually requires the mutual understanding of each other goals, the ability to associate actions in a shared environment with their motivating goals seems a relevant feature for coordination models, nowadays. This holds in particular for socio-technical systems, where humans may easily exhibit mixed, hidden, and even unaware goals, whereas artificial components could be either *goal-driven* or *goal-oriented*, according to their goal being implicitly or explicitly represented, respectively [14].

On the one hand, models of coordination supporting goal / plan exchange exist (e.g., [15]), but are often too specific to be adopted as general-purpose coordination models. On the other hand, general-purpose coordination models do not account for goals anyway expressed, nor for their association with coordinated activities. As a result, a first challenge for innovative coordination models and technologies will be to deal with the expression of components' goals, with their association with component's activities, and their visibility/hiding in a shared environment.

More generally, coordination models will be required to deal with large numbers of components of many sorts, featuring many diverse sorts of autonomous behaviours [16]—as such, with largely different coordination needs.

C. Context-awareness and Self-organisation

When it comes to coordinating the activities of large-scale decentralised systems of heterogeneous components situated in dynamic and unpredictable environments, nature may have something to teach us [17]. In fact, many large-scale natural systems that exhibit seemingly goal-oriented behaviours [18], achieve them by relying on self-organising coordination schemes that are inherently adaptive, as well as capable of tolerating dynamic environment and unpredictable contingencies.

In the past few years, we extensively investigated the possibility of exploiting nature-inspired approaches as a means to enable self-organising coordination in context-aware and situated pervasive computing systems [19]. In the context of the EU project SAPERE, for instance, we developed a novel coordination model based on distributed tuple spaces, relying on a simple set of nature-inspired coordination laws [20], [21].

However, despite the encouraging results achieved by the SAPERE project, what is still missing is a real assessment of whether nature-inspired coordination model can support the inherent heterogeneity of the emerging systems, and whether they can really scale to support millions of components and large-scale decentralised scenarios. Attacking such an issue would represent a very promising research direction.

D. Incentives for Participation

A key assumption of most coordination models and languages is that the coordinated components are under the control of engineers, and willing to be coordinated. However, when components belong to multiple stakeholders, or, when such components are humans expected to deliver some service, the effectiveness of the coordination scheme is also related to the effectiveness in incentivising components to participate in the coordination scheme and deliver the necessary services on need.

As far as human are involved, recent work on persuasive technologies analyses how to induce specific behavioural changes and persuade people to establish a desired behaviour [22]. We expect persuasive technologies will be an integral part of future coordination models infrastructure: nevertheless, understanding which forms such technologies could take in a model, and how they could be integrated in a coordination middleware is still an open, yet promising, research direction.

In any case, there could always be specific classes of services and behaviours for which persuasive technologies can hardly apply—e.g., convincing people to park farther than they would autonomously do, in order to provide for a better overall parking availability; or, convincing the owner of a robot to lend it for some time. Therefore, a coordination model should account for more explicit means to incentivise participation, such as monetary rewards or social rewards [23]. However, since the sustainability of such mechanisms and their general effectiveness in coordinated systems is far from having been assessed, there is plenty of room for research in these directions.

E. Economic Dimension

The emergence of cloud computing has clearly pointed out how the economic dimension of contemporary ICT services cannot be ruled out even from models. On large-scale socio-technical systems, money is no longer a secondary concern, which could be deferred after the system design and development: every minimal shift in the costs could produce huge unbalancing in the overall sustainability of systems. And, the availability of certain actions in in given situations is often bound to some measure of cost—differently expressed when actions are by humans (which are typically directly liable for costs) or by ICT components (which are more easily not directly liable)

As far as coordination is concerned, cost issues have generically often been included among the concerns: however, the economic dimension has rarely been explicitly considered. Accordingly, a promising direction for research on coordination is to design models and technologies that explicitly include the economical concerns, possibly associating both individual actions (by either humans or artificial agents) and coordinated activities to their costs (either actual or putative), and making it possible to express coordination policies in terms of economic issues.

F. Knowledge Intensive Environments

The vast availability of data, information, and knowledge, along with their strong dynamics (e.g., data streams) is a typical feature of nowadays complex systems—in particular, large-scale socio-technical systems, where millions of humans and software agents interact by exploiting and exchanging huge amounts of data and information. On the one hand, both human and ICT agents in socio-technical systems typically work as powerful knowledge sources, accumulating and sharing information and data, and act according to knowledge and beliefs, both owned by individuals and shared. On the other hand, shared environments where coordination activities take

place are typically knowledge-intensive environments, so that both individual actions and social activities also depend on the amount, sort, and accessibility of data and information in the shared environment [24], [25].

This is why innovative coordination models and technologies will be more and more required to deal with large amount of information and data, to provide support for knowledge modelling and representation, to promote knowledge-rich interactions, and to express knowledge-dependent coordination policies.

IV. Conclusions

In this position paper we argue that the peculiar features of emerging socio-technical software-intensive systems – up to millions of decentralised situated components, lacking central control, mixing humans and ICT components – call for innovative coordination models and technologies. Accordingly, we discuss the key challenges to be faced by researchers in the area of coordination, and accordingly sketch promising research directions.

As a part of our current research, we are working on nature-inspired coordination models and languages [20], [17], which we consider as a promising approach to tackle the identified challenges.

References

[1] M. Wirsing, J.-P. Banatre, M. Hölzl, and A. Rauschmayer, Eds., *Software-Intensive Systems and New Computing Paradigms: Challenges and Visions*, ser. Lecture Notes in Computer Science. Springer, 2008, vol. 5380.

[2] A. Omicini and M. Viroli, "Coordination models and languages: From parallel computing to self-organisation," *The Knowledge Engineering Review*, vol. 26, no. 1, pp. 53–59, Mar. 2011.

[3] N. R. Jennings, L. Moreau, D. Nicholson, S. D. Ramchurn, S. J. Roberts, T. Rodden, and A. Rogers, "Human-agent collectives," *Communications of the ACM*, vol. 57, no. 12, pp. 80–88, Dec. 2014.

[4] F. Zambonelli, "Toward sociotechnical urban superorganisms," *IEEE Computer*, vol. 45, no. 8, pp. 76–78, 2012.

[5] S. Ossowski, *Co-ordination in Artificial Agent Societies. Social Structures and Its Implications for Autonomous Problem-Solving Agents*, ser. Lecture Notes in Artificial Intelligence. Springer, 1999, vol. 1535.

[6] G. J. Offer, "Automated vehicles and electrification of transport," *Energy & Environmental Science*, vol. 8, no. 1, pp. 26–30, 2015.

[7] A. Sassi and F. Zambonelli, "Coordination infrastructures for future smart social mobility services," *IEEE Intelligent Systems*, vol. 29, no. 5, pp. 78–82, 2014.

[8] S. Kohlbrecher, A. Romay, A. Stumpf, A. Gupta, O. von Stryk, F. Bacim, D. A. Bowman, A. Goins, R. Balasubramanian, and D. C. Conner, "Human-robot teaming for rescue missions: Team ViGIR's approach to the 2013 DARPA robotics challenge trials," *Journal of Field Robotics*, 2014.

[9] N. Schurr, J. Marecki, M. Tambe, and P. Scerri, "Towards flexible coordination of human-agent teams," *Multiagent and Grid Systems*, vol. 1, no. 1, pp. 3–16, 2005.

[10] G. Cabri, N. Capodieci, L. Cesari, R. De Nicola, R. Pugliese, F. Tiezzi, and F. Zambonelli, "Self-expression and dynamic attribute-based ensembles in SCEL," in *Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change*, ser. Lecture Notes in Computer Science, vol. 8802, 2014, pp. 147–163.

[11] S. Hachem, A. Pathak, and V. Issarny, "Service-oriented middleware for large-scale mobile participatory sensing," *Pervasive and Mobile Computing*, vol. 10, pp. 66–82, 2014.

[12] E. D'Hondt, J. Zaman, E. Philips, E. G. Boix, and W. De Meuter, "Orchestration support for participatory sensing campaigns," in *2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2014, pp. 727–738.

[13] I. Korhonen and J. Bardram, "Guest editorial: Introduction to the special section on pervasive healthcare," *IEEE Transactions on Information Technology in Biomedicine*, vol. 8, no. 3, pp. 229–234, 2004.

[14] C. Castelfranchi, A. Cesta, R. Conte, and M. Miceli, "Foundations for interaction: The dependence theory," in *Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science. Springer, 1993, vol. 728, pp. 59–64.

[15] D. Ancona, V. Mascardi, J. Hubner, and R. Bordini, "Coo-AgentSpeak: Cooperation in AgentSpeak through plan exchange," in *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, Jul. 2004, pp. 696–703.

[16] S. Mariani, A. Omicini, and L. Sangiorgi, "Models of autonomy and coordination: Integrating subjective & objective approaches in agent development frameworks," in *Intelligent Distributed Computing VIII*, ser. Studies in Computational Intelligence, L. Braubach, D. Camacho, S. Venticinque, and C. Bădică, Eds., vol. 570. Springer International Publishing, 2015, pp. 69–79, 8th International Symposium on Intelligent Distributed Computing (IDC 2014), Madrid, Spain, 3-5 Sep. 2014. Proceedings.

[17] A. Omicini, "Nature-inspired coordination for complex distributed systems," in *Intelligent Distributed Computing VI*, ser. Studies in Computational Intelligence, G. Fortino, C. Bădică, M. Malgeri, and R. Unland, Eds., vol. 446. Springer, 2013, pp. 1–6.

[18] V. Parunak, "Go to the ant: Engineering principles from natural multi-agent systems," *Annals of Operations Research*, vol. 75, pp. 69–101, 1997.

[19] F. Zambonelli and M. Viroli, "A survey on nature-inspired metaphors for pervasive service ecosystems," *Journal of Pervasive Computing and Communications*, vol. 7, pp. 186–204, 2011.

[20] F. Zambonelli, A. Omicini, B. Anzengruber, G. Castelli, F. L. DeAngelis, G. Di Marzo Serugendo, S. Dobson, J. L. Fernandez-Marquez, A. Ferscha, M. Mamei, S. Mariani, A. Molesini, S. Montagna, J. Niemi-nen, D. Pianini, M. Risoldi, A. Rosi, G. Stevenson, M. Viroli, and J. Ye, "Developing pervasive multi-agent systems with nature-inspired coordination," *Pervasive and Mobile Computing*, vol. 17, pp. 236–252, Feb. 2015, Special Issue "10 years of Pervasive Computing" in Honor of Chatschik Bisdikian.

[21] F. Zambonelli, G. Castelli, L. Ferrari, M. Mamei, A. Rosi, G. Di Marzo Serugendo, M. Risoldi, A.-E. Tchao, S. Dobson, G. Stevenson, Y. Ye, E. Nardini, A. Omicini, S. Montagna, M. Viroli, A. Ferscha, S. Maschek, and B. Wally, "Self-aware pervasive service ecosystems," in *Proceedings of the 2nd European Future Technologies Conference and Exhibition 2011 (FET 11)*, ser. Procedia Computer Science, E. Giacobino and R. Pfeifer, Eds., vol. 7. Budapest, Hungary: Elsevier Science B.V., 4–6 May 2011, pp. 197–199.

[22] B. Fogg, *Persuasive Technology: Using Computers to Change What We Think and Do*. Morgan Kaufmann, Dec. 2002.

[23] O. Scekic, H.-L. Truong, and S. Dustdar, "Incentives and rewarding in social computing," *Commun. ACM*, vol. 56, no. 6, pp. 72–82, Jun. 2013.

[24] S. Mariani and A. Omicini, "Molecules of Knowledge: Self-organisation in knowledge-intensive environments," in *Intelligent Distributed Computing VI*, ser. Studies in Computational Intelligence, G. Fortino, C. Bădică, M. Malgeri, and R. Unland, Eds., vol. 446. Springer, 2013, pp. 17–22.

[25] ——, "MoK: Stigmergy meets chemistry to exploit social actions for coordination purposes," in *Social Coordination: Principles, Artefacts and Theories (SOCIAL.PATH)*, H. Verhagen, P. Noriega, T. Balke, and M. de Vos, Eds., AISB Convention 2013, University of Exeter, UK, 3–5 Apr. 2013, pp. 50–57.

A Hybrid Agent Architecture for Endowing Floor Field Pedestrian Models with Tactical Level Decisions

Luca Crociani, Alberto Invernizzi, Giuseppe Vizzari
CSAI - Complex Systems & Artificial Intelligence Research Center,
University of Milano-Bicocca, Milano, Italy
{luca.crociani,giuseppe.vizzari}@disco.unimib.it
alby.inve@gmail.com

***Abstract*—For a comprehensive modeling of pedestrian dynamics in real-world scenarios the consideration of tactical level decisions in addition to operational ones is necessary. This paper presents a hybrid agent architecture employing a Floor Field approach at the operational level but granting agents an abstract representation of the simulated environment. The paper briefly presents the environmental model and hybrid agent architecture based on the floor field approach, then a sample practical application in a simple case study is also presented to show how it allows specifying abstract behavioural *scripts* for different groups of agents.**

***Index Terms*—pedestrian simulation, agent-based modeling, floor-field model, hybrid agents, environments for multi-agent systems**

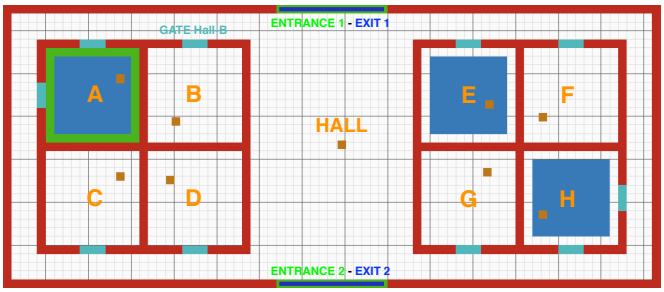
I. Introduction

The Floor Field approach to the modeling and simulation of pedestrian dynamics, first introduced by [1], represents a viable option for the implementation of quantitatively validated simulation systems based on a discrete approach to the representation of the environment. However, a comprehensive simulation system for pedestrian dynamics in real-world scenarios requires the consideration of tactical level decisions in addition to operational ones, as discussed by [2], that are the main focus of the Floor Field approach. This paper presents a hybrid agent architecture essentially employing a Floor Field approach at the operational level but providing agents an abstract representation of the simulated environment for tactical level deliberation, a map automatically derived from an annotated CAD-like description of the environment in which the simulation must take place. This form of knowledge is essentially a labeled graph in which nodes are associated to regions and links represent connections among them; links and other relevant points in the environment are associated to static floor fields allowing agent navigation at the operational level. Considering, instead, tactical level aspects, agents are provided with a goal, a final target destination potentially enriched by intermediate steps and movement constraints; they initially autonomously inspect their knowledge and derive a plan indicating intermediate destinations, associated to specific static floor fields to be followed. The paper will briefly present the environmental model, including a base CAD-like geometric representation from which both the abstract representation and a set of layers associated to static floor fields are automatically constructed. The tactical level extension of a previous agent-based model, also allowing the management of groups of pedestrians, introduced by [3] will then be described to show how this level and the existing operational layer interact. Finally, a sample practical application in a simple case study is also presented to show how it allows specifying abstract behavioural *scripts* for different groups of agents.

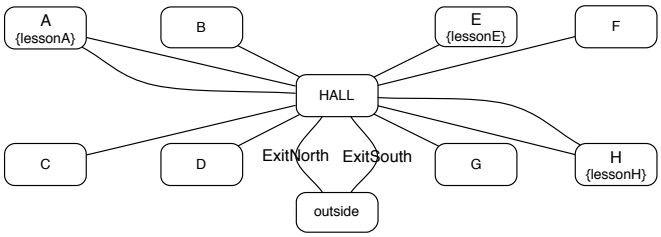
II. Environment

As discussed by [4], the environment of an agent-based system is "a first class-abstraction that provides the surrounding conditions for agents to exist and that mediates both the interaction among agents and the access to resources". An environment for agent-based systems can encompass both abstractions and mechanisms, for instance regulating the outcomes of agents' chosen lines of action, as discussed by [5]. Within this framework, for this particular application of an agent-based modeling and simulation approach, the environment does not only encompass a spatial representation of the simulated area, but also a set of abstractions and data structures (e.g. static floor field matrices) enabling agents' perceptions, deliberations and actions. In particular, for our purposes we need (i) a discretization describing the walkable area subdivided into cells of configurable size (e.g. 40 cm sided square cells); (ii) a similar discrete layer representing the effect of obstacles on the overall cell desirability; (iii) similar discrete layers representing the static floor field associated to a given point of reference/interest; (iv) a graph-like abstract representation of relevant sub-areas in the simulated space connected according to the reachability relationship.

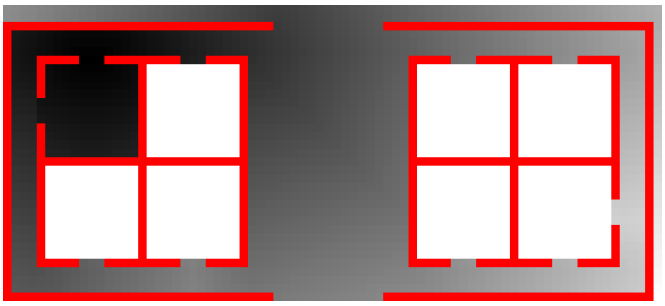
In order to support an automated production of the above elements and related data structures, a spatial representation of the area in which the simulation must take place, in the form of a CAD-like file, is required: on the other hand, this kind of map is generally produced when planning the construction of a building or available to managers of a premise. In order to allow algorithms to actually explore this representation and make sense of it, the designer is required to produce some



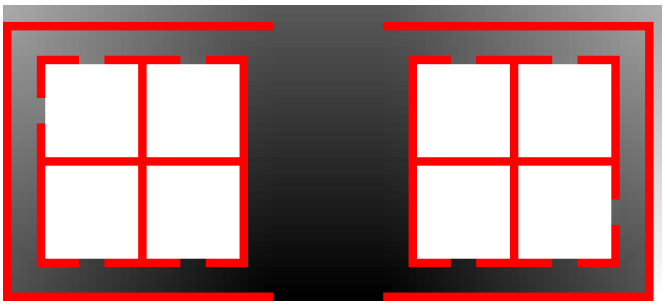
(a) CAD-like representation of the environment.



(b) Abstract commonsense representation of the environment.



(c) Static floor field associated to the gateway between "Hall" and "A".



(d) Static floor field associated to the southern exit.

Fig. 1. Relevant elements of pedestrian agents' environment.

form of *annotation* in it, as exemplified in Figure 1(a). In particular, the sub-areas in which the environment is divided into must be constrained by *obstacles* (in red in the figure), or passages, *gateways* to another sub-area (in cyan), and they must contain a specific block indicating a label that will be associated to the area. Both gateways and these *label blocks* are annotations that do not influence the walkability of the associated cells. Additional annotations represent *start areas*, in which pedestrian agents can be created (either initially or even at later stages of the simulation), *end areas*, final targets of movements in which pedestrian agents actually exit the simulation, and *intermediate destinations* (also associated to labels, not shown in the figure) that pedestrians must reach at a certain point of a more articulated movement plan.

The construction of such a plan requires the possibility to explore and process a much simpler data structure, in particular an abstract map in terms of a graph-like commonsense representation of the environment, as discussed by [6]. This structure, also exemplified in Figure 1(b), can be automatically derived by the annotated CAD-like representation employing an algorithm that cannot be reported here for sake of space. We want to emphasize here the fact that intermediate destinations (such as the one included in area A, labeled as "lessonA") are essentially included in the sub-area they are part of. Moreover, final exits are represented as annotated edges (ExitNorth and ExitSouth in the figure) leading to a vertex not associated to a sub-area in the CAD-like representation of the environment but rather related to the "outside" world. As we will discuss in the following section, this structure is particularly suited for simple path planning algorithms that can be employed in agent's tactical level.

Instead, for managing operational level tasks in the Floor Field approach, additional discrete grids containing gradient-like structures supporting agents' navigation of the environment are necessary. Examples of these data structures are shown in Figure 1(c) and 1(d), respectively related to the static floor fields leading towards the gateway between the sub-areas labeled as "Hall" and "E" and the southern exit of the scenario. Once again, the annotated CAD-like representation of the environment supports the automated generation of these layers, by means of a simple cellular automaton whose description is omitted here for sake of space. Please notice that, however, we chose not to extend the diffusion of the static floor field associated to an area or marker to all the discrete representation of the environment, but to limit this operation to the sub-areas that are in direct connection to the target in the abstract commonsense representation. This, on one hand, simplifies the environment set up phase (especially considering relatively large environments, in which it would not be practically feasible to do this) and, on the other, is sufficient since the agents will be provided with a tactical level behavioural model allowing them to generate plans requiring the perception of these fields only in these adjacent sub-areas.

Additional layers are actually included in the agent environment to support the gathering of statistics about their dynamics, but also their interactions (in particular, the mutual perception of members of groups) and the management of conflicts (movement intentions are stored into one of these additional layers to support a simple identification and management of the conflicts by the environment itself).

III. Agent Architecture

Considering the above structure for the agent environment, it is clear that the information provided to agents' perceptions is sufficient to support basic operational level behaviour for pedestrian agents. In fact, whenever an agent knows where

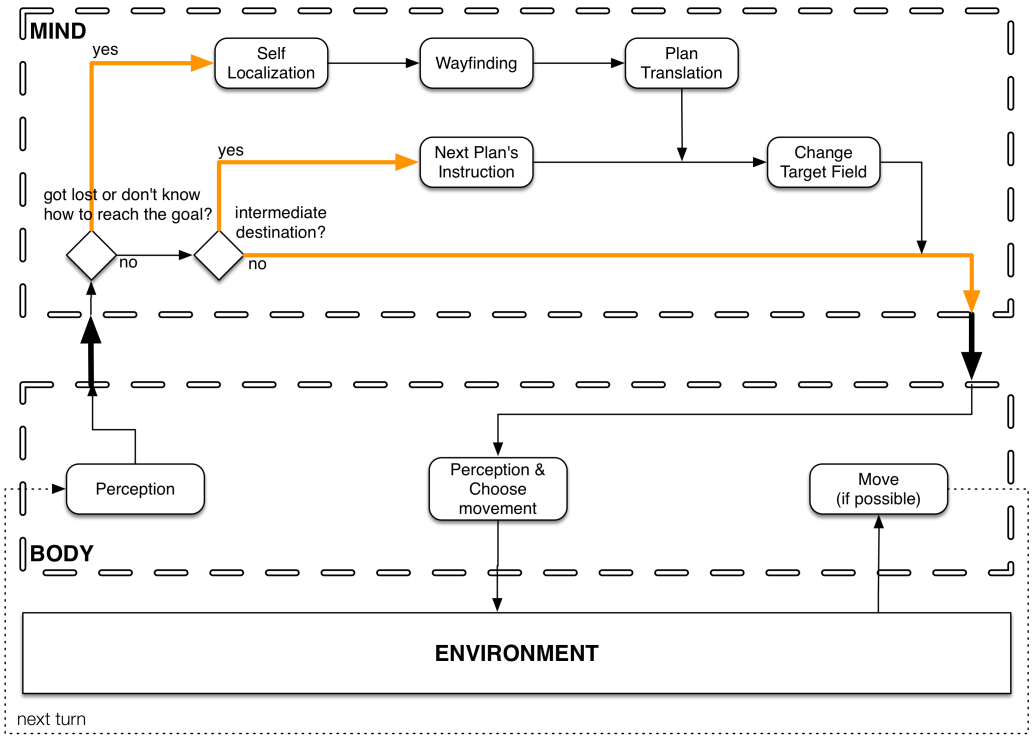


Fig. 2. Hybrid agent architecture: the operational level manages perceptions and actions, invoking the tactical level whenever plan creation or management is required.

it's headed (i.e. can perceive the static floor field associated to that destination), the basic floor field model is sufficient to allow the agent to achieve its own movement goal. However, as mentioned in the previous section, the static floor fields is not spread in the complete discrete representation of the environment, so an agent must actually plan a course of action, implying a sequence of intermediate way-points associated to other static floor fields, leading to an area in which the target is finally perceivable. This particular reasoning requires a *hybrid* architecture of the agent, whose "body" component reproduces the pure reactive behaviour (i.e. the raw movement at operational level), while a "mind" is dedicated to this cognitive level reasoning, aimed at achieving a sequence of fields to follow in order to reach the final target. A schematic description of the devised agent architecture is shown in Figure 2: the operational level layer (denoted as body) is actually an implementation of the extension to the floor-field model described by [3][1] slightly modified to trigger the computation of tactical level choices (carried out by the layer denoted as mind) whenever it is necessary. An obvious condition for the activation of the tactical level is the fact that an agent has a final goal that is not immediately perceivable at operational level. For instance, in the sample environment introduced in the previous section, an agent situated in the sub-area "E" cannot perceive the floor field associated to the northern exit. Nonetheless, agents are provided with the spatial knowledge associated to the abstract commonsense representation of the environment, which is a data structure accessible by their tactical level. Therefore, since they are able to understand what is their current location in this structure, they can search for their goal in the abstract map and in particular the intermediate steps leading towards this goal. The result of this search operation (that employs state of the art graph search algorithms whose description is omitted for sake of space), leads to the construction of a plan in the form of a set of operating instructions (in the vein of [8]) as depicted in Figure 3. In this case, an abstract plan leading an agent from the "Hall" to the lecture hall where "Lesson E" is held, then to the hall where "Lesson H" is held, then finally to leave the building through the southern exit will be expanded to become a sequence of seven steps (intermediate passages through gateways among rooms require a specific operating instruction).

The current implementation of the search algorithm represents a basic approach, that does not consider agent preferences (e.g. avoid stairs and use escalators), or the crowding conditions of the environment (e.g. when the most direct path is getting too congested, change the plan), but these extensions have already been considered for extended search strategies: recent developments along this line of research are described in another contribution in this volume [9].

IV. Example Application

The example application of the hybrid agent model is related to the previously described environment; in particular, it is a rough representation of a building including a set of lecture halls (labeled from "A" to "E"), all of which reachable from a central hall (actually including a surrounding corridor), with a northern and southern exits.

Within this scenario, exploiting the previously described tactical level extension, it is relatively simple to model different

[1]Additional details about the implementation are provided by [7].

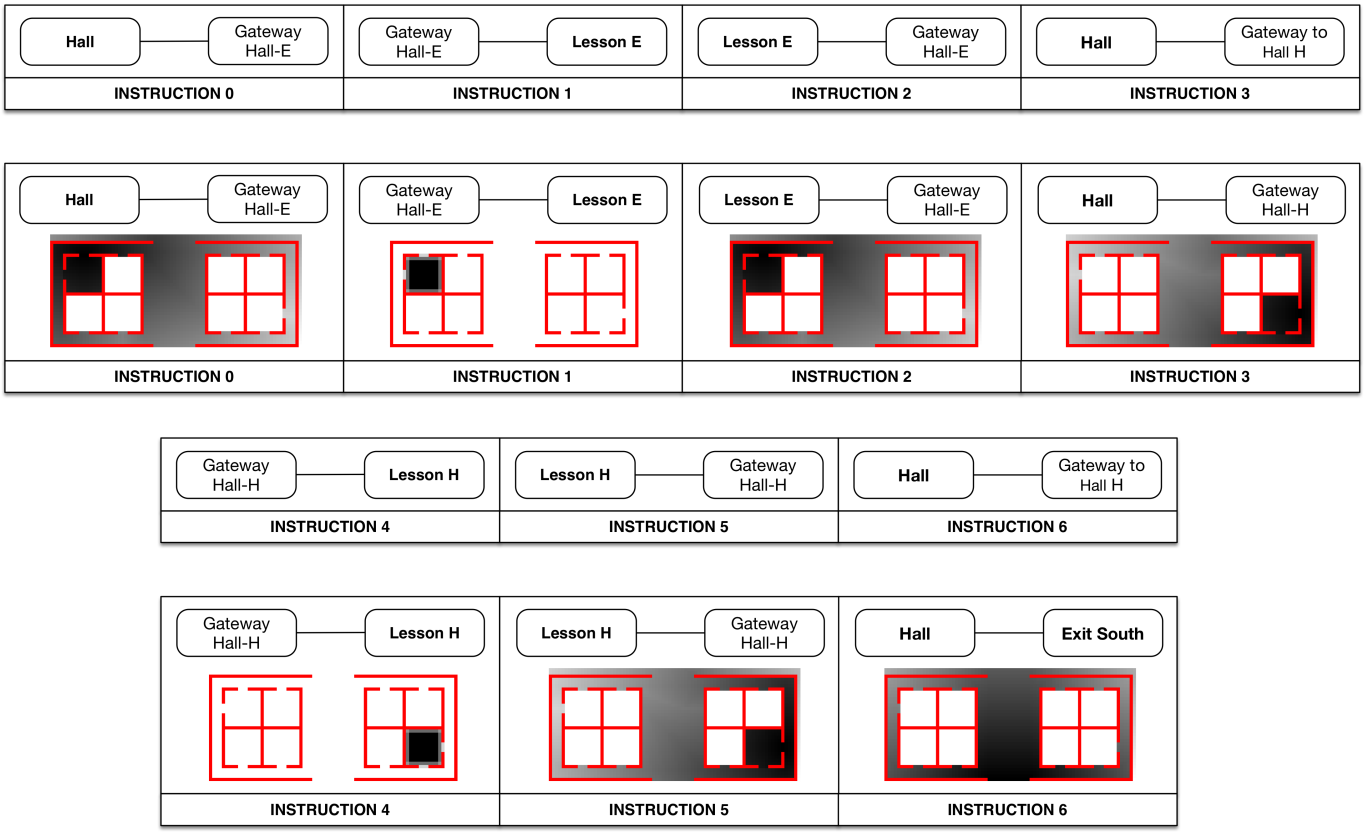


Fig. 3. Operating instructions for a plan leading from "Hall" to the lecture hall where "Lesson E" is held, then to the hall where "Lesson H" is held, then finally to leave the building through the southern exit.

groups of pedestrian agents associated to different groups of students, having different timetables. For instance we could define three groups (Green, Yellow and White agents) whose tasks in the environment are the following:

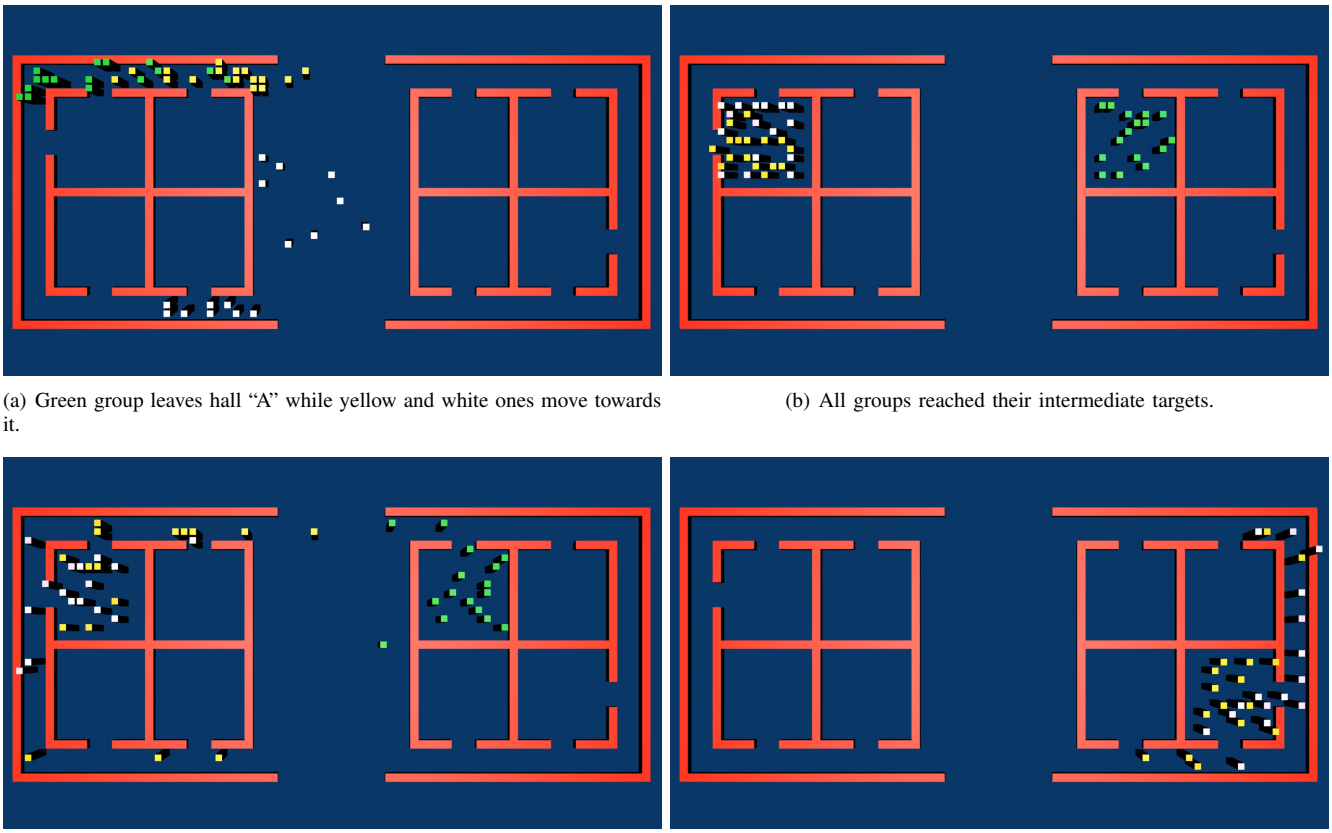
Green $[A \rightarrow lessonE \rightarrow Exit\{0.5\,North; 0.5\,South\}]$
Yellow $[EntranceNorth \rightarrow lessonA \rightarrow lessonH \rightarrow Exit\{0.5\,North; 0.5\,South\}]$
White $[EntranceSouth \rightarrow lessonA \rightarrow lessonH \rightarrow Exit\{0.5\,North; 0.5\,South\}]$

The green agents, in other words, are already in the environment at the beginning of the simulation, in particular in lecture hall "A", then they have to move to the sub-area in which "lessonE" takes place (i.e. lecture hall "E") and finally they must leave the environment, half of them through the northern exit and the other half through the souther one. Yellow and white agents, instead, enter the simulation area respectively through the northern and southern entrances, then move to the sub-area where "lessonA" takes place (lecture hall "A"), then they have to move to "lessonH" (lecture hall "H") and finally leave the environment, also splitting equally between northern and southern exits.It must be stressed the fact that the modeler is not forced to precisely and extensively define the path to be followed by the agents employing it, which will expand an abstract plan into a proper sequence of actionable operational level movements according to their tactical level knowledge. The yellow agents' plan, for instance, will be expanded into a sequence that is analogous to the one described in Figure 3.

Some screenshots of this simulation are shown in Figure 4. In particular, in Figure 4(a) green agents are exiting hall "A" and moving towards hall "E" (using both exits of hall "A"), while yellow and white agents are moving to hall "A". In Figure 4(b) the three groups have reached the respective destinations and, later on in Figure 4(c), the green group leaves the environment while the yellow and white groups move towards lecture hall "H". Finally, in Figure 4(d), both groups are reaching their final lesson before leaving the environment. The fact that agents from the same group employ different paths to reach the same movement target may depend, on one hand, on the nature of the static floor field layer, but also on the fact that, at the tactical level, they may choose different intermediate steps (e.g. there are two gateways leading from the main hall to lecture hall "A").

V. Conclusions and Future Developments

The paper has shown an extension of a floor field model to encompass tactical level tasks and information. We introduced a particular structure of agents' environment allowing them to perceive and act at the operational level, but also deliberate at tactical level: environmental data structures are automatically generated starting from an annotated CAD-like environment description. The hybrid agent architecture, including a simple reactive operational level able to trigger deliberation activities of the tactical level whenever it is necessary, has also been introduced. A sample application illustrating how this approach



(a) Green group leaves hall "A" while yellow and white ones move towards it.

(b) All groups reached their intermediate targets.

(c) Green group leaves the environment, while yellow and white ones move towards hall "H".

(d) Yellow and white groups have almost completed their movement towards hall "H".

Fig. 4. Application in a university building scenario.

allows specifying simple behavioural scripts for relatively complicated agent's plans has finally been described. Future works are aimed, on one hand, at supporting the possibility to enrich the abstract commonsense spatial representation for allowing tactical level reasoning about information like estimated distances, level of crowdedness of visible areas or passages, additional relevant information (e.g. a certain area is a steep ramp or staircase, which would hinder the movement of elderlies or persons on wheelchairs). Moreover, we are also going to extend agents' behavioural specification to allow the coordination of group path planning and the definition of area activities, in the vein of [10].

Acknowledgements

This work was partly supported by the ALIAS project ("Higher education and internationalization for the Ageing Society"), funded by Fondazione CARIPLO.

References

[1] C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz, "Simulation of pedestrian dynamics using a two-dimensional cellular automaton," *Physica A: Statistical Mechanics and its Applications*, vol. 295, no. 3 - 4, pp. 507 – 525, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437101001418

[2] A. Schadschneider, W. Klingsch, H. Klüpfel, T. Kretz, C. Rogsch, and A. Seyfried, "Evacuation dynamics: Empirical results, modeling and applications," in *Encyclopedia of Complexity and Systems Science*, R. A. Meyers, Ed. Springer, 2009, pp. 3142–3176.

[3] G. Vizzari, L. Manenti, and L. Crociani, "Adaptive pedestrian behaviour for the preservation of group cohesion," *Complex Adaptive Systems Modeling*, vol. 1, no. 7, 2013.

[4] D. Weyns, A. Omicini, and J. Odell, "Environment as a first class abstraction in multiagent systems," *Autonomous Agents Multi-Agent Systems*, vol. 14, no. 1, pp. 5–30, 2007.

[5] S. Bandini and G. Vizzari, "Regulation function of the environment in agent-based simulation," in *Environments for Multi-Agent Systems III, Third International Workshop, E4MAS 2006, Hakodate, Japan, May 8, 2006, Selected Revised and Invited Papers*, ser. Lecture Notes in Computer Science, D. Weyns, H. V. D. Parunak, and F. Michel, Eds., vol. 4389. Springer–Verlag, 2007, pp. 157–169.

[6] S. Bandini, A. Mosca, and M. Palmonari, "Common-sense spatial reasoning for information correlation in pervasive computing," *Applied Artificial Intelligence*, vol. 21, no. 4&5, pp. 405–425, 2007.

[7] L. Crociani, L. Manenti, and G. Vizzari, "Makksim: Mas-based crowd simulations for designer's decision support," in *PAAMS*, Y. Demazeau, T. Ishida, J. M. Corchado, and J. Bajo, Eds., vol. 7879. Springer, 2013, pp. 25–36.

[8] M. Viroli, A. Ricci, and A. Omicini, "Operating instructions for intelligent agent coordination," *The Knowledge Engineering Review*, vol. 21, no. 01, pp. 49–69, 2006.

[9] L. Crociani, A. Piazzoni, and G. Vizzari, "Adaptive hybrid agents for tactical decisions in pedestrian environments," in *Proceedings of WOA 2015 - XVI Workshop "From Objects to Agents"*, 2015.

[10] J. Was and R. Lubaś, "Towards realistic and effective agent-based models of crowd dynamics," *Neurocomputing*, vol. 146, pp. 199–209, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231214007838

Composing Cognitive Agents from Behavioural Models in PRESTO

Paolo Busetta
Delta Informatica Spa
Trento, Italy
Email: paolo.busetta@deltainformatica.eu

Mauro Dragoni
FBK
Trento, Italy
Email: mauro.dragoni@fbk.eu

***Abstract*—The PRESTO project applies agent technologies to serious gaming. PRESTO has developed an AI infrastructure and an agent framework called DICE for the creation of game-independent, modular Non-Player Characters (NPC) behaviours based on a BDI (Belief-Desire-Intention) approach enriched with cognitive extensions for human simulation. Behavioural models can be combined via end-user development tools to form the behavioural profiles of NPCs in a game. DICE provides the coordination between body-controlling behavioural models (for navigation as well as posture, facial expressions, actioning) and decision-making models representing e.g. the standard operating procedures of professional roles, the cognitive appraisal of events and perceptions, the modality of reaction to unplanned events occurring during a game. Behavioural models are largely if not completely independent of the specific scenario or even game engine thanks to abstractions of both the environment and the internal state of the NPC provided by means of ontologies. PRESTO is producing a set of behavioural models targeted at its pilot project's needs or expected to be of common use, including navigation in the virtual environment sensitive to the cognitive state of the NPC. This paper gives a brief overview of PRESTO, DICE and of its ontologies.**

I. Introduction

PRESTO (Plausible Representation of Emergency Scenarios for Training Operations) [2], [3] aims at adding semantics to a virtual environment and modularising the artificial intelligence controlling the behaviours of NPCs (Non Player Characters, i.e. artificial players in a game). Its main goal is to support a productive end-user development environment directed to trainers building scenarios for serious games (in particular to simulate emergency situations such as road and industrial accidents, fires and so on) and in general to game masters wanting to customize and enrich the human player's experience. The framework for behavioural modeling in PRESTO, called DICE, was inspired by a BDI (Belief-Desire-Intention) [1], [9] multi-agent system with cognitive extensions, CoJACK [10], [6]. PRESTO offers powerful end-user development tools for defining the parts played by virtual actors (as end user-written behaviours) and the overall session script of a game. PRESTO supports a specific virtual reality, XVR from E-Semble, a well known tool in use for Emergency Management and Training (EMT) in a number of schools and organisations around the world, as well as Unity 3D and, at least in principle, is agnostic with respect to the game engine in use.

The next section explains the motivations behind PRESTO with an example and gives an overview of the overall system, while Sec. III briefly presents DICE. The motivations and design of the semantic and end-user facilities are sketched out in Sec. IV. Sec. V provides details about the navigation subsystem and how it is affected by the cognitive state of agents. Sec. VI briefly presents the work currently in progress in the area of coordination, while Sec. VII summarizes the state of development and experimentation at the time of writing.

II. Directing NPCs as virtual actors in a virtual stage.

Serious games have the potential to dramatically improve the quality of training in a number of fields where the trainee has to face complex and potentially life-threatening situations. In particular, open-world 3D simulations (also called ''sand-box" or ''free-roaming" games) have been used for quite a long time by the military, with a few products reaching a significant market success, and are becoming common in civilian emergency training because they allow the rapid construction of scenarios for the rehearsal of safety procedures. The main limitation of current technology concerns NPCs, whose behaviour may be quite sophisticated when performing predefined tasks but is often unaffected by context; further, a professional programmer is required for the implementation of any procedure that cannot be described with the simple selection of a few waypoints and the choice of a few actions, let alone introducing variants due to psychological factors. These issues lead to repetitive and hardly credible scenarios and to the slow and costly development of new ones when many NPCs are involved.

As an example, consider a fire breaking in a hospital ward during daytime with patients with different impairments, visitors of various ages and professionals with different roles, experiences and training. In this scenario, which is taken from the pilot project of PRESTO, most characters are NPCs while the human players, i.e. the trainees, are either health professionals that could be in charge for a ward at the time of an accident or emergency staff called to help. A training session would require two apparently conflicting abilities from NPCs. From the one hand, they should act autonomously according to a variety of parameters concerning e.g. their physical and psychological state, their current position, their capabilities; e.g. visitors may act rationally and follow well-marked escape routes or flee panicking to the closest exits, nurses at the start of their shift are fully responsive and careful while at the end of the shift fatigue may lead to errors, and so on. On the other hand, in order to make training effective

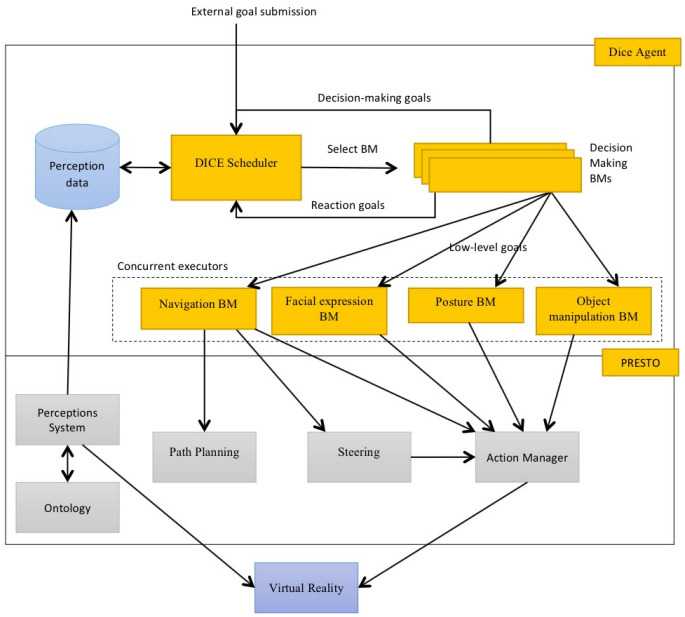


Fig. 1. Simplified DICE architecture with navigation highlighted (BM: Behavioural Model)

and engaging, the trainer supervising a simulation session should be able to temporarily suspend it (e.g. to give feedback to the trainees), change the course of events or affect the way certain characters behave (e.g. to introduce more drama or rehearse different procedures), as well as introducing or removing characters in following runs of the same scenario. Hard-coding all possibilities, assuming that this is supported by the game in use, is a laborious task to say the least.

The objective of PRESTO is to allow NPCs to act as "virtual actors" because they are able to "interpret" a part written at a higher level of abstraction than with common scripting languages, with additional modalities (that may correspond to, e.g., levels of skills or psychological profiles) that can be selected at the beginning but changed during a game as a result of the application of rules or by explicit user choice. The game's master (i.e. the trainer) is empowered to become a "director" able to "brief" virtual actors, that is, to define the parts the artificial characters have to play by means of a language aimed to non-programmers that composes more fundamental even if potentially very complex behaviours into game-specific sequences. Key enablers are end-user development tools [7] and the ability to mix and match behavioural components taken off-the-shelf from a market place (similar in principle to asset stores in popular gaming platforms such as Unity).

III. NPC Cognitive Architecture

PRESTO creates a DICE agent for each NPC in a game according to scenario-specific configurations.

DICE (Fig. 1) is a BDI framework that supports multi-goal modeling of NPC behaviours, where navigation, body postures and facial expressions, manipulation of objects and decision-making concerning tactical and long-term objectives are controlled by concurrent threads (implemented, in BDI speak, as intention trees achieving independent hierarchies of goals and subgoals). Furthermore, decision-making in DICE happens at two levels, controlled by independent "planned" and "reaction" intention trees. A decision-making behaviour started in reaction to an event pre-empts and blocks the execution of a planned behaviour until it is fully completed, at which point the planned behaviour is resumed. This allows, for instance, to have short-term reactions to perceptions (such as hearing a noise) that partially change the NPC state (e.g. by pointing the head towards the source of the noise) while not affecting navigation or longer-term procedures if not required. All behaviours in the body-controlling intention trees and in decision-making can be overriden by new behaviours at any time, e.g. as new perceptions are processed, as part of a decision-making routine, as a user choice from a GUI, as a command from a PRESTO session-controlling script; at any time, no more than one behaviour for each intention tree is active.

A DICE agent is built by composing so-called ''behavioural models", which are BDI capabilities [5] able to achieve a predefined set of goals (''role"). Changes in behaviour due to emotions, fatigue or other non-rational factors can be dealt within DICE in various ways, of which the most novel (and dramatic) is by defining behavioral rules that select alternative behavioural models according to the current cognitive state of the NPC. These rules can be defined directly by the end user, who is enabled to change the behavioural profiles of her characters according to the evolution of the game or even in real-time by explicit choice and from the session-level script. As in CoJACK [10], cognitive states are represented in DICE by moderators (i.e. numeric values modeling specific factors such as fear and fatigue levels) and a set of cognitive parameters computed from those moderators (modeling e.g. reactivity and accuracy), even if greatly simplified with respect to the original. Any behavioural model can use moderators and cognitive parameters to tune its own internal parameters, e.g. to decide the speed of execution of action or memory fading. Changes to moderators are normally performed by behavioural models for cognition according to appraisal rules (concerning e.g. the perception of threatening things) and time; however, it is possible to force the value of moderators at any time from any behavioural model (e.g. because of the realization of a dangerous situation) or from the session-controlling script, thus allowing the trainer to fully control the overall behaviour of an NPC during a game.

Figures 2 and 3 illustrate a (simplified) example NPC profile built as a DICE agent. The example is of a shopper able to achieve a "shop visited" goal. Fig. 2 provides a static view, organized in descriptions of capabilities (''roles"), behaviours implementing those capabilities and switching rules based on moderators that select which behavioural models are currently active. Observe that a single role (e.g. "Shopper" in the figure) may correspond to multiple behavioural models; the switching rules determine the current profile of the NPC, i.e. which models are active for each role at run-time. Roles and behavioural models are described by metadata, and they can be composed and configured by means of graphical editors.

Fig. 3 represents a snapshot of the dynamic state of the agent, with the concurrently executing intention trees and the two-level decision making. One of the implications of

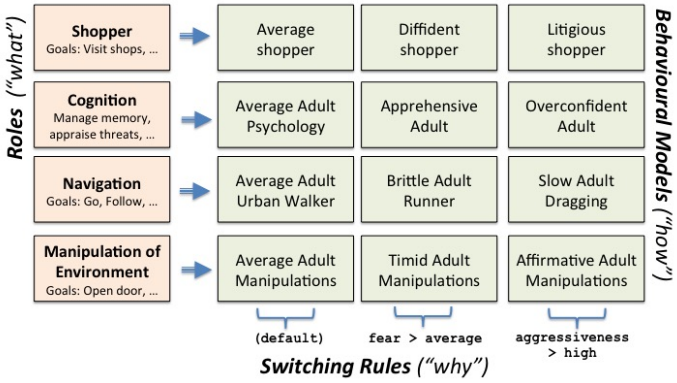


Fig. 2. Simplified DICE profile of a shopper NPC

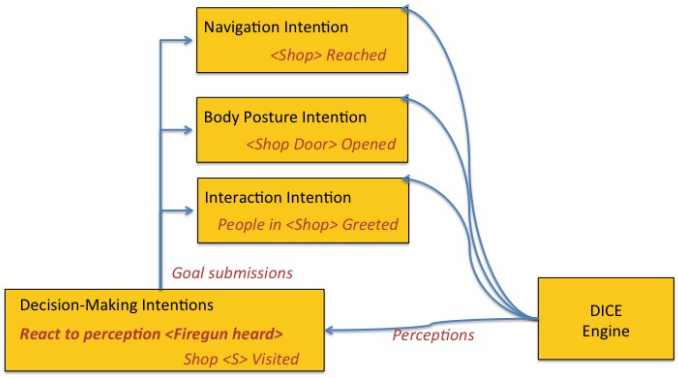


Fig. 3. Simplified DICE view of a shopper NPC during a game

the DICE approach on navigation is that, at any time, the travel direction (decided by a behaviour) can be changed and may be resumed later (e.g. when a reaction is completed); similarly, any body-controlling behaviour can be overridden and then resumed later. The APIs make programming this concurrent machinery a straightforward business, while the end-user development tool for behaviour modeling (called the DICE Parts Editor) provides an extremely powerful yet intuitive way to write scripts that affect one or more intention trees at each step [8].

IV. End-User Adaptation to Specific Scenarios

Currently, the programming of NPCs mostly relies on ad hoc specifications / implementations of their behaviors done by game developers. Thus, a specific behavior (e.g., a function emulating a panicking reaction) is hardwired to a specific item (e.g., the element "Caucasian_boy_17" in XVR) directly in the code. This generates a number of problems typical of ad hoc, low level solutions: the solution is scarcely reusable, it often depends on the specific knowledge of the code of a specific developer, and is cumbersome to modify, since every change required by the trainer has to be communicated to the developers and directly implemented in the code in a case by case manner. While this is not perceived as a major issue in entertainment games (but economics and a push for better game experiences are changing this, too), in serious gaming the cost and complexity of ad-hoc development is not covered by normal budgets. Typical solutions to this problem include, in multi-player games, the recruitment of experts to impersonate characters (such as team mates, enemies, victims, injured people, and so on) or, as in XVR, letting the trainer changing the scenario in real-time by hand.

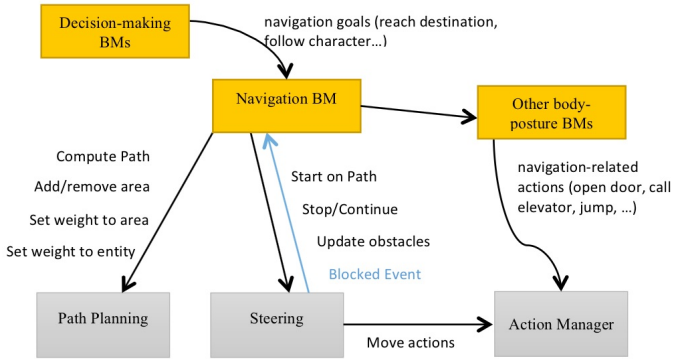


Fig. 4. Navigation subsystem architecture

PRESTO provides three main mechanisms that enable the reuse and adaptation of behavioural models to different scenarios, games or even game engines: semantic facilities, an interpreter of scripts in DICE, and facilities for game session control.

The semantization of the game environment and of part of the cognitive states of an NPC supports decision-making based on game- and scenario-independent properties. To this end, ontologies are used for the classification of objects and locations [4], for annotating them with properties and states (called "qualities") that allow abstract reasoning and for the (agent-specific) appraisals of perceptions, in particular to deal with potentially dangerous situations.

The current version of the PRESTO ontologies, targeted at its pilot project in a hospital domain, have been based on the upper level ontology DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [11] and the classification of elements provided by XVR. DOLCE was chosen as this ontology not only provides one of the most known upper level ontologies in literature but it is also built with a strong cognitive bias, as it takes into account the ontological categories that underlie natural language and human common sense. This cognitive perspective was considered appropriate for the description of an artificial world that needs to be plausible from a human perspective. The decision to use the classification of elements provided by XVR was due to the extensive range of item available in their libraries (approximatively one thousand elements describing mainly human characters, vehicles, road related elements, and artifacts like parts of buildings) and the popularity of XVR as virtual reality platform for emergency

management and training.

The construction of the PRESTO ontologies was performed by following a middle-out approach, which combined the reuse and adaptation of the conceptual characterization of top-level entities provided by DOLCE and the description of extremely concrete entities provided by the XVR environment. More in detail,

- we performed an analysis and review of the conceptual entities contained in DOLCE-lite [11] together with virtual reality experts (both trainers and developers) and selected the ones referring to concepts than needed to be described in a scenario; this analysis has originated the top part of the PRESTO ontology;
- we performed a similar analysis and review of the XVR items, together with their classifications, in order to select general concepts (e.g, vehicle, building, and so on) that refer to general game scenarios; this analysis has originated the middle part of the PRESTO ontology;
- as a third step we have injected (mapped) the specific XVR items into the ontology, thus linking the domain independent, virtual reality platform independent ontology to the specific libraries of a specific platform.

The flow of perceptions, the properties of entities, the appraisal values of DICE behavioural models are classified by means of the PRESTO ontologies, thus enabling the development of generic BDI logic (goals, plans and beliefs) independent of the scenario of use. Additionally, DICE provides an interpreter for high-level scripts, called "DICE Parts", written by means of a graphical editor by the end-user (typically a trainer during the preparation of a specific scenario). A DICE Part can invoke multiple goals concurrently on the various DICE-managed intention trees, terminate them when specific events happen (including timeouts and perceptions), define reactions to perceptions or to modifications of the internal state of the agent (including appraisals and moderators), change the state of the agents itself, and so on. While the DICE Part language is limited in its expressivity, the cost of producing a part is minuscule with respect to directly programming the underlying BDI logic; further, DICE Parts can be distributed as decision-making behavioural models on their own. Thus, an effort is required on developers of behavioural models in BDI logic to provide goal-directed behaviours that are suitable for composition within user-written parts and adaptable to different scenarios thanks to semantic-based reasoning; the PRESTO pilot project and other demonstrators are helping in accumulating experience that will feed future guidelines.

Finally, PRESTO has an end-user facility to edit and control session-level scripts inspired by interactive books. A session script is composed by a set of scenes connected as a graph. At each scene, goals can be given to NPCs (which may trigger user-written parts), their internal state changed (including emotions) and objects manipulated. The trainer starts a script at the beginning of a training session and advances it by manually navigating the graph of scenes or letting PRESTO choose the next one e.g. when certain events happen or when a timer expires. This allows a large, potentially unlimited number of different sessions to unfold from a single script with no need to reprogram the NPCs once equipped with all required behavioural models and DICE Parts. In the hospital ward example presented earlier, the initial scene would command visitors, patients and nurses to accomplish their routine goals; the script may continue with alternative scenes such as "fire breaking in a patient room" or "fire breaking in a surgical facility", each with different people involved, and then with sequences that may lead e.g. to smoke filling the area and visitors fleeing or an orderly managed situation with the intervention of fire fighters, chosen according to the decisions of the trainer and the events occurring during a session.

V. Cognitive navigation in PRESTO

As extensively discussed in [3], at the time of writing the most developed models concern the navigation in virtual environments. As it can be seen in picture in Fig. 4, navigation is split in two levels: the lower level facilities look after path planning and steering within the decided path and are implemented within the PRESTO infrastructure, close to the game engine; the higher level is concerned with control and is implemented as behavioural models for DICE. In turn, navigation control models are of two types. One type, identified as "navigation BM" in Fig. 1 and 4, satisfies the navigation goals submitted by decision-making behaviours (e.g., of reaching a destination); slightly different navigation models are provided that depend on the main physical features of the NPC, e.g. of being a human rather than a vehicle, and consequently on the NPC's ability to move and affect the environment. As mentioned above, the navigation BM runs in its own intention tree (thread of execution) concurrently with decision-making and other body-controlling behaviours. The navigation BM calls path planning and controls steering, acting according to the latter's indication in particular when it blocks because there are obstacles or there is a closed gate. A number of different decisions can be taken according to the model and to the semantics of gates or obstructing objects, which may in turn cause goals to be submitted to other body-parts behaviours (e.g. opening a door, calling a lift, and so on).

A second type of behavioural model, referred to as "navigation capabilities" and included as a decision-making module in DICE, looks after some of the cognitive aspects of navigation. In particular, the navigation capability of an NPC decides which navigation mesh (i.e. navigable surface data) to use on creation, then changes the default speed, default animations and so on according to the current sub-rational state of the agent (i.e. its moderators and cognitive parameters). Thus, PRESTO can provide capabilities specialized e.g. for quiet or excited people, for permanent or temporary physical impairments, for different types of vehicles, and so on. Navigation capabilities may access the cognitive state to tune their parameters (e.g. speed or animations); furthermore, behavioural rules may be defined to switch navigation capabilities entirely during a game depending on the NPC's moderators. For instance, a high level of fear may select a model whose default speed is running and movement animations are jerky, while a high level of fatigue may select a model doing exactly the opposite. Furthermore, the navigation capabilities satisfy goals concerning path selection, such as "stay out of sight of entity E" or "don't go through location L" (which may have been classified as dangerous by a decision-making model according to the appraisal rules of the agent), by taking note of what to

avoid and manipulating the navigation data accordingly, based on current knowledge and the flow of perceptions.

Behavioural models in DICE have their own configuration parameters, called "background knowledge". As mentioned above, the background knowledge of the navigation capability of an agent determines how much the agent knows *a priori* about the environment – it can be everything or being limited to a few areas; the navigation data is created accordingly. The flow of perceptions arriving from the PRESTO infrastructure includes also the visible polygons of the various navigation meshes; this data is used by the navigation capability to update the navigation data. The cognitive model of DICE, not discussed here, looks after short-term memory management, which includes calling the navigation capability to purge its data; that is, the agent literally forgets about where to navigate according to timing and frequency of perceptions from the environment. Out of scope of the navigation subsystem, and not discussed here, is a "search" behaviour, which is a set of decision-making procedures that can be started when a navigation goal fails with an "unknown path" error.

In the hospital fire scenario presented in the introduction, the navigation capability of a patient on a wheel chair would use a different mesh than the one selected for a visitor with normal walking capabilities, e.g. to avoid steps and stairs. The patient's background knowledge would include the navigation areas of the entire ward (since she has been there for a while) while the visitor's knowledge would be initially empty and populated while she moves in the ward; a decision-making procedure of the visitor that invokes a goal such as "go to patient room nr. 3" would initially fail because, indeed, no path can be computed and a search behaviour would need to be invoked allowing the progressive discovery of the navigation areas of the selected mesh. If, at any time during the game, a fire alarm starts ringing, its perception on both visitor and patient would trigger a (decision-making) reaction that is handled differently according to the currently active behavioural models, which in turn may depend on cognitive states such as fear. The perception of smoke and fire would submit goals such as "don't go through that area" handled by the navigation capability as mentioned above. A rationally-behaving NPC that knows the position of a location ontologically classified as "fire exit" would navigate to the latter, with a speed and a modality that depend on the currently active navigation capability (excited / not excited, walking / pushing the wheel chair); an NPC that doesn't know about fire exits or that it's too fearful to act rationally would run to the closest exit.

VI. Multi-agent coordinated behaviour via semantic tagging

Work is in progress on game-theoretical descriptions of coordinated behaviour, which include queuing and other crowding behaviours, access to shared resources, and so on that allows the definition of policies at a very abstract (meta-) level. This exploits the support in DICE for introspection, semantic tagging of goals and plans, dynamic assignment and aborting of goals and intentions as well as the ability to dynamically manipulate semantic tags (called ''qualities") of any entities including NPCs offered by PRESTO. The specification of policies is expected to substantially reduce the coding required by models and allows the reuse of the same coordination patterns in many different situations, e.g. for queuing to pass through a gate (which will be part of the navigation BMs) as well as for queuing at the entrance of an office or at the cashier in a supermarket (which are decision-making behaviours not related to navigation goals).

A simplistic (but already available and of great practical use) coordinated behaviour exploiting qualities is goal delegation from an agent to another agent. By means of the PRESTO API, any entity in a game can submit a goal to be pursued by any other entity; when the goal is enriched with a few predefined parameters, the destination DICE agent publishes the fact that it has accepted a goal or that has achieved it (or failed to achieve or refused), allowing the submitter (or any other observer, including the session script engine) to monitor and coordinate behaviours without the use of any additional agent protocol.

VII. Current state of developments and experimentation

At the time of writing (May 2015), most of the DICE framework is in place. Work is in progress on the automatic publishing of qualities concerning intentions, required by the meta-level policies described above among others. Its end-user development editors are being evaluated within a laboratory run in collaboration with the University of Trento. To this end, Delta Informatica has developed a serious game with Unity that allows the definition of the behaviour of a multiplicity of characters coordinating to run a business with walk-in visitors, emergencies forced by the human player, and so on. This game is being used as workbench for the students participating to the laboratory and for the testing and validation of the current and future developments for the entire PRESTO project.

The ontologies concerning the hospital fire scenario, used also by the session-control tool described in Sec. IV, are being validated in a pilot project in collaboration with the main Trento hospital. The purpose of this pilot is the introduction of virtual reality training for the management of fires in wards. Given the level of novelty with respect to traditional training support tools (oral presentations, questionnaires and live simulations), at this stage the pilot is still focusing on having the trainers understanding the potentiality of virtual reality by creating courses for teaching basic emergency procedures by means of XVR by E-semble. A few NPCs have been already tested within Delta Informatica's lab; their introduction in more complex training scenarios with the need of managing evacuations, coordinating personnel, etc. are expected to be experimented in the next 12 months.

VIII. Conclusions and future works

The DICE framework and the underlying PRESTO infrastructure are a practical and powerful way to introduce agent-oriented programming and semantics into games. Benefits against traditional approaches, such as finite state machines, behavioural trees and so on, include the ability to represent non-rational factors such as emotions and the support of powerful session scripting that make a PRESTO-enabled game similar to a sort of theatrical improvisation. We presented examples from the serious gaming world, but nothing prevents using PRESTO in entertainment games or for simulation without human-in-the-loop.

Work in in progress on many aspects, including multi-agent coordination as presented above, libraries of reusable behavioural models, improvements to the various engines and to the end-user tools, including facilities for community sharing of models, parts and scenario scripts.

ACKNOWLEDGMENT

In addition to the members of the implementation team in Delta Informatica (Matteo Pedrotti, Mauro Fruet, Paolo Calanca, Michele Lunelli), we thank all PRESTO research partners and in particular Chiara Ghidini (FBK), Zeno Menestrina ed Antonella De Angeli (University of Trento). PRESTO is funded by the Provincia Autonoma di Trento (PAT).

REFERENCES

[1] Michael E. Bratman, *Intention, Plans, and Practical Reason*, Harvard University Press, November 1987.

[2] Paolo Busetta, Chiara Ghidini, Matteo Pedrotti, Antonella De Angeli, and Zeno Menestrina, 'Briefing virtual actors: a first report on the presto project', in *Proceedings of the AI and Games Symposium at AISB 2014*, ed. Daniela Romano, (April 2014).

[3] Paolo Calanca, and Paolo Busetta, 'Cognitive Navigation in PRESTO', in *Proceedings of the AI and Games Symposium at AISB 2015*, to appear, (April 2015).

[4] Mauro Dragoni, Chiara Ghidini, Paolo Busetta, Mauro Fruet, and Matteo Pedrotti, 'Using ontologies for modeling virtual reality scenarios', in *to appear in Proceedings of ESWC 2015*.

[5] Paolo Busetta, Nicholas Howden, Ralph Rönnquist, and Andrew Hodgson, 'Structuring BDI Agents In Functional Clusters', in *Proceedings of the Workshop on Agent Theories Architectures and Languages (ATAL-99)*, volume 1757 of *Lecture Notes in Artificial Intelligence*, Orlando, Florida, (15-17th July 1999). Springer Verlag.

[6] Rick Evertsz, Matteo Pedrotti, Paolo Busetta, Hasan Acar, and Frank Ritter, 'Populating VBS2 with Realistic Virtual Actors', in *Conference on Behavior Representation in Modeling & Simulation (BRIMS)*, Sundance Resort, Utah, (March 30 – April 2 2009).

[7] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf, 'End-User Development: An Emerging Paradigm', *End User Development*, **9**, 1–8, (2006).

[8] Zeno Menestrina, Antonella De Angeli, and Paolo Busetta, 'APE: end user development for emergency management training', in *6th International Conference on Games and Virtual Worlds for Serious Applications, VS-GAMES 2014, Valletta, Malta, September 9-12, 2014*, pp. 1–4. IEEE, (2014).

[9] Anand S. Rao and Michael P. Georgeff, 'Bdi agents: From theory to practice', in *IN PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON MULTI-AGENT SYSTEMS (ICMAS-95*, pp. 312–319, (1995).

[10] Frank E. Ritter, Jennifer L. Bittner, Sue E. Kase, Rick Evertsz, Matteo Pedrotti, and Paolo Busetta, 'CoJACK: A high-level cognitive architecture with demonstrations of moderators, variability, and implications for situation awareness', *Biologically Inspired Cognitive Architectures*, **1**, 2–13, (July 2012).

[11] Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L.: Sweetening ontologies with dolce. In: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, Springer-Verlag (2002) 166–181

Agent based Modeling and Simulation with ActoMoS

Agostino Poggi
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma
Parma, Italy
agostino.poggi@unipr.it

***Abstract* — ActoMoS is an actor-based software library for the development of agent-based models and for their simulation. This library offers software components and tools for modeling and simulating systems in different application domains. In particular, it allows the definition of agent model by reusing or extending a set of predefined agent models and supports efficient and scalable agent-based simulations involving a large number of agents. This paper, after an introduction of the actor model and implementation used by the software library, underlines the main features of the software library and presents its experimentation in some well-known domains.**

Keywords – Agent Based Modeling and Simulation, Actor model, Java.

I. INTRODUCTION

Simulation models are increasingly being used for solving problems and for helping in decision-making. The size and complexity of systems that are usually modeled are ever increasing. Modeling and simulation of such systems is challenging because it requires suitable and efficient modelling and simulation tools that take advantage of the power of current computing architectures, programming languages and software frameworks, and that make easy the development of applications.

Agent-based modeling and simulation (ABMS) tools and techniques seem be the most suitable means to cope with such challenges [19], [30]. In fact, ABMS has been and is widely used with success for studying complex and emergent phenomena in many research and application areas, including agriculture, biomedical analysis, ecology, engineering, sociology, market analysis, artificial life, social studies, and others fields. However, the limit of such tools and libraries is that their agent models shown a very limited use of the features offered by the computational agents found in Multi-Agent Systems (MAS) or Distributed Artificial Intelligence (DAI) techniques [12]. Therefore, it may be difficult to model some kinds of problem that, for example, require complex interaction among agents, and is usually less natural to distribute a simulation on a network of computational nodes.

This paper presents an actor based software library, ActoMoS, (Actor Modeling and Simulation) providing a set of suitable software components for the development of ABMS applications, the visualization of the simulations and the analysis of their results. The next section provides an overview of the software framework used for the implementation of the software library. Section 3 describes the features of the software library and shows how it makes easy the developing of agent based models and simulations. Section 4 shows its experimentation in some well-known ABMS application domains. Section 5 introduces related work. Finally, section 6 concludes the paper by discussing its main features and the directions for future work.

II. CODE SOFTWARE FRAMEWORK

CoDE (Concurrent Development Environment) is an actor-based software framework aimed at both simplifying the development of large and distributed complex systems and guarantying an efficient execution of applications [27]. CoDE is implemented by using the Java language and takes advantage of preexistent Java software libraries and solutions for supporting concurrency and distribution. CoDE has a layered architecture composed of an application and a runtime layer. The application layer provides the software components that an application developer needs to extend or directly use for implementing the specific actors of an application. The runtime layer provides the software components that implement the CoDE middleware infrastructures to support the development of standalone and distributed applications.

In CoDE a system is based on a set of interacting actors that perform tasks concurrently. An actor is an autonomous concurrent object, which interacts with other actors by exchanging asynchronous messages. Communication between actors is buffered: incoming messages are stored in a mailbox until the actor is ready to process them. After its creation, an actor can change several times its behavior until it kills itself. Each behavior has the main duty of processing incoming messages through some handlers called cases. Each case can process only the messages that match a specific message pattern represented by an object that can apply a combination of constraints on the value of all the fields of a message. Therefore, if an unexpected message arrives, then the actor mailbox maintains it until a next behavior will be able to process it.

An actor can perform different types of action. It can send messages to other actors or to itself, create new actors, update its local state, set a timeout for waiting for the next message, change its behavior and kill itself. An actor can be viewed as a logical thread that implements an event loop [11], [21]. This event loop perpetually processes events representing the reception of messages, the exchange of behavior and the firing

of timeouts. In response on a reception of a message or the firing of a timeout, the actor finds and executes the suitable case for the processing of such an event. When the event represents the change of the behavior, the actor moves to the new behavior. In particular, the API of an actor does not offer any action for managing the reception of messages and for monitoring the firing of timeouts. In fact, an application developer uses an actor implementation provided by the CoDE runtime and needs only to provide the behaviors of the different actors of the application by defining the methods for their initialization and the message pattern – process method pairs of their cases.

Depending on the complexity of the application and on the availability of computing and communication resources, one or more actor spaces can manage the actors of the application and so an application can be distributed on a network of computational nodes. An actor space acts as "container" for a set of actors and provides the services necessary for their execution. In particular, an actor space takes advantages of two special actors: the scheduler and the service provider. The scheduler manages the concurrent execution of the actors of the actor space. The service provider enables the actors of an application to perform new kinds of action. The current implementation of the software framework provides services for supporting the broadcast of messages, the exchange of messages through the "publish and subscribe" pattern, the mapping of actors address to symbolic names, the mobility of actors, the interaction with users through emails and the creation of actors. The last service is important because an actor cannot directly create actors in other actor spaces, but can delegate it to their service providers.

One of the most important features of CoDE is the possibility of configuring an application with different implementations of the runtime components. It allows the use of different actor implementations, different schedulers and service providers. The type of the implementation of an actor is one of the factors that mainly influence the attributes of the execution of an application. In particular, actor implementation can be divided in two classes that allow to an actor either to have its own thread (from here named active actors) or to share a single thread with the other actors of the actor space (from here named passive actors). Moreover, the duties of a scheduler depend on the type of the actor implementation. Of course, a scheduler for passive actors is different from a scheduler for active actors, but for the same kind of actor can be useful to have different scheduler implementations. For example, it can allow the implementation of "cooperative" schedulers in which actors can cyclically perform tasks varying from the processing of the first message in the buffer to the processing of all the messages in it.

The most important decision that influence the quality of the execution of an application is the choice of the actor and scheduler implementations. In fact, the use of one or another couple of actor and scheduler causes large differences in the performance and in the scalability of the applications [6].

CoDE provides three types of actor implementation and four types of scheduler. The first two types of actor implementation represent active and passive actors. The third type of implementation represents special passive actors, called shared actors, which get messages from a shared queue. The first two types of scheduler implementation drive the execution of either active or passive actors (active and passive schedulers). The third type of implementation is used for shared actors (shared schedulers). Finally, the forth type of implementation is used in actor spaces containing both active and passive actors (hybrid schedulers).

The identification of the best couple of actor and scheduler implementations for a specific application mainly depends on the number of actors, the number of exchanged messages, the preeminent type of communication used by actors (i.e., point-to-point or broadcast) and the possible presence of a subset of actors that consume a large part of the computational resources of the application. Table 1 shows what should be the best choices for a qualitative partition of the values of the previous parameters. In particular, the third column indicates the preeminence of either point-to-point communication (P) or broadcast communication (B), the forth column indicates the presence/absence of a subset of heavy actors and the word "any" is used when the value of the associate parameter has not effect on the choice of actor and scheduler implementations.

TABLE 1

actors	**messages**	**P/B**	**Heavy**	**scheduler**
few	any	any	any	active
many	any	P	no	passive
many	few	B	no	passive
many	many	B	no	shared
many	any	any	yes	hybrid

Finally, an actor space can enable the execution of an additional runtime component called logger. The logger has the possibility to store (or to send to another application), in a textual or binary format, the relevant information about the execution of the actors of the actor space (e.g., creation and deletion of actors, exchange of messages, processing of messages and timeouts, exchange of behaviors and failures). Therefore, users and other applications can use such information for understanding the activities of an application, for diagnosing the causes of execution problems, and for solving them.

III. ACTOMOS

The features of the actor model and the flexibility of its implementation make CoDE suitable for building ABMS applications [28]. In particular, actors have the suitable features for defining agent models that can be used in ABMS applications and to model the computational agents found in MAS) and DAI systems. In fact, actors and computational agents share certain characteristics: i) both react to external stimuli (i.e., they are reactive), ii) both are self-contained, self-regulating, and self-directed, (i.e., they are autonomous), and iii) both interact through asynchronous messages and such messages are the basis for their coordination and cooperation (i.e., they are social). Moreover, given that actors interact only

through messages and there is not a shared state among them, it is not necessary to maintain an additional copy of the environment to guarantee that agents decide their actions with the same information (thing that is usually necessary in some application domain with other ABMS platforms). Finally, the use of messages for exchanging state information decouples the code of agents. In fact, agents do not need to access directly to the code of the other agents to get information about them, and so the modification of the code of a type of agent should cause lesser modifications in the code of the other types of agent. Finally, the use of actors simplifies the development of real computational agents in domain where, for example, they need to coordinate themselves or cooperate through direct interactions.

Moreover, the use of CoDE simplify the development of flexible and scalable ABMS applications. In fact, the use of active and passive actors allows the development of applications involving large number of actors, and the availability of different schedulers and the possibility of their specialization allow a correct and efficient scheduling of the agents in application domains that require different scheduling algorithms [20]. Moreover, the efficient implementation of broadcasting and multicast removes the overhead given to the need that agents must often diffuse the information about their state to the other agents of the application (e.g., their location in a spatial domain).

However, CoDE does not offer specific components for ABMS (e.g., simulators, agent models and simulation viewers). Therefore, we defined a software library, called ActoMoS (Actor Modelling and Simulation), that, starting from CoDE, provides a set of software components and tools for making easy the development of ABMS applications.

In large part of ABMS platforms usually a simulation is given by a sequence of steps where each agent needs only to get information about its surround (i.e., about a subset of the other agents and about the environment) and then to use such information for deciding its actions. In ActoMoS the simulation is similar, but agents get information about agents and the environment through messages.

In ActoMoS, to simplify the interaction between agents and the environment, the relevant parts of an environment are represented by a set of actors whose goals are to inform the agents acting in the environment about their presence and their state, and to update their state when the agents act on them. Given that the behavior of such actors is similar to the one expressed by the agents acting in the environment, we call both agents, but we divided them in active and passive agents. Active agents are the typical agents of an ABMS, i.e., they represent the entities able to move and cooperate with other entities acting in the environment. Passive agents define the environment of an ABMS, i.e., they represent the relevant elements of the environment (e.g., in a spatial domain the obstacles and the reference points for the movement of the active agents).

Such agents are usually implemented taking advantage of the shared actor implementation provided by CoDE, but it is necessary to develop a specific scheduler. Such a scheduler executes repeatedly all the agents and after each execution step broadcasts them a "clock" message. This last message allows to the agents to understand that they have all the information for deciding their actions, therefore, they decide, perform some actions and, at the end, broadcast the information about their new state.

In ActoMos, all the agents are usually represented by one or more actor behaviors that process the input messages through two cases. The first case processes the messages informing an agent about the state of the other agents. The second case processes the "clock" messages. However, while active agents exchange messages and perform other types of action (e.g., in a spatial domain to change their location), often, passive agents have the only duty of sending messages for informing the active agents about their presence (e.g., immutable obstacles or path points in a spatial domain). Therefore, such passive agents are represented by an actor behavior providing a case that get the "clock" messages for deciding when sending the information about their presence and state.

Of course, different types of agent have different implementations of the cases of their behaviors. In particular, ActoMoS provides some abstract behavior implementations for developing applications in different domains. Such implementations define the state information that an agent need to maintain in its specific application domain and provides a set of abstract methods for processing incoming information and for performing the actions in response to the "clock" messages.

Often the modelling of some systems (e.g., social networks) requires a massive number of agents. However, in such kind of systems, usually only a part of them is simultaneously active and the actions of the different agents do not need a synchronization. Therefore, it is necessary a scheduler that can manage a massive number of agents, but that can try to optimize the execution by scheduling only the active agents. The solution we implemented derives from the virtual memory techniques used by operating systems: agents increment an inactivity counter in the scheduling cycles in which they do not process messages and reset it in the cycles in which they process a message. The scheduler can get the value of such counters and can move an actor in a persistent store when its inactivity counter becomes greater than a fixed (or dynamic) threshold. The scheduler reload an actor from the persistent store when it receives a new message from another agent.

Of course, the number of active agents can vary over the simulation, but the quality of the simulation can be guaranteed if the number of the agents, maintained by the scheduler, remains in a range that depends on the available computational resources. The adopted solution, to limit to the number of active actors and to guarantee good performances, is to provide a scheduler able to move an inactive agent in the persistent storage on the basis of a variable number of inactive cycles. In particular, this number becomes high when the number of scheduled agents decreases (i.e., the scheduler does not spend time for storing agents in the persistence storage and reloading them) and becomes more and more low with the increasing of the number of scheduled agents.

Two important features that an ABMS framework should provide are the availability of graphical tools for the

visualization of the evolution of simulations and the possibility of analyzing the data obtained from simulations. CoDE does not provide any specific tool for ABMS, but provides a logging service that allows the recording of the Java objects describing the relevant actions of an actor (i.e., its initialization, reception, sending and processing of messages, creation of actors, change of behavior, and its shutdown). Therefore, we developed two graphical tools, that use such logging data for visualizing the evolution of simulations in spatial domains based on continuous and discrete 2D space representations, and another tool that use them for extracting statistical information about simulations. Fig. 9 shows two views of the GUI that supports 2D spatial simulations. In particular, it presents the initial and final views of the evacuation of a large number of pedestrians from a building.

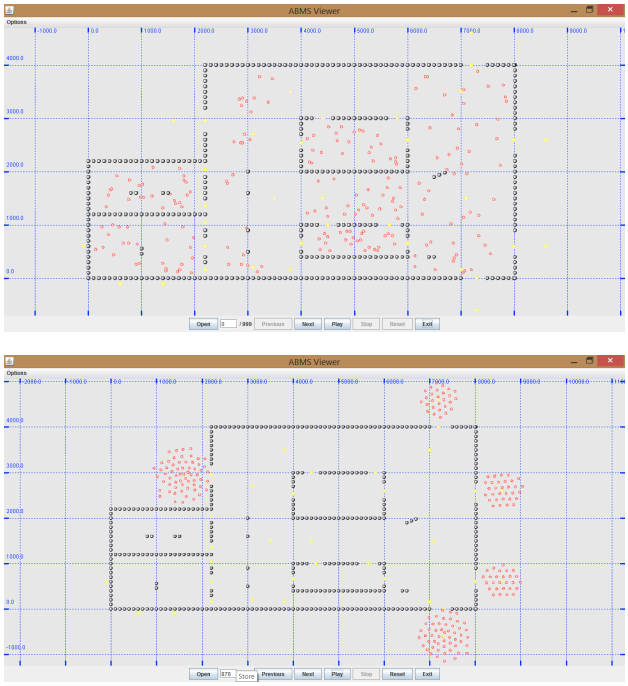


Fig. 9. Initial and final view of the simulation of a crowd evacuation.

IV. EXPERIMENTATION

We are using ActoMoS for the simulation of two well-known problems in a spatial environment: the prey-predator pursuit problem [2], and the crowd evacuation problem [33]. It is possible because ActoMoS offers all the software components necessary for modelling agents in a continuous or discrete 2D (or 3D) space (e.g., implementation of the algorithms that drive the movement of agents, agent models representing obstacles and path points).

The first experimentation of the prey-predator pursuit problem had be done in a 2D discrete space. Its main result is the definition of a flexible agent model that allows the execution of simulations with different algorithms, which drive the movement of the prey and of the predators, by simply changing the values of some configuration properties. Moreover, this agent model allows the solution of the conflicts among the moves of agents (i.e., two or more agents cannot share the same cell) with different coordination algorithms. The only constraint for using such coordination algorithms is that each agent needs to perform the current move, compute its next move and inform the other agents about it at each cycle. Therefore, each agent knows the intentions of the other agents before performing its move and performs the move only if the rules of the coordination algorithm allow it. This constraint does not cause a different behavior of the prey and the predators respect to the implementations of other ABMS platforms, because even in this implementation an agent can only either perform the previously decided move or remain in the same cell.

The first experimentation of the crowd evacuation problem had be done in a 2D continuous space and by implementing the agents of the crowd by using the boid model [31]. This experimentation take advantage of an extended set of boid rules that allow to agents to reach a meeting point outside the building by following either other agents or a set of alternative paths. Even in this case, the main result of the experimentation is the definition of a flexible agent model. Such a model allows the definition of different types of simulation by using different sets of boid rules. In particular, the experimentation shows how the possibility to follow other agents and the presence of paths towards the exit points make possible successful evacuations. Moreover, the use of the boid rules shows how is possible to obtain intelligent behaviors without using complex AI algorithms. However, the "calibration" of the model requires in some cases a large number of simulations to obtain a successful evacuation where agents both do not collide among them or with obstacles and do not lost time inside the building. In fact, the movement of each agent of the crowd is defined by the boid rules and so it is necessary to find the correct weights with which such rules contribute to the movement of the agent.

We are working for some time in the analysis and simulation of social networks [3], [4]. In particular, currently we are using ActoMoS for designing a peer-to-peer social network that can guarantee the same services of the centralized ones. The problem of peer-to-peer social network is that they do not have a centralized service that maintain the information shared among the users and so, for example, is difficult for a user that wakes up after a period of inactivity to get all the information of her/his interest that has been published when she/he was offline.

In particular, we defined a model of a peer-to-peer social network where a user can move from the online and offline state, publish new information and subscribe to new types of information. Each user is defined by a simple agent, which performs her/his actions using non-deterministic rules. Moreover, such an agent has also the duty to cooperate with the other agents to avoid that offline users lost information of their interest. Of course, the actors representing the online users should perform such a task. Moreover, its implementation should have the goals of limiting the amount of the recovered information (i.e., only the information of interest for the offline users should be recovered) and of guaranteeing privacy (i.e., users should not have additional information about the other users through the execution of such a task).

The experimentation is in the initial phase and we obtain good results with a simple algorithm of election that allows to the agents, representing the users that are moving offline, to assign the recording of the information of their interest, to agents of the users that remain online. The problems we need to solve is that in some situations there are few or no online users. In fact, when there are few users then their agents are overloaded in the recording of the information for the other users. When there are not users, when a user becomes online again it cannot have the lost information and cannot act as recorder for the other users. The solution that we are studying to solve such two problems is based on the introduction of "auxiliary agents", i.e., agents that: i) do not represent a user, ii) are running on computational nodes that are usually operative, and iii) have only the task of recording the information for the offline users.

V. Related Work

A lot of work has been done in the field of agent-based modeling and simulation. Moreover, some researchers used the actor model for the modeling and simulation of complex systems. The rest of the section presents some of the most interesting works presented in the previous two fields.

Swarm [22] is the ancestor of many of the current ABMS platforms. The basic architecture of Swarm is the simulation of collections of concurrently interacting agents, and this paradigm is extended into the coding, including agent inspector actions as part of the set of agents. So in order to inspect one agent on the display, you must use another hidden, non-interacting agent. Swarm is a stable platform, and seems particularly suited to hierarchical models. Moreover, it supports good mechanisms for structure formation using multi-level feedback between agents, groups of agents, and the environment (all treated as agents).

Ascape [26] is a framework for developing and analyzing agent based models following some of the ideas of Swarm. However, it is somewhat easier to develop models with Ascape than with Swarm. Indeed, its goal is to allow people with only a little programming experience to develop quite complex simulations by providing a range of end user tools. Ascape is implemented in Java and users would require some ability to program in Java together with understanding of the object orientation philosophy.

NetLogo [32] is an ABMS platform based on the Logo programming language. Its initial goal was to provide a high-level platform allowing students, down to the elementary level, to build and learn from simple ABMS applications. Now it offers many sophisticated capabilities and tools that make it suitable for complex applications too. Moreover, a big advantage respect to the other platforms is the simplicity of its own language.

Repast [25] is a well-established ABMS platform with many advanced features. It started as a Java implementation of the Swarm toolkit, but rapidly expanded to provide a very full featured toolkit for ABMS. Although full use of the toolkit requires Java programming skills, the facilities of the last implementations allow the development of simple models with little programming experience [24].

MASON [16] is a Java ABMS tool designed to be flexible enough to be used for a wide range of simulations, but with a special emphasis on "swarm" simulations of a very many (up to millions of) agents. MASON is based on a fast, orthogonal, software library to which an experienced Java programmer can easily add features for developing and simulating models in specific domains.

ATC [7] is a framework for the modeling and validation of real-time concurrent systems based on the actor model. In particular, it inherits all the functional capabilities of actors and further allows the expression of most of the temporal constraints pertaining to real-time systems: exceptions, delays and emergencies.

The Adaptive Actor Architecture [17] is an actor-based software infrastructure designed to support the construction of large-scale multi-agent applications by exploiting distributed computing techniques for efficiently distribute agents across a distributed network of computers. This software infrastructure uses several optimizing techniques to address three fundamental problems related to agent communication between nodes: agent distribution, service agent discovery and message passing for mobile agents.

An actor-based infrastructure for distributing Repast models is proposed in [9]. This solution allows, with minimal changes, to address very large and reconfigurable models whose computational needs (in space and time) can be difficult to satisfy on a single machine. Novel in the approach is an exploitation of a lean actor infrastructure implemented in Java. In particular, actors bring to RePast agents migration, location-transparent naming, efficient communication, and a control-centric framework.

Statechart actors [10] are an implementation of the actor computational model that can be used for building a multi-agent architecture suitable for the distributed simulation of discrete event systems whose entities have a complex dynamic behavior. Complexity is dealt with by specifying the behavior of actors through "distilled" statecharts [14]. Distribution is supported by the theatre architecture [8]. This architecture allows the decomposition of a large system into sub-systems (theatres) each hosting a collection of application actors, allocated for execution on to a physical processor.

Simulator X [18] is a software research platform for intelligent interactive simulation that takes advantage of the actor model for supporting fine-grained concurrency and parallelism. The architecture uses actors to obtain a distributed application state and execution model. Simulator X is mainly used in the areas of real-time interactive systems, virtual reality and multimodal interaction.

VI. Conclusions

This paper presented a software library, called ActoMoS, which makes easy the development of agent-based models and supports efficient agent-based simulations involving a large number of agents. ActoMoS has been implemented on the top of CoDE (Concurrent Development Environment) that is an actor-based software framework aimed at both simplifying the

development of large and distributed complex systems and guarantying an efficient execution of applications [27].

ActoMoS has been experimented with success in the development of ABMS applications. Of course, its current implementation does not provide all the features of the most known ABMS platforms (i.e., NetLogo [32], Repast [25] and MASON [16]). However, the use of the actor model for the definition of agents allows to define real agent models where agent interact through the exchange of messages avoiding the use of a shared state and it simplifies the development of non-trivial applications where the management of concurrent activities may be of primary importance. Moreover, the availability of techniques to reduce the overhead of the diffusion of broadcast and multicast messages generally allows the development of applications whose performances are comparable with the ones provide by applications implemented by platforms that do not use messages for diffusing the state of the environment and of the agents of applications.

Current work has the goals of extending the functionalities of the software library and of continuing its current experimentation. Moreover, future work will be dedicated to the modeling and simulation of systems for e-business services [23], collaborative work services [13] and for the management of information in pervasive environment [5][29].

REFERENCES

[1] G.A. Agha, "Actors: A Model of Concurrent Computation in Distributed Systems," Cambridge, MA, USA: MIT Press, 1986.

[2] M. Benda, V. Jagannathan, and R. Dodhiawalla, "On optimal cooperation of knowledge sources," Tech. Rep. BCS-G2010-28, Boeing AI Center, Bellevue, WA, USA, 1986.

[3] F. Bergenti, E. Franchi, and A. Poggi,. "Selected models for agent-based simulation of social networks," in Proc. of 3rd Symp. on Social Networks and Multiagent Systems (SNAMAS 2011), York, UK. 2011, pp. 27-32.

[4] F. Bergenti, E. Franchi and A. Poggi, "Agent-based interpretations of classic network models," Computational and Mathematical Organization Theory, Vol. 19, No. 2, 2013, pp. 105-127, 2013.

[5] F. Bergenti, and A. Poggi, "Ubiquitous Information Agents," International Journal on Cooperative Information Systems, Vol. 11, No. 3-4, pp. 231-244, 2002.

[6] F. Bergenti, A. Poggi, and M. Tomaiuolo, "An Actor Based Software Framework for Scalable Applications," in Internet and Distributed Computing Systems, Berlin, Germany: Springer, 2014, pp. 26-35.

[7] L. Boualem, and S. Yamina, "On equivalences for actors expressions and configurations of ATC," WSEAS Transactions on Computers vol. 4, no. 9, pp. 1045-1053, 2005.

[8] F. Cicirelli, A. Furfaro, and L. Nigro, "Exploiting agents for modelling and simulation of coverage control protocols in large sensor networks," Journal of Systems and Software, vol. 80, no. 11, pp. 1817-1832, 2007.

[9] F. Cicirelli, A. Furfaro, A. Giordano and L. Nigro, "Distributing Repast simulations using actors," in Proc. of 23rd European Conf. on Modelling and Simulation, Madrid, Spain, 2009, pp. 226–231.

[10] F. Cicirelli, A. Furfaro, and L. Nigro, "Modeling and simulation of complex manufacturing systems using Statechart-based actors," Simulation Modelling Practice and Theory, vol. 19, no. 2, pp. 685–703, 2011.

[11] J. Dedecker, T. Van Cutsem, S. Mostinckx, T. D'Hondt and W. De Meuter, "Ambient-oriented programming in ambienttalk," in ECOOP 2006 – Object-Oriented Programming, Berlin, Germany: Springer, 2006, pp. 230-254.

[12] A. Drogoul, D. Vanbergue, and T. Meurisse, "Multi-agent based simulation: Where are the agents?," In Multi-agent-based simulation II, Berlin, Germany: Springer, 2003, pp. 1-15.

[13] E. Franchi, A. Poggi, and M. Tomaiuolo. "Open social networking for online collaboration," International Journal of e-Collaboration, vol. 9, no. 3, pp. 50-68, 2013.

[14] D. Harel, and A. Naamad, "The Statemate semantics of Statecharts," ACM Transactions on Software Engineering and Methodology, vol. 5, no. 4, pp. 293–333, 1996.

[15] C. E. Hewitt, "Viewing controll structures as patterns of passing messages," Artificial Intelligence, vol. 8, no. 3, pp. 323–364, 1977.

[16] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan, "MASON: A multiagent simulation environment," Simulation, vol. 81, no. 7, pp. 517-527, 2005.

[17] M. Jang, and G.A. Agha, "Scalable agent distribution mechanisms for large-scale UAV simulations," in Proc. of Int. Conf. of Integration of Knowledge Intensive Multi-Agent Systems, Waltham, MA, USA, 2005.

[18] M. E. Latoschik, and H. Tramberend. "A scala-based actor-entity architecture for intelligent interactive simulations," Proc. of 5th IEEE Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2012, pp. 9-17.

[19] C.M. Macal, and M.J. North, "Tutorial on agent-based modelling and simulation, Journal of Simulation," vol. 4, no. 3, pp. 151–162, 2010.

[20] P. Mathieu, and Y Secq, "Environment Updating and Agent Scheduling Policies in Agent-based Simulators," in Proc. of 4th Int. Conf. on Agents and Artificial Intelligence (ICAART), Algarve, Portugal, 2012, pp. 170-175.

[21] M. S. Miller, E. D. Tribble, and J. Shapiro, "Concurrency among strangers," in Trustworthy Global Computing, Berlin, Germany: Springer, 2005, pp. 195-229.

[22] N. Minar, R. Burckhart, C. Langton, and V. Askenasi, 1996. "The Swarm simulation system: a toolkit for building multi-agent systems," Santa Fe Institute, Santa Fe, NM, USA. http://www.swarm.org/ [Accessed March 25, 2015].

[23] A. Negri, A. Poggi, M. Tomaiuolo, and P. Turci, "Agents for e-Business Applications," in 5th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems, Hakodate, Japan: ACM., 2006, pp. 907-914.

[24] M. J. North, T. R. Howe, N. T. Collier, and J. R. Vos, "The Repast Simphony runtime system," in Proc. of the Agent 2005 Conf. on Generative Social Processes, Models, and Mechanisms, Chicago, IL, USA, 2005

[25] M. J. North, N. Collier, and J. Vos, "Experiences in creating three implementations of the repast agent modeling toolkit," ACM Transactions on Modeling and Computer Simulation, vol. 16, no. 1, pp. 1-25, 2006.

[26] M. T. Parker, "What is Ascape and why should you care," Journal of Artificial Societies and Social Simulation, vol. 4, no. 1, 2001.

[27] A. Poggi, "Developing Scalable Applications with Actors," WSEAS Transactions on Computers vol. 14, pp. 660-669, 2014.

[28] A. Poggi, "Replaceable Implementations for Agent-Based Simulations," SCS M&S Magazine, vol. 4, no. 3, pp. 1-10, 2014.

[29] A. Poggi, and M. Tomaiuolo, "A DHT-based multi-agent system for semantic information sharing," in New Challenges in Distributed Information Filtering and Retrieval, Berlin, Germany: Springer, 2013, pp. 197-213.

[30] S.F. Railsback, S.L. Lytinen, and S.K. Jackson, "Agent-based simulation platforms: Review and development recommendations," Simulation, vol. 82, no. 9, pp. 609–623, 2006.

[31] C.W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," ACM SIGGRAPH Computer Graphics, vol. 21, no. 4, pp. 25-34, 1987.

[32] S. Tisue, and U. Wilensky, "Netlogo: A simple environment for modeling complexity," in Proc. of Int. Conf. on Complex Systems (ICCS 2004), 16-21, Boston, MA, USA, 2004, pp. 16-21.

[33] M.H. Zaharia, F. Leon, F. C. Pal, and G. Pagu, Agent-based simulation of crowd evacuation behavior, in Proc. of 11th WSEAS Int. Conf. on Automatic control, modelling and simulation (ACMOS'09), Istanbul, Turkey, 2009, pp. 529-533.

Simulations of Opinion Formation in Multi-Agent Systems using Kinetic Theory

Stefania Monica, Federico Bergenti
Dipartimento di Matematica e Informatica
Parco Area delle Scienze 53/A, 43124 Parma, Italy
Email: {stefania.monica, federico.bergenti}@unipr.it

***Abstract*—In this paper we formulate the problem of opinion formation using a physical metaphore. We consider a multi-agent system where each agent is associated with an opinion and interacts with any other agent. Interpreting the agents as the molecules of a gas, we model the evolution of opinion in the system according to a kinetic model based on the analysis of interactions among agents. From a microscopic description of each interaction between two agents, we derive the stationary profiles of the opinion under given assumption. Results show that, depending on the average opinion and on the parameters of the model, different profiles can be found, but all stationary profiles are characterized by the presence just of one or two maxima. Analytic results are confirmed by simulations shown in the last part of the paper.**

I. Introduction

In this paper we describe a model for opinion formation among agents. We assume that each agent is associated with an opinion $v \in I \subseteq \mathbb{R}$ and that it can change its opinion each time it interacts with another agent. In the literature, various approaches that study opinion evolution in a society of agents have been proposed, and many of them are based on *Cellular Automata* (*CA*) because CA describe well global effects of local phenomena (see, e.g., [1]). In order to overcome the synchronism that CA assume, in recent years the use of microscopic models inspired from physics has been introduced to describe asynchronous social interactions among agents in a society [2]. Such models are based on the idea that the laws of kinetic theory, which are typically used to describe the effects of interactions between two molecules of a gas, can also be used to model interactions between two agents.

Statistical mechanics and kinetic theory describe the details of each interaction between two molecules in a gas but they also allow finding classic laws which describe macroscopic properties of gases [3]. Analogously, from the microscopic laws which describe the details of each interaction between two agents, collective behaviour can be described from a macroscopic point of view [4]. All the research challenges related to the application of kinetic and statistical formalisms to describe multi-agent systems gave rise to new disciplines known as *econophysics* and *sociophysics* [5]. Such new disciplines have been used to describe, e.g., wealth evolution [6] and market economy [7], and they have also been adopted to characterize opinion evolution in a society [8].

In this paper, we focus on a model for opinion formation based on kinetic theory of gases and we analyze the evolution of the opinion in a society of agents. In particular, we analyze numerically the opinion evolution according to a model introduced in [9], which mandates that each agent can change its own opinion because of two different reasons [10]. The first reason is related to *compromise* between interacting agents. More precisely, the model assumes that both agents involved in an interaction can change their respective opinions in favor of that of the other agent. The second reason only involves each single agent and it is related to the fact that an agent can change its opinion autonomously, giving rise to a process known as *diffusion*.

This paper is organized as follows. In Section II, we describe the considered kinetic model from an analytic point of view. In Section III, we derive a stationary profile of opinion in a specific case. In Section IV, we show relevant simulation results. Section V concludes the paper.

II. Kinetic Formulation of Opinion Formation

Kinetic theory of gases describes, from a microscopic point of view, the effects of interactions among molecules in gases. By reinterpreting the molecules of a gas as agents, one can use the kinetic framework to describe social interactions among agents. While molecules are typically associated with relevant physical properties, like their velocities, agents can be associated with attributes that represent some of their characteristics. In particular, since in this work we are interested in modeling the evolution of the opinion, we assume that each agent is associated with a scalar parameter v which represents its opinion and which is defined in a given interval I. In the following, we consider $I = [-1, 1]$, where ± 1 represent extremal opinions.

In order to use the kinetic approach, we need to define a function, denoted as $f(w, t)$, which represents the density of opinion w at time t, and which is defined for each opinion $w \in I$ and for each time $t \geq 0$. According to such a definition,

$$\int_I f(w, t) \mathrm{d}w = 1. \tag{1}$$

In order to formulate the problem of opinion evolution in kinetic terms, we assume that the function $f(w, t)$ evolves on the basis of the Boltzmann equation, which, under our assumptions, can be written as

$$\frac{\partial f}{\partial t} = \mathcal{Q}(f, f)(w, t) \tag{2}$$

where $\mathcal{Q}$ is denoted as *collisional operator*. According to (2), the temporal evolution of the opinion density is governed by the collisional operator $\mathcal{Q}$, whose explicit formulation depends

on the details of binary interactions between any pairs of agents. Before deriving a formula for $\mathcal{Q}$, let us describe the effects of each binary interaction.

Denoting as (w, v) the opinions of two agents before their interaction, we assume that the following model holds

$$\begin{cases} w' = w + \gamma(v - w) + \eta D(|w|) \\ v' = v + \gamma(w - v) + \eta_* D(|v|) \end{cases} \tag{3}$$

where: (w', v') are the post-interaction opinions of the two agents; γ is a constant defined in $(0, \frac{1}{2})$; η and η_* are two independent random variables with the same statistics; and $D(\cdot)$ is a function that describes the impact of diffusion in the considered interaction [9]. From (3) it can be observed that the post-interaction opinions of the two agents are obtained by adding to their pre-interaction opinions two terms: the first one is related to compromise, while the second one is related to diffusion through function $D(\cdot)$.

Observe that the contribution of the compromise is proportional to the difference between the two pre-interaction opinions. Taking, for instance, the first equation in (3), we can conclude that the second addend on the right hand side of (3) is positive if $v > w$, so that the opinion of the considered agent (whose pre-interaction opinion is w) increases if it interacts with an agent with greater opinion. At the opposite, if $w > v$ the second addend is negative, so that the contribution of compromise decreases the opinion of the considered agent (towards that of the agent it interacts with). Observe that the contribution of compromise is negligible if $\gamma \simeq 0$, while it becomes relevant as γ increases.

Concerning the diffusion term, we assume that function $D(\cdot)$ depends on the absolute value of the opinion, meaning that the propensity of changing opinion is symmetrical with respect to 0 (namely, with respect to the middle point of I). Moreover, we assume that $D(\cdot)$ is non increasing with respect to the absolute value of the opinion, coherently with the fact that, typically, extremal opinions are more difficult to change. Finally, we assume that $0 \leq D(|w|) \leq 1$ for all $w \in I$. According to such assumptions, the contribution of diffusion can be either positive or negative depending on the value of η and η_*. In the following, we denote the probability density function of η and η_* as $\vartheta(\cdot)$ and we assume that

$$\begin{aligned} \int \eta \vartheta(\eta) \mathrm{d}\eta = \int \eta_* \vartheta(\eta_*) \mathrm{d}\eta_* = 0 \\ \int \eta^2 \vartheta(\eta) \mathrm{d}\eta = \int \eta_*^2 \vartheta(\eta_*) \mathrm{d}\eta_* = \sigma^2. \end{aligned} \tag{4}$$

Such a choice corresponds to considering η and η_* as 0 mean random variable with standard deviation σ.

The effects of diffusion in the opinion evolution are taken into account through the *transition rate*, which is defined as

$$W(w, v, w', v') = \vartheta(\eta)\vartheta(\eta_*)\chi_I(w')\chi_I(v') \tag{5}$$

where χ_I is the indicator function of set I (equals to 1 if its argument belongs to I, and to 0 otherwise). The indicator function in (5) is meant to impose that post-interaction opinions still belong to interval I [9].

Now that we have completed the definition of the law that describe each single interaction, we can write the explicit expression of the collisional operator $\mathcal{Q}$ used in (2), which is given by

$$Q(f, f) = \int_{\mathbb{B}^2} \int_I \left[{}'W \frac{1}{J} f({}'w) f({}'v) - W f(w) f(v) \right] \mathrm{d}v \mathrm{d}\eta \mathrm{d}\eta_*$$

where $\mathbb{B}$ is the support of ϑ, ${}'w$ and ${}'v$ are the pre-interaction variables which generate w and v, respectively, ${}'W$ is the transition rate and J is the Jacobian of the transformation of $({}'w, {}'v)$ in (w, v). The two addends in the previous equation represent the gain and the loss of agents in $\mathrm{d}w$, respectively [9].

In order to study the opinion evolution, we need to introduce the weak form of the Boltzmann equation. Generally speaking, the weak form of a differential equation is obtained by multiplying both sides by a test function, namely a smooth function with compact support, and integrating. The weak form of the Boltzmann equation can then be found by multiplying both sides of (2) by a test function $\phi(w)$ and integrating with respect to w. Using a proper change of variable in the integral, the weak form of the Boltzmann equation can be written as:

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}t} \int_I f(w, t) \phi(w) \mathrm{d}w = \\ &\int_{\mathbb{B}^2} \int_{I^2} W f(w) f(v) (\phi(w') - \phi(w)) \mathrm{d}w \mathrm{d}v \mathrm{d}\eta \mathrm{d}\eta_*. \end{aligned} \tag{6}$$

Setting $\phi(w) = 1$ in (6) leads to

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_I f(w, t) \mathrm{d}w = 0. \tag{7}$$

Such an equality corresponds to the fact that the number of agents is time invariant. This property is also found in classic kinetic theory and it corresponds to mass conservation.

Considering $\phi(w) = w$ as a test function and using (3) in (6) gives

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}t} \int_I f(w, t) w \mathrm{d}w = \\ &\gamma \int_{\mathbb{B}^2} \int_{I^2} W f(w) f(v) (v - w) \mathrm{d}w \mathrm{d}v \mathrm{d}\eta \mathrm{d}\eta_* \\ &+ \int_{\mathbb{B}^2} \int_{I^2} W f(w) f(v) \eta D(|w|) \mathrm{d}w \mathrm{d}v \mathrm{d}\eta \mathrm{d}\eta_*. \end{aligned} \tag{8}$$

Denoting as $u(t)$ the average value of the opinion at time t, namely

$$u(t) = \int_I f(w, t) w \, \mathrm{d}w \tag{9}$$

the left hand side of (8) corresponds to the derivative $\dot{u}(t)$ of the average opinion. Moreover, observe that the right hand side term of (8) is 0. As a matter of fact, the first integral is 0 for symmetry reasons, while the second integral is 0 because, according to (4), the average value of ϑ is 0. From (8) it can then be concluded that $\dot{u}(t) = 0$, and, therefore, the average opinion is conserved, namely $u(t) = u(0) = u$. This property corresponds to the conservation of momentum in kinetic theory.

We are now interested in studying the asymptotic behaviour of the distribution function $f(w, t)$. For this reason, in order to simplify notation, let us define a new temporal variable

$$\tau = \gamma t \tag{10}$$

where γ is the coefficient which appears in (3) and it is related to compromise. Assuming that $\gamma \to 0$, namely that each interaction causes small changes of opinions, the function

$$g(w,\tau) = f(w,t) \tag{11}$$

describes the asymptotic behaviour of $f(w,t)$. In [9] it is shown that by substituting $f(w,t)$ with $g(w,\tau)$ in (6) and using a Taylor series expansion of $\phi(w)$ around w in (6) the following equation of g can be derived

$$\frac{\mathrm{d}g}{\mathrm{d}\tau} = \frac{\lambda}{2}\frac{\partial^2}{\partial w^2}(D(|w|)^2 g) + \frac{\partial}{\partial w}((w-u)g) \tag{12}$$

where

$$\lambda = \sigma^2/\gamma. \tag{13}$$

Equation (12) is known in the literature as the weak form of the Fokker-Planck equation [11].

We are now interested in studying stationary solutions of this equation, namely those which satisfy

$$\frac{\mathrm{d}g}{\mathrm{d}\tau} = 0. \tag{14}$$

In the following, we denote such solutions as g_∞. In next section we analyze such solutions for different diffusion functions $D(|w|)$ and for different values of the parameter λ.

III. Results

In this section we derive relevant stationary profiles for the opinion density g. Such profiles are defined as solutions of (14) and, therefore, they depend on the parameters u and λ, which represent the average opinion and the ratio σ^2/γ, respectively, and on the choice of the diffusion function D. Recalling the initial assumptions on D, we develop our results considering

$$D(|w|) = 1 - |w| \qquad w \in I \tag{15}$$

which is symmetrical with respect to 0 and decreasing in $|w|$.

With this choice of the diffusion function, equations (3) become

$$\begin{cases} w' = w - \gamma(w-v) + \eta(1-|w|) \\ v' = v - \gamma(v-w) + \eta_*(1-|v|). \end{cases} \tag{16}$$

In order to guarantee that post-collisional opinions still belong to the considered interval I, we need to set the support of ϑ to

$$\mathbb{B} = (-(1-\gamma), 1-\gamma). \tag{17}$$

As a matter of fact, from (16)

$$\begin{aligned} |w'| &\le (1-\gamma)|w| + \gamma|v| + |\eta|(1-|w|) \\ &\le (1-\gamma)|w| + \gamma + |\eta|(1-|w|) \end{aligned} \tag{18}$$

and if, according to (17), $|\eta| \le (1-\gamma)$ then

$$|w'| \le (1-\gamma)|w| + \gamma + (1-\gamma)(1-|w|) = 1 \tag{19}$$

Hence, we can conclude that if $\eta \in \mathbb{B}$, then $w' \in I$. Analogous results can be derived for v'.

Now, if we substitute the expression of D defined in (15) in (12) the stationary solution g_∞ can then be found, according to (14), by solving the following partial differential equation.

$$\frac{\lambda}{2}\frac{\partial}{\partial w}\big((1-|w|)^2 g\big) + (w-m)g = C \tag{20}$$

where C is a constant which is necessarily 0. As a matter of fact, by integrating (20) one obtains

$$\frac{\lambda}{2}\int_{-v_1}^{v_2}\frac{\partial}{\partial w}\big((1-|w|)^2 g\big)\mathrm{d}w + \int_{-v_1}^{v_2}(w-m)g\mathrm{d}w = C(v_2+v_1)$$

from which, assuming that $v_1 \to 1$ and $v_2 \to 1$ one obtains

$$0 + m - m = 2C$$

which corresponds to $C = 0$.

Let us start by considering $w > 0$, so that (20) can be written as

$$\frac{\lambda}{2}(1-w)^2\frac{\partial g}{\partial w} + [(w-m) + \lambda(w-1)]\, g = 0. \tag{21}$$

Dividing both sides by g one obtains

$$\frac{g'}{g} = \frac{2}{(1-w)} + \frac{2(m-w)}{\lambda(1-w)^2} \tag{22}$$

and observing that

$$\frac{2(m-w)}{\lambda(1-w)^2} = \frac{\mathrm{d}}{\mathrm{d}w}\left(-\frac{2}{\lambda}\log(1-w) + \frac{2(m-1)}{\lambda(1-w)}\right) \tag{23}$$

equation (22) can be written as

$$(\log g(w))' = \left(\log(1-w)^{-2-\frac{2}{\lambda}} + \frac{2(m-1)}{\lambda(1-w)}\right)' \tag{24}$$

where we have used the facts that

$$\begin{aligned} \frac{g'}{g} &= (\log g(w))' \\ \frac{2}{(1-w)} &= -2(\log(1-w))'. \end{aligned} \tag{25}$$

Integrating (24) and applying the exponential function, one finally obtains the following expression for the stationary profile

$$g_\infty(w) = \tilde{c}_{u,\lambda}(1-|w|)^{-2-\frac{2}{\lambda}}\exp\left(\frac{2(m-1)}{\lambda(1-|w|)}\right) \tag{26}$$

where $\tilde{c}_{u,\lambda}$ is a normalization constant that depends on the average opinion u and on λ.

Let us now consider $w < 0$ so that equation (20) becomes

$$\frac{\lambda}{2}(1+w)^2\frac{\partial g}{\partial w} + [(w-m) + \lambda(w+1)]\, g = 0. \tag{27}$$

Dividing both sides by g leads to

$$\frac{g'}{g} = -\frac{2}{(1+w)} + \frac{2(m-w)}{\lambda(1+w)^2} \tag{28}$$

and by applying analogous calculation to the case with $w > 0$ one obtains

$$(\log g(w))' = \left(\log(1+w)^{-2-\frac{2}{\lambda}} - \frac{2(m+1)}{\lambda(1+w)}\right)'. \tag{29}$$

Integrating (29) leads to the following formula for the stationary profile

$$g_\infty(w) = \hat{c}_{u,\lambda}(1-|w|)^{-2-\frac{2}{\lambda}}\exp\left(\frac{-2(m+1)}{\lambda(1-|w|)}\right) \tag{30}$$

where w has been substituted by $-|w|$ and $\hat{c}_{u,\lambda}$ is a normalization constant.

Since g_∞ is the solution of a differential equation it must be continuous. From (26) and (30) it is evident that g_∞ is continuous for $w > 0$ and $w < 0$. Imposing that g_∞ is also continuous in $w = 0$, the following equality needs to hold

$$\tilde{c}_{u,\lambda} \exp\left(\frac{2u}{\lambda}\right) = \hat{c}_{u,\lambda} \exp\left(\frac{-2u}{\lambda}\right). \tag{31}$$

Finally, the solution of (20) is

$$g_\infty(w) = c_{u,\lambda}(1-|w|)^{-2-\frac{2}{\lambda}} \exp\left[-\frac{2(1-uw)}{\lambda(1-|w|)}\right] \tag{32}$$

where $c_{u,\lambda}$ is the quantity in (31) and it needs to be determined in order to ensure

$$\int_I g_\infty(w) = 1. \tag{33}$$

Observe that g_∞ is piecewise $\mathcal{C}^1$ and it is non-differentiable in $w = 0$ (as the function D). Moreover, the solution is symmetric if we change w and u with $-w$ and $-u$, namely

$$g_\infty(w; u, \lambda) = g_\infty(-w; -u, \lambda). \tag{34}$$

If $u = 0$, from (34) we can conclude that g_∞ is an even function. Moreover, using a change of variable for negative values of w, the integral of g_∞ can be written as

$$\int_{-1}^{1} g_\infty(w)\mathrm{d}w = 2c_{0,\lambda} \int_0^1 (1-w)^{-2-\frac{2}{\lambda}} \exp\left(\frac{-2}{\lambda(1-w)}\right) \mathrm{d}w$$

and, using the change of variable $t = \frac{-2}{\lambda(1-w)}$, the previous integral can be expressed as

$$2c_{0,\lambda}\left(\frac{\lambda}{2}\right)^{\frac{2}{\lambda}+1} \int_{\frac{2}{\lambda}}^{+\infty} t^{\frac{2}{\lambda}} e^{-t}\mathrm{d}t. \tag{35}$$

Finally, introducing the incomplete gamma function defined as

$$\Gamma(x, a) = \int_a^{+\infty} t^{x-1}\mathrm{e}^{-t}\mathrm{d}t, \tag{36}$$

the value of $c_{0,\lambda}$ which satisfies (33) is then

$$c_{0,\lambda} = \left[2\left(\frac{\lambda}{2}\right)^{\frac{2}{\lambda}+1}\Gamma\left(\frac{2}{\lambda}+1, \frac{2}{\lambda}\right)\right]^{-1}. \tag{37}$$

The case with $u = 0$ is the only one where the value of $c_{u,\lambda}$ can be found analytically. Other cases can be studied numerically.

We are now interested in studying the derivative of g_∞ in order to find singular points which correspond to maximum or minimum points. Deriving (26) it can be shown that if $w > 0$

$$g'_\infty(w) = 0 \iff 2\lambda(1-w) + 2(1-w) + 2(u-1) = 0.$$

Hence the (unique) singular point is

$$w = \frac{u+\lambda}{\lambda+1}$$

and it is positive if and only if $\lambda > -u$. Deriving (30), instead, it can be shown that if $w < 0$

$$g'_\infty(w) = 0 \iff 2\lambda(1+w) + 2(1+w) + 2(u+1) = 0$$

leading to the following singular point

$$w = \frac{u-\lambda}{\lambda+1}.$$

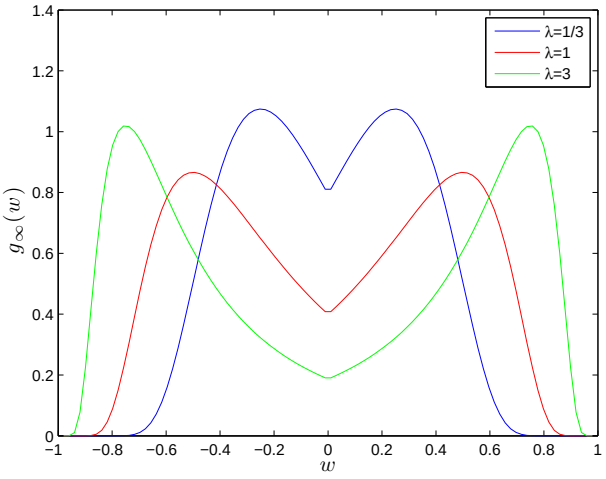


Fig. 1. Stationary profiles g_∞ for $u = 0$ and $\lambda = 1/3$ (blue line), $\lambda = 1$ (red line), and $\lambda = 3$ (green line)

Observe that this value is negative if and only if $\lambda > u$. Finally, the following cases can be considered

- if $u = 0$ then $g'_\infty(w) = 0$ in two points that are symmetric with respect to 0, namely $w = \pm\frac{\lambda}{\lambda+1}$
- if $u > 0$
  - if $0 < \lambda < u$ then $g'_\infty(w) = 0$ in a unique point, namely $w = \frac{u+\lambda}{\lambda+1}$
  - if $\lambda > u$ then $g'_\infty(w) = 0$ in two points, namely $w = \frac{u\pm\lambda}{\lambda+1}$
- if $u < 0$
  - if $0 < \lambda < -u$ then $g'_\infty(w) = 0$ in a unique point, namely $w = \frac{u-\lambda}{\lambda+1}$
  - if $\lambda > -u$ then $g'_\infty(w) = 0$ in two points, namely $w = \frac{u\pm\lambda}{\lambda+1}$

Observe that simple manipulations shows that

$$\begin{aligned} \lim_{w\to 0^+} g'_\infty(w) &> 0 \\ \lim_{w\to 0^-} g'_\infty(w) &< 0 \end{aligned} \tag{38}$$

so that $w = 0$, which is a non-differentiable point, can be considered as a point of minimum.

IV. Numerical Simulations

In this section, relevant numerical results are shown for stationary profiles for different values of u and λ. We focus on values of $u \geq 0$ as the stationary profiles relative to negative values of u can be obtained by symmetry, according to (34). The constant $c_{u,\lambda}$, which appears in g_∞, is evaluated numerically, using Newton-Cotes formulas [12].

First, we assume that $u = 0$ so that the average opinion corresponds to the middle point of I. As already observed in the previous section, if $u = 0$ then g_∞ is symmetric with respect to 0 and it has two maxima at

$$w = \pm\frac{\lambda}{\lambda+1}.$$

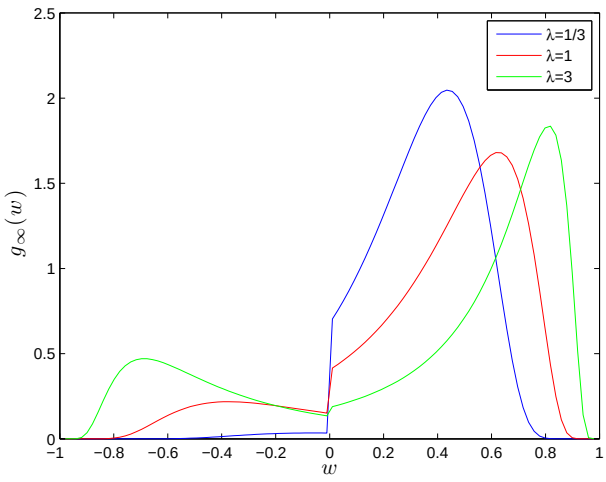


Fig. 2. Stationary profiles g_∞ for $u = 1/4$ and $\lambda = 1/3$ (blue line), $\lambda = 1$ (red line), and $\lambda = 3$ (green line)

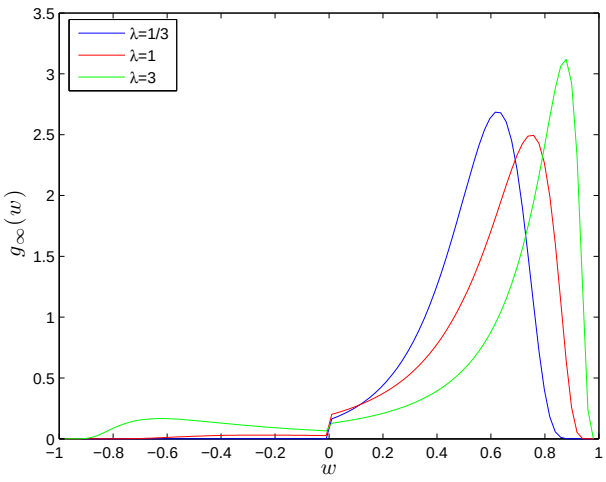


Fig. 3. Stationary profiles g_∞ for $u = 1/2$ and $\lambda = 1/3$ (blue line), $\lambda = 1$ (red line), and $\lambda = 3$ (green line)

As λ increases, such points get nearer to the extremal values of I, namely to extremal opinions. Observe that increasing the value of λ corresponds to incrementing the impact of diffusion with respect to compromise.

Fig. 1 shows the stationary profiles $g_\infty(w)$ when $u = 0$ for different values of λ. As expected from (34), the function $g_\infty(w)$ is symmetric with respect to 0 and it has a minimum in correspondence of $w = 0$ and two maxima whose values depend on λ. The stationary profiles $g_\infty(w)$ in Fig. 1 correspond to $\lambda = 1/3$ (blue line), $\lambda = 1$ (red line), and $\lambda = 3$ (green line). If $\lambda = 1/3$ the two maxima are in correspondence of $w = \pm 1/4$. Observe that in this case extremal distributions are associated with a very low probability. If $\lambda = 1$, instead, the maxima correspond to $w = \pm 1/2$; while in $\lambda = 3$ they correspond to $w = \pm 3/4$.

Therefore, it can be concluded that as λ increases the points of maximum move towards the extremes of the considered interval I. Observe that, according to its definition, any increase of λ corresponds to assuming that the contribution of diffusion is more relevant than that of compromise. Moreover, according to the results in Fig. 1, any increase of λ leads to stationary profiles with small values in correspondence of opinions in the middle of the interval I.

In Fig. 2 the stationary profiles $g_\infty(w)$ are shown when considering as average opinion the value $u = 1/4$. In this case, the function $g_\infty(w)$ is not symmetric and, as expected, it has a local minimum in $w = 0$. We consider the same values of λ as in the previous case. For each of these values, the number of maxima is two, since the condition $\lambda > u$ is always satisfied. If $\lambda = 1/3$ the positive maximum point is $w = 7/16$ and the negative one is $w = -1/16$. While the negative maximum is near the middle of the interval I, the positive one is farther. In Fig. 2 the stationary profile $g_\infty(w)$ obtained with $\lambda = 1/3$ is shown (blue line) and it can be observed that the value of the maximum in $w = 7/16$ is far more significant than that corresponding to $w = -1/16$, namely the positive opinions are far more likely than the negative ones. This is in agreement with the fact that the average opinion u is positive. If $\lambda = 1$ the maxima correspond to $w = \pm 5/8$ and $w = \mp 3/8$, as shown in Fig. 2 (red line). Finally, Fig. 2 also shows the results obtained with $\lambda = 3$ (green line). In this case the maximum points are $w = \pm 13/16$ and $w = \mp 11/16$. As in the previous case, the points of maximum get nearer to ± 1, namely to the extremes of I, as λ increases. Moreover, observe that largest values of λ correspond to increasing the likelihood of negative opinions.

Let us now increase the value of the average opinion to $u = 1/2$. The stationary profiles $g_\infty(w)$ are shown in Fig. 3 for $\lambda = 1/3$ (blue line), $\lambda = 1$ (red line), and $\lambda = 3$ (green line). Observe that if $\lambda = 1/3$ the function $g_\infty(w)$ has only one maximum, namely the positive one. As a matter of fact, according the results in the previous section, the negative one only exists if $\lambda > u$. The maximum point is $w = 5/8$. This value is greater than the one obtained for the same λ in the case $u = 1/4$, accordingly with the fact that, in this case, we consider a higher average opinion u. When considering $\lambda = 1$, the function $g_\infty(w)$ has two maxima since the condition $\lambda > u$ is satisfied. Such points are $w = 3/4$ and $w = -1/4$. As in Fig. 2, the value of the maximum corresponding to the negative value of w is less significant with respect to that relative to the positive value of w. If $\lambda = 3$ the two maxima correspond to $w = 5/8$ and $w = -7/8$ and they are nearer to the extremes of I with respect to those obtained with lower λ. A comparison of the results in Fig. 2 with those in Fig. 3 shows that in the latter the values of the positive maxima are greater while those of the negative maxima are smaller.

Finally, we consider a greater value of the average opinion, namely $u = 3/4$. This corresponds to considering an extremist society. Fig. 4 shows the stationary profiles for $\lambda = 1/3$ (blue line), $\lambda = 1$ (red line), and $\lambda = 3$ (green line). As in the previous case, since $\lambda < u$, the profile $g_\infty(w)$ has only one maximum if $\lambda = 1/3$. The point of maximum is $w = 13/16$ and it is closer to 1 than the other points of maximum obtained with the same λ for lower values of the average opinion u. From Fig. 4 it can be shown that, once again, the positive maximum point moves towards the extreme 1 as λ increases, since it corresponds to $w = 7/8$ if $\lambda = 1$ and to $w = 15/16$

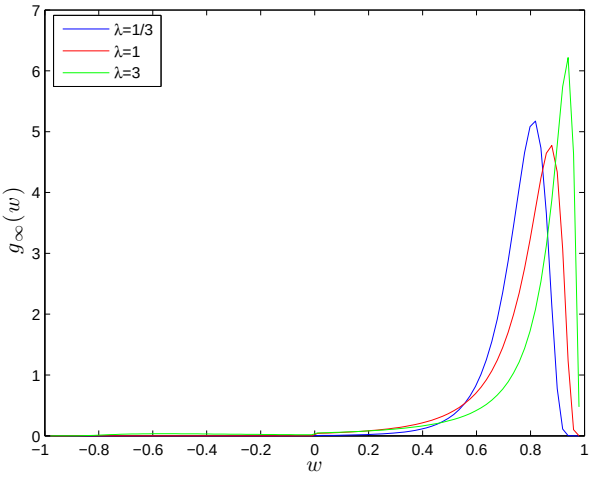


Fig. 4. Stationary profiles g_∞ for $u = 3/4$ and $\lambda = 1/3$ (blue line), $\lambda = 1$ (red line), and $\lambda = 3$ (green line)

if $\lambda = 3$. Moreover, Fig. 4 shows that an increase of the value of the average opinion significantly reduces the value of the negative maximum, which in this case is negligible. This result is not surprising since if the average opinion is near the positive extreme, then the number of agents with negative opinion has to be small.

V. Conclusions

In this paper we discussed the opinion dynamics in a multi-agent system through a kinetic approach. We considered an opinion evolution model inspired from the interactions of molecules in a gas and we studied, from an analytic point of view, the asymptotic behaviour of the opinion distribution. Assuming that the opinion of an agent can change each time it interacts with any other agent because of compromise and diffusion, we showed that the average opinion of the system is conserved. The stationary profiles can have different characteristics, depending on the parameters of the model and on the explicit expressions of the function which represents the diffusion. For a particular diffusion function, we showed that the asymptotic distribution is characterized by one or two maxima, depending on the parameters of the model. The stationary profiles are shown for different values of the average opinion and for different parameters of the model.

We recognize in kinetic models the possibility of describing complex and decentralized systems that can exhibit interesting emergent behaviours. In particular, we are mainly interested in using kinetic models as a conceptual framework that captures essential characteristics of opinion formation in multi-agent systems, and to adopt it in the design of mobile scenarios that would eventually use general-purpose industrial strength technology (see, e.g., [13], [14]). Moreover, we recognize that kinetic models can be effectively used to model agent-based cooperation (like the ones discussed in, e.g., [15]), and that they can be used to study large scale systems (like the ones discussed in, e.g., [16]). Finally, we are interested in modeling the emergent behaviors of wireless sensor networks used to support accurate localization (see, e.g., [17], [18]).

Further investigation on this subject is currently under development. In particular, we are interested in deriving the explicit expressions of the stationary profiles with a different choice of the diffusion function. We aim at studying the properties of such stationary profiles for different parameters of the model. At the same time, we are studying the application of kinetic models to multi-agents systems also from a simulative point of view. More precisely, we are interested in comparing analytic results with simulation experiments and in studying the number of iterations necessary to approximate to a certain degree an analytic stationary profile.

References

[1] S. Monica and F. Bergenti, "A stochastic model of self-stabilizing cellular automata for consensus formation," in *Proceedings of* 15^{th} *Workshop "Dagli Oggetti agli Agenti"* (WOA 2014), Catania, Italy, September 2014.

[2] L. Pareschi and G. Toscani, *Interacting Multiagent Systems: Kinetic Equations and Montecarlo Methods*. Oxford: Oxford University Press, 2013.

[3] M. Groppi, S. Monica, and G. Spiga, "A kinetic ellipsoidal BGK model for a binary gas mixture," *EPL: Europhysics Letter*, vol. 96, December 2011.

[4] W. Weidlich, *Sociodynamics: a systematic approach to mathematical modelling in the social sciences*. Amsterdam: Harwood Academic Publisher, 2000.

[5] B. K. Chakraborti, A. Chakrabarti, and A. Chatterjee, *Econophysics and sociophysics: Trends and perspectives*. Berlin: Wiley, 2006.

[6] F. Slanina, "Inelastically scattering particles and wealth distribution in an open economy," *Physical Review E*, vol. 69, pp. 46–102, 2004.

[7] S. Cordier, L. Pareschi, and G. Toscani, "On a kinetic model for simple market economy," *Journal of Statistical Physics*, vol. 120, pp. 253–277, 2005.

[8] K. Sznajd-Weron and J. Sznajd, "Opinion evolution in closed community," *International Journal of Modern Physics C*, vol. 11, pp. 1157–1166, 2000.

[9] G. Toscani, "Kinetic models of opinion formation," *Communications in Mathematical Sciences*, vol. 4, pp. 481–496, 2006.

[10] E. Ben-Naim, "Opinion dynamics: Rise and fall of political parties," *Europhysics Letters*, vol. 69, pp. 671–677, 2005.

[11] G. Toscani, "One-dimensional kinetic models of granular flows," *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 34, pp. 1277–1291, 2000.

[12] G. Naldi, L. Pareschi, and G. Russo, *Introduzione al Calcolo Scientifico*. The Mcgraw-Hill Company, 2001.

[13] F. Bergenti, G. Caire, and D. Gotta, "Agents on the move: JADE for Android devices," in *Procs. Workshop From Objects to Agents*, 2014.

[14] F. Bergenti, G. Caire, and D. Gotta, "Agent-based social gaming with AMUSE," in *Procs.* 5^{th} *Int'l Conf. Ambient Systems, Networks and Technologies (ANT 2014) and* 4^{th} *Int'l Conf. Sustainable Energy Information Technology (SEIT 2014)*, ser. Procedia Computer Science. Elsevier, 2014, pp. 914–919.

[15] F. Bergenti, A. Poggi, and M. Somacher, "A collaborative platform for fixed and mobile networks," *Communications of the ACM*, vol. 45, no. 11, pp. 39–44, 2002.

[16] F. Bergenti, G. Caire, and D. Gotta, "Large-scale network and service management with WANTS," in *Industrial Agents: Emerging Applications of Software Agents in Industry*. Elsevier, 2015, pp. 231–246.

[17] S. Monica and G. Ferrari, "Accurate indoor localization with UWB wireless sensor networks," in *Procs.* 23^{rd} *IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WETICE 2014), Parma, Italy, June 2014, pp. 287–289.

[18] S. Monica and G. Ferrari, "Swarm intelligent approaches to auto-localization of nodes in static UWB networks," *Applied Soft Computing*, vol. 25, pp. 426–434, December 2014.

Location-Aware JADE Agents in Indoor Scenarios

Stefania Monica, Federico Bergenti
Dipartimento di Matematica e Informatica
Parco Area delle Scienze 53/A, 43124 Parma, Italy
Email: {stefania.monica, federico.bergenti}@unipr.it

***Abstract*—This paper presents a novel JADE add-on that enables the implementation of location-aware agents by interfacing an underlying ranging technology which provides accurate distance measures both indoor and outdoor. First, the paper motivates the work and it presents the features and the architecture of the add-on. Then, the paper provides a detailed description of the implemented localization algorithms and it validates them in an indoor scenario. The experimental results show that accurate indoor localization can be achieved and that the presented add-on can be used to support location-aware agents with sufficient accuracy for targeted educational and ludic applications.**

I. Introduction

The use of software agents on mobile devices dates back to first cellular telephone prototypes capable of running Java applications, and JADE was one of the first tools that enabled FIPA agents on devices that were then called Java-enabled phones [1]. We used to call them *nomadic agents* [2] at the time—to make a clear distinction with then popular mobile agents—and they were considered one of the most promising applications of agent technology. More recently, the porting of JADE to Android devices [3], and the widespread adoption of JADE and related technologies in crucial parts of large-scale networks [4], revitalized the idea of having FIPA agents on mobile appliances to support large communities of users in their daily activities, which include both collaborative (see, e.g., [5], [6]) and competitive tasks (see, e.g., [7], [8]). Notably, the appliances of today offer much more resources than first Java-enabled phones, and the challenge of implementing agents for mobile devices is no longer about fitting complex software into constrained-resource devices; rather, it is about interfacing agents with the physical world they live in to ensure that user can be provided with contextualized services. This paper tackles the problem of interfacing agents with the physical world by presenting a novel software module which can be used to develop JADE agents that can sense the presence of nearby localization beacons or agents, both in outdoor and indoor scenarios. We prototyped such a module as a conventional JADE add-on which uses *Ultra Wide Band* (*UWB*) signals (see, e.g., [9]) to measure the distances between the appliance where the agent is running and target beacons or other appliances.

The acronym UWB was first used by the US Department of Defense in late '80s and it became popular after the Federal Communications Commission allowed the unlicensed use of UWB devices in February 2002 under specific emission constraints [10]. In 2004, IEEE established standardization group IEEE 802.15.4a with the aim of defining a new physical layer for the already existing IEEE 802.15.4 standard for *Wireless Personal Area Networks* (*WPANs*). In 2007 the new IEEE 802.15.4a standard was finally completed and since then it provides a standardized physical layer for short-range, low data rate communications, and for high-precision ranging using low-power devices [11].

The use of UWB signals is particularly promising for high-precision localization because it ensures high ranging accuracy. As a matter of fact, due to their large bandwidth, UWB signals are characterized by very short duration pulses—usually in the order of one nanosecond—which guarantee accurate *Time of Flight* (*ToF*) estimation for signals traveling between nodes. This implies that the distance between a transmitter and a receiver can be accurately determined, yielding high ranging accuracy. At the opposite, pulses received via multiple paths using conventional narrow-band signals can easily overlap, causing wrong ToF estimates, hence wrong range estimates [11]. Besides their short pulses, UWB signals are also characterized by low duty cycle which leads to low energy consumption. Moreover, since UWB signals occupy a large portion of the spectrum, in order to avoid interference problems with other devices operating in the same frequency spectrum, UWB systems normally use low-power transmissions. Finally, UWB signals are characterized by their capability of penetrating through obstacles thanks to the large frequency spectrum that characterizes them (which includes low-frequency components as well as high-frequency ones) [12]. Such a feature is particularly interesting in indoor environments where the presence of walls and objects can cause *Non-Line-of-Sight* (*NLoS*) effects between sensors.

Such unique aspects make UWB technology a good candidate for accurate and low-power positioning systems. One of the main drawbacks that caused the slow adoption of UWB technology for accurate indoor localization was that UWB transceivers were normally very expensive because of the intrinsic challenges that their design and construction involve, which also include high frequency logics for measuring very short delays. Only recently, in 2013, a company named *BeSpoon* (www.bespoon.com) started producing add-on modules for smartphones integrating an UWB transceiver and an antenna at a price compatible with the consumers' demands. They also provide a smartphone, called *spoonphone*, which natively accommodates the UWB module and which provides needed drivers for Android. This makes the development of accurate ranging techniques attractive also for general-purpose technologies, like JADE, which are not intended to tackle the issues of interfacing with proprietary, and expensive, hardware.

This paper is organized as follows. Section II describes the architecture of the add-on and it details the implemented localization algorithms. Section III describes the experimental campaign that we performed to validate the usability of the proposed approach. Section IV concludes the paper.

II. Distance- and Position-Aware JADE Agents

The presented add-on for JADE uses the tools and techniques that JADE provides to integrate an underlying ranging technology, like UWB signaling, with the common agent loop. In details, it provides two customizable JADE behaviours that can be used to access the services of the add-on.

The first behaviour, called `RangeChangeBehaviour`, is in charge of connecting with the underlying ranging technology to measure the distances with targeted beacons or smart appliances. The behaviour can accommodate any ranging technology, provided that an implementation of a specific interface is available. In the presented prototype we opt for an implementation that we developed using BeSpoon APIs which acquires real-time information from the UWB transceiver installed in a spoonphone. The `RangeChangeBehaviour` is scheduled when an above-threshold change in the distances from selected beacons or smart appliances occurs, or when the agent decides to change the set of devices that it monitors. In details, the implemented prototype uses hardware-specific unique identifiers associated with transceivers that allow targeting beacons and smart appliances indifferently. This way the agent developer is free to track beacons and smart appliances with a single, generic mechanism.

The second behaviour that ships with the add-on is called `LocationChangeBehaviour` and it uses the first behaviour to acquire needed ranging information to inform an agent about its current location in a predefined reference frame. The behaviour can be configured to acquire needed information for performing localization from a set of beacons, commonly known as *Anchor Nodes* (*ANs*), and its main duty is to invoke a pluggable localization algorithm to estimate the location of the agent every time the distances from ANs change significantly. The developer is free to implement custom localization algorithms but the add-on contains two general-purpose algorithms that can be used if no additional information besides the distance from a fixed number of ANs is available. Such algorithms are discussed in details below.

For the sake of simplicity, the implemented localization algorithms currently consider only a bi-dimensional scenario, i.e., they assume that all the nodes lay on the same plane. The same algorithms can be extended to the case of a three-dimensional scenario (e.g., see [13]). Both implemented localization algorithms are based on the ToF between some nodes with known positions, the ANs, and the *Target Node* (*TN*), whose position is to be estimated.

Denoting as M the number of ANs, we indicate the coordinates of the ANs as

$$\underline{s}_i \;=\; [x_i, y_i]^T \qquad i \in \{1, \ldots, M\}. \tag{1}$$

The (unknown) TN position and its estimate are denoted as $\underline{u} = [x, y]^T$ and $\underline{\hat{u}} \;=\; [\hat{x}, \hat{y}]^T$, respectively. Moreover, the true and the estimated distances between the $i-$th AN and the TN are denoted as

$$r_i \triangleq ||\underline{u} - \underline{s}_i|| = \sqrt{(\underline{u} - \underline{s}_i)^T(\underline{u} - \underline{s}_i)} \qquad i \in \{1, \ldots, M\}$$

$$\hat{r}_i \triangleq ||\underline{\hat{u}} - \underline{s}_i|| = \sqrt{(\underline{\hat{u}} - \underline{s}_i)^T(\underline{\hat{u}} - \underline{s}_i)} \qquad i \in \{1, \ldots, M\}.$$

Observe that if the coordinates of the ANs and the true distances $\{r_i\}_{i=1}^M$ are known, the true position of the TN can be found according to simple geometric considerations. As a matter of fact, the TN lies on each of the circumferences $\{\mathcal{C}_i\}_{i=1}^M$, centered in $\{\underline{s}_i\}_{i=1}^M$ with radii $\{r_i\}_{i=1}^M$ and, therefore, its coordinates satisfy the following system of equations

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = r_1^2 \\ \ldots \\ (x - x_M)^2 + (y - y_M)^2 = r_M^2. \end{cases} \tag{2}$$

Since the true distances are unknown, we can only rely on their estimates $\{\hat{r}_i\}_{i=1}^M$. The circumferences $\{\hat{\mathcal{C}}_i\}_{i=1}^M$ centered in $\{\underline{s}_i\}_{i=1}^M$ with radii $\{\hat{r}_i\}_{i=1}^M$ lead to the system of equations

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = \hat{r}_1^2 \\ \ldots \\ (x - x_M)^2 + (y - y_M)^2 = \hat{r}_M^2. \end{cases} \tag{3}$$

Obviously, due to the errors that affect the range estimates $\{\hat{r}_i\}_{i=1}^M$, the circumferences $\{\hat{\mathcal{C}}_i\}_{i=1}^M$ would hardly intersect in a unique point, hence proper localization algorithms need to be used. For such algorithms we assume that the errors which affect the range measurements $\{\hat{r}_i\}_{i=1}^M$ can be modeled as independent additive random variables $\{\nu_i\}_{i=1}^M$, namely

$$\hat{r}_i = r_i + \nu_i \qquad i \in \{1, \ldots, M\}. \tag{4}$$

Let us define as $\underline{\nu}$ the vector whose $i-$th element is ν_i and let us denote as $\underline{\underline{Q}}$ its covariance matrix.

Many range-based localization algorithms have been proposed in the literature, and they can be broadly classified into *passive* and *active* [14]. Passive localization relies on the fact that wireless communications strongly depend on the environment and it is based on the analysis of the scattering caused by obstacles found along signal propagation and/or of the variance of a measured signal. Such analysis allows finding changes in the received signals that can be used to detect and locate targets [15].

In active techniques, instead, all nodes are equipped with sensors and with an electronic device which sends needed information to support a proper localization algorithm. Since all implemented algorithms follow in this category, we focus on range-based localization with active tags, which can be based on the ToF, the *Angle of Arrival* (*AoA*), or the *Received Signal Strength* (*RSS*) of the signals [10]. As explained in the introduction, we are mostly interested in ToF-based localization algorithms because are particularly well suited when dealing with UWB signaling. In details, if two synchronized nodes communicate, the node receiving the signal can determine the *Time of Arrival* (*ToA*) of the incoming signal from the timestamp of the sending node. If the nodes are not synchronized, *Time Difference of Arrival* (*TDoA*) techniques can be employed, which are based on the estimation of the difference between the arrival times of UWB signals traveling between the TN and ANs. A large number of ToF-based localization techniques have been proposed in the literature. Among them, it is worth mentioning iterative methods [16], graph-based methods [17], closed-form algorithms [18] and optimization methods [19]. Observe that the accuracy of some of such algorithms may strongly depend on the number of ANs [20] and on their topology [21].

In the following subsections, the two implemented localization algorithms are described in details, namely: the *Circumference Intersection* (*CI*) algorithm and the *Two-Stage Maximum-Likelihood* (*TSML*) algorithm. From now on, we assume that $M = 3$ ANs are used to locate the TN.

A. Circumference Intersection (CI) algorithm

This subsection describes the CI localization algorithm, which is a very intuitive algorithm that can be considered as the basis of other more elaborate algorithms (see, e.g. [22]).

In order to better explain the CI algorithm, let us make a few geometric considerations on the considered problem. As already observed, the position of the TN coincides with the (unique) intersection of the three circumferences in (2). Since such circumferences are unknown, in order to find the TN position estimate $\hat{\underline{u}} = [\hat{x}, \hat{y}]$ it is necessary to consider proper localization approaches based on the set of equations (3). According to the CI algorithm, since the circumferences do not intersect in a single point, we intersect pairs of them.

More precisely, we define the following three sets—each of which contains two points—obtained by intersecting the three different pairs of circumferences

$$I_1 = \{\underline{p}_{12}, \underline{q}_{12}\} \triangleq \hat{\mathcal{C}}_1 \cap \hat{\mathcal{C}}_2 \quad (5)$$

$$I_2 = \{\underline{p}_{13}, \underline{q}_{13}\} \triangleq \hat{\mathcal{C}}_1 \cap \hat{\mathcal{C}}_3 \quad (6)$$

$$I_3 = \{\underline{p}_{23}, \underline{q}_{23}\} \triangleq \hat{\mathcal{C}}_2 \cap \hat{\mathcal{C}}_3. \quad (7)$$

We then choose a point from each of the three sets, namely $\underline{p}_1 \in I_1$, $\underline{p}_2 \in I_2$, and $\underline{p}_3 \in I_3$, so that the three selected points are the nearest ones to each other. More precisely, since such points belong to circumference intersections, it can be shown that they can be chosen as

$$||\underline{p}_1 - \underline{p}_2|| = \min_{\underline{p} \in I_1, \underline{q} \in I_2} ||\underline{p} - \underline{q}|| \quad (8)$$

$$||\underline{p}_1 - \underline{p}_3|| = \min_{\underline{q} \in I_3} ||\underline{p}_1 - \underline{q}||. \quad (9)$$

Given these three points, the TN position estimate is chosen as their baricenter.

We remark that the intersection between two circumferences can be empty. Assume, for instance, that in (5) the set I_1 is empty, namely the circumferences $\hat{\mathcal{C}}_1$ and $\hat{\mathcal{C}}_2$ do not intersect. In this case, the two nearest points of $\hat{\mathcal{C}}_1$ and $\hat{\mathcal{C}}_2$ are found, and if their distance is below a given threshold, they are considered the two intersection points and the set I_1 is redefined as the set containing such two points. Otherwise, the TN position estimate is found based on the remaining intersections, whenever possible.

B. Two-Stage Maximum-Likelihood (TSML) algorithm

This subsection describes a TSML method that uses ToA information, hence known as TSLM-ToA method [23]. Such method is a well-known localization algorithm, and it is particularly interesting because it was shown that it can attain the Cramer-Rao Lower Bound, which is a lower bound for the variance of an estimator [24]. A detailed derivation of this method can be found in [25].

The starting point of the TSML algorithm is once again the quadratic system (3), where we set $M = 3$ as in the case of the CI algorithm. In order to solve it, a two-step approach based on *Maximum-Likelihood* (*ML*) technique can be considered. First, let us define as a_i the Euclidean norm of the coordinate vector of the $i-$th AN, namely

$$a_i \triangleq ||\underline{s}_i|| = \sqrt{x_i^2 + y_i^2} \qquad i \in \{1, \ldots, 3\}. \quad (10)$$

Moreover, let us introduce the new variable

$$\hat{n} \triangleq ||\hat{\underline{u}}||^2 = \hat{x}^2 + \hat{y}^2 \quad (11)$$

so that system (3) can be rewritten, in matrix notation, as

$$\underline{\underline{G}}_1 \, \hat{\underline{\omega}}_1 = \hat{\underline{h}}_1 \quad (12)$$

where

$$\underline{\underline{G}}_1 = -2 \begin{pmatrix} x_1 & y_1 & -0.5 \\ \vdots & \vdots & \vdots \\ x_3 & y_3 & -0.5 \end{pmatrix}$$

$$\hat{\underline{\omega}}_1 = \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{n} \end{pmatrix} \qquad \hat{\underline{h}}_1 = \begin{pmatrix} \hat{r}_1^2 - a_1^2 \\ \vdots \\ \hat{r}_3^2 - a_3^2 \end{pmatrix}. \quad (13)$$

Let us also introduce $\underline{\omega}_1$ and $\underline{h}_1$ the vectors analogous to $\hat{\underline{\omega}}_1$ and to $\hat{\underline{h}}_1$, respectively, where estimated quantities are substituted by true ones, namely

$$\underline{\omega}_1 = \begin{pmatrix} x \\ y \\ n \end{pmatrix} \qquad \underline{h}_1 = \begin{pmatrix} r_1^2 - a_1^2 \\ \vdots \\ r_3^2 - a_3^2 \end{pmatrix}. \quad (14)$$

While (12) might look like a linear system, it is not, since the third element of the solution vector $\hat{\underline{\omega}}_1$ depends on the first two according to (11). The solution $\hat{\underline{\omega}}_1$ of the system (12) can be determined through an ML approach. In particular, as suggested in [18], let us define the error vector

$$\underline{\psi}_1 \triangleq \hat{\underline{h}}_1 - \underline{\underline{G}}_1 \underline{\omega}_1. \quad (15)$$

Given a positive definite matrix $\underline{\underline{W}}_1$, the weighted *Least Square* (*LS*) solution of (12) that minimizes $\underline{\psi}_1^T \underline{\underline{W}}_1 \underline{\psi}_1$ is

$$\hat{\underline{\omega}}_1 = (\underline{\underline{G}}_1^T \, \underline{\underline{W}}_1 \, \underline{\underline{G}}_1)^{-1} \underline{\underline{G}}_1^T \, \underline{\underline{W}}_1 \, \hat{\underline{h}}_1. \quad (16)$$

The simplest choice of the weighting matrix $\underline{\underline{W}}_1$ is the identity matrix. In [25] it is shown that the choice of $\underline{\underline{W}}_1$ which minimizes the variance of $\hat{\underline{\omega}}_1$ is

$$\underline{\underline{W}}_1 \triangleq \mathrm{Cov}[\underline{\psi}_1]^{-1} = (4 \underline{\underline{B}} \, \underline{\underline{Q}} \, \underline{\underline{B}})^{-1} \quad (17)$$

where $\underline{\underline{Q}}$ is the covariance matrix of the ToA range measurements $\{r_i\}_{i=1}^M$, $\underline{\underline{B}}$ is a diagonal matrix whose elements are $\{r_i\}_{i=1}^M$, and the last equality follows from the fact that, from (4) and (13), $\underline{\psi}_1$ can be written as

$$\underline{\psi}_1 = \hat{\underline{h}}_1 - \underline{h}_1 = 2 \underline{\underline{B}} \, \underline{\nu} + \underline{\nu} \odot \underline{\nu} \simeq 2 \underline{\underline{B}} \, \underline{\nu} \quad (18)$$

where $\odot$ denotes the entrywise product and the last approximation is obtained neglecting second order perturbations. With this choice of the weighting matrix one obtains that $\mathrm{Cov}[\hat{\underline{\omega}}_1] = (\underline{\underline{G}}_1^T \, \underline{\underline{W}}_1 \, \underline{\underline{G}}_1)^{-1}$.

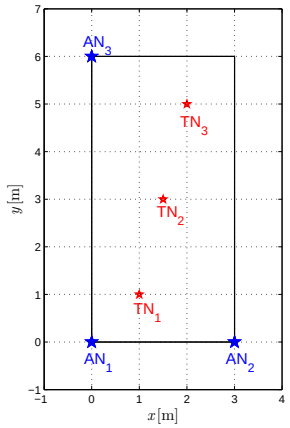


Fig. 1. The positions of the three considered ANs are shown (blue stars), together with the three different TN positions (red stars).

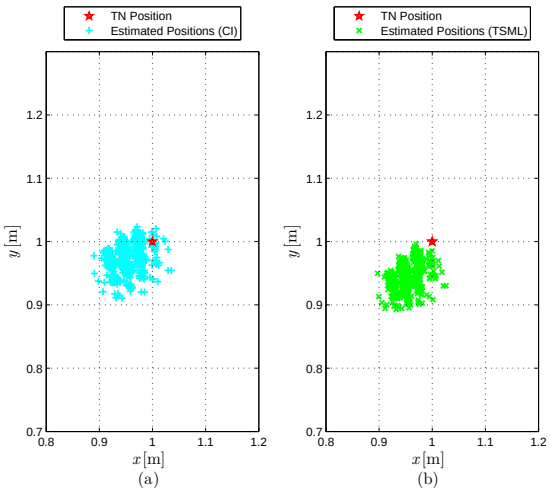


Fig. 2. The 1000 position estimates of TN_1 obtained using: (a) the CI algorithm; and (b) the TSML algorithm.

The second stage of the algorithm is meant to take into account the dependence of $\hat{n}$ on the other elements of the unknown vector in the system of equations (12) and it involves the solution of the system

$$\underline{\underline{G}}_2 \, \underline{\hat{\omega}}_2 = \underline{\hat{h}}_2 \tag{19}$$

where $\underline{\hat{\omega}}_2 = (\hat{x}^2, \hat{y}^2)^T$ and

$$\underline{\underline{G}}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \qquad \underline{\hat{h}}_2 = \begin{pmatrix} [\underline{\hat{\omega}}_1]_1^2 \\ [\underline{\hat{\omega}}_1]_2^2 \\ [\underline{\hat{\omega}}_1]_3 \end{pmatrix} \tag{20}$$

with $[\underline{\hat{\omega}}_1]_j$ denoting the $j-$th component of $\underline{\hat{\omega}}_1$. The linear system (19) can be solved, once again, through the ML technique. Defining the error vector

$$\underline{\psi}_2 \triangleq \underline{\hat{h}}_2 - \underline{\underline{G}}_2 \, \underline{\omega}_2 \tag{21}$$

the weighted LS solution of (19) that minimizes the weighted norm of $\underline{\psi}_2$ with a positive definite matrix $\underline{\underline{W}}_2$ is

$$\underline{\hat{\omega}}_2 = (\underline{\underline{G}}_2^T \, \underline{\underline{W}}_2 \, \underline{\underline{G}}_2)^{-1} \underline{\underline{G}}_2^T \, \underline{\underline{W}}_2 \, \underline{\hat{h}}_2. \tag{22}$$

As considered to solve (12), the simplest choice of the weighting matrix $\underline{\underline{W}}_2$ is the identity matrix. In [25], it is shown that the choice of $\underline{\underline{W}}_2$ which minimizes the variance of $\underline{\hat{\omega}}_2$ is

$$\underline{\underline{W}}_2 \triangleq \mathrm{Cov}[\underline{\psi}_2]^{-1} = (4 \underline{\underline{B}}_2 \, \mathrm{Cov}[\underline{\hat{\omega}}_1] \, \underline{\underline{B}}_2)^{-1} \tag{23}$$

where $\underline{\underline{B}}_2 = \mathrm{diag}(x, y, 0.5)$. Finally, the position estimate can be expressed as

$$\underline{\hat{u}} = \underline{\underline{U}} \left[\sqrt{[\underline{\hat{\omega}}_2]_1}, \sqrt{[\underline{\hat{\omega}}_2]_2} \right]^T \tag{24}$$

where $\underline{\underline{U}} = \mathrm{diag}(\mathrm{sign}(\underline{\hat{\omega}}_1))$.

III. Experimental Setup and Results

This section shows localization results obtained with the two implemented algorithms, as described in the previous section, using a JADE agent running on a spoonphone used as TN. We use three localization beacons as ANs and we put them at three corners of a rectangular room whose sides measures 3 m and 6 m. Using the same notation introduced in Section II, the ANs position in the experimental setup are denoted as

$$\underline{s}_1 = [0, 0]^T \qquad \underline{s}_2 = [3, 0]^T \qquad \underline{s}_3 = [0, 6]^T \tag{25}$$

where the coordinates are expressed in meters. In Fig. 1 the ANs positions are shown (blue stars) in the considered room.

Inside the same room, we consider three different TN positions, denoted as red stars in Fig. 1. First, we put the TN in the points whose coordinates, expressed in meters, are

$$\underline{u}_1 = [1, 1]^T. \tag{26}$$

Observe that this point, denoted as TN_1 in Fig. 1, is close to one of the corners of the room and close to AN_1. The second TN position we consider is

$$\underline{u}_2 = [1.5, 3]^T \tag{27}$$

which corresponds to the point in the middle of the room, denoted as TN_2 in Fig. 1. Finally, the coordinates of the last TN position, expressed in meters, are

$$\underline{u}_3 = [2, 5]^T. \tag{28}$$

Observe that this point is symmetric of $\underline{u}_1$ with respect to the center of the room.

For each TN position $\{\mathrm{TN}_i\}_{i=1}^3$, we first acquire the three distance estimates between the TN and the three ANs and we use such distances to feed the two localization algorithms described in Section II. This process is iterated $N_I = 1000$ times, thus obtaining 1000 position estimates for each of the two localization algorithms and for each TN position. For each iteration $j \in \{1, \ldots, N_I\}$ we define the distance between the true TN position and its estimate in the j-th iteration as

$$d_j = ||\underline{\hat{u}}_j - \underline{u}|| \tag{29}$$

where $\underline{\hat{u}}_j$ is the TN position estimate in the j-th iteration.

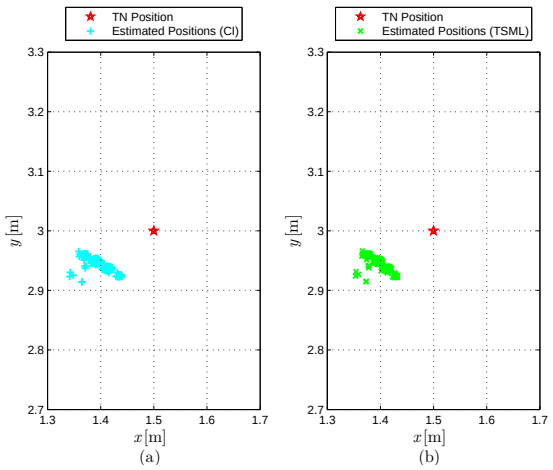


Fig. 3. The 1000 position estimates of TN_2 obtained using: (a) the CI algorithm; and (b) the TSML algorithm.

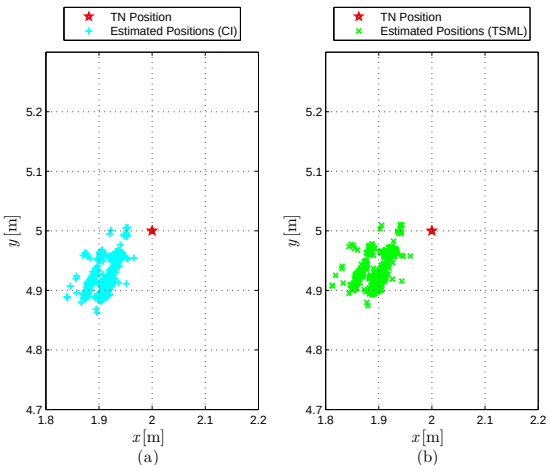


Fig. 4. The 1000 position estimates of TN_3 obtained using: (a) the CI algorithm; and (b) the TSML algorithm.

We can define the minimum, the maximum, and the average distance between the true TN position and its estimates as

$$\begin{aligned} d_{\min} &= \min_{j\in\{1,\ldots,N_I\}} d_j \\ d_{\max} &= \max_{j\in\{1,\ldots,N_I\}} d_j \\ d_{\text{avg}} &= \frac{1}{N_I}\sum_{j=1}^{N_I} d_j. \end{aligned} \tag{30}$$

The performance of the localization algorithms is evaluated in terms of the values in (30).

Fig. 2 is relative to the position estimates of the target TN_1 of Fig. 1. More precisely, Fig. 2 (a) shows the 1000 position estimates obtained using the CI algorithm and Fig. 2 (b) shows the 1000 position estimates obtained according to the TSML algorithm. A comparison between Fig. 2 (a) and Fig. 2 (b) shows that the CI algorithm performs slightly better that the TSML algorithm in terms of the distance between the true position and the estimated ones. As a matter of fact, from Table I it can be observed that while the average distance d_{avg} when using the CI algorithm is 5.7 cm, the value of d_{avg} obtained with the TSML algorithm is 7.2 cm. Analogous considerations hold when considering $d_{\min}$ and $d_{\max}$. More precisely, the value of $d_{\min}$ relative to the CI algorithm is only 0.03 cm and it becomes 1.4 cm when using the TSML algorithm, while the values of $d_{\max}$ correspond to 12.2 cm and 13.8 cm, respectively.

Fig. 3 refers to the position estimates of the target TN_2 of Fig. 1, corresponding to the case where the TN is in the middle of the room. The 1000 position estimates obtained using the CI algorithm are shown in Fig. 3 (a) while Fig. 2 (b) shows the 1000 position estimates obtained with the TSML algorithm. In this case, the performance of the two algorithms are similar, as also shown in Table I, which shows that the values of d_{avg} obtained with the CI and the TSML algorithms differ by only 2 mm. Observe that in this case the values of d_{avg} are greater than those obtained when considering TN_1, meaning that the localization in the center of the room is less accurate. In particular, Table I shows that the distances between the true TN position and its estimates are always greater than $d_{\min} \simeq 10$ cm.

TABLE I. VALUES OF MINIMUM, MAXIMUM, AND AVERAGE DISTANCES BETWEEN THE TRUE TN POSITIONS AND THEIR ESTIMATES.

	$d_{\min}$ [cm]	$d_{\max}$ [cm]	d_{avg} [cm]
TN_1 - CI	0.03	12.2	5.7
TN_1 - TSML	1.4	13.8	7.2
TN_2 - CI	9.6	17.6	11.6
TN_2 - TSML	10.1	16.5	11.4
TN_3 - CI	4.6	19.5	11.5
TN_3 - TSML	5.7	20.8	12.2

Finally, Fig. 4 is relative to the position estimates of the target TN_3 of Fig. 1 obtained: (a) using the CI algorithm; and (b) using the TSML algorithm. In this case, the average distance d_{avg} between the true TN position and its estimates is 11.5 cm when using the CI algorithm and 12.2 cm with the TSML algorithm, as shown in Table I. The performance of the two algorithms are similar also in terms of $d_{\min}$ and $d_{\max}$. Observe that the values of $d_{\max}$ are greater than those obtained in the previous two TN positions. The values of $d_{\min}$, instead, are greater than those relative to TN_1 but they are smaller than those relative to TN_2.

IV. CONCLUSIONS

This paper presents a novel JADE add-on that enables the implementation of distance- and location-aware agents. Using this add-on an agent can measure the distance that separates the smart appliance that hosts it from target beacons and other smart appliances. Experimental results summarized in Table I show that the average error in locating the smart appliance in an empty room is less than 15 cm, which ensures sufficient accuracy for considered education and ludic applications.

REFERENCES

[1] F. Bergenti and A. Poggi, "Ubiquitous information agents," *Int'l J. Cooperative Information Systems*, vol. 11, no. 34, pp. 231–244, 2002.

[2] F. Bergenti, A. Poggi, B. Burg, and G. Caire, "Deploying FIPA-compliant systems on handheld devices," *IEEE Internet Computing*, vol. 5, no. 4, pp. 20–25, 2001.

[3] F. Bergenti, G. Caire, and D. Gotta, "Agents on the move: JADE for Android devices," in *Procs. Workshop From Objects to Agents*, 2014.

[4] F. Bergenti, G. Caire, and D. Gotta, "Large-scale network and service management with WANTS," in *Industrial Agents: Emerging Applications of Software Agents in Industry*. Elsevier, 2015, pp. 231–246.

[5] F. Bergenti and A. Poggi, "Agent-based approach to manage negotiation protocols in flexible CSCW systems," in *Procs.* 4^{th} *Intl Conf. Autonomous Agents*, 2000, pp. 267–268.

[6] F. Bergenti, A. Poggi, and M. Somacher, "A collaborative platform for fixed and mobile networks," *Communications of the ACM*, vol. 45, no. 11, pp. 39–44, 2002.

[7] F. Bergenti, G. Caire, and D. Gotta, "An overview of the AMUSE social gaming platform," in *Procs. Workshop From Objects to Agents*, 2013.

[8] F. Bergenti, G. Caire, and D. Gotta, "Agent-based social gaming with AMUSE," in *Procs.* 5^{th} *Int'l Conf. Ambient Systems, Networks and Technologies (ANT 2014) and* 4^{th} *Int'l Conf. Sustainable Energy Information Technology (SEIT 2014)*, ser. Procedia Computer Science. Elsevier, 2014, pp. 914–919.

[9] S. Monica and G. Ferrari, "Accurate indoor localization with UWB wireless sensor networks," in *Proceedings of the* 23^{rd} *IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WETICE 2014)*, track on Capacity-Driven Processes and Services for the Cyber-Physical Society* (CPS), Parma, Italy, June 2014, pp. 287–289.

[10] Z. Sahinoglu, S. Gezici, and I. Guvenc, *Ultra-wideband positioning systems: Theoretical limits, ranging algorithms and protocols*. Cambridge, U.K.: Cambridge University Press, 2008.

[11] J. Zhang, P. V. Orlik, Z. Sahinoglu, A. F. Molisch, and P. Kinney, "UWB systems for wireless sensor networks," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 313–331, February 2009.

[12] S. Gezici and H. V. Poor, "Position estimation via Ultra-Wide-Band signals," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 386–403, February 2009.

[13] S. Monica and G. Ferrari, "Optimized anchors placement: An analytical approach in UWB-based TDOA localization," in *Proceedings of the* 9^{th} *International Wireless Communications & Mobile Computing Conference* (IWCMC 2013), Cagliari, Italy, July 2013, pp. 982–987.

[14] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 6, pp. 1067–1080, November 2007.

[15] D. Dardari and R. D'Errico, "Passive ultrawide bandwidth RFID," in *Proceedings of the IEEE Global Telecommunications Conference* (GLOBECOM '08), New Orleans, LA, December 2008, pp. 1–6.

[16] C. Mensing and S. Plass, "Positioning algorithms for cellular networks using TDOA," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP 2006), Toulouse, France, May 2006, pp. 513–516.

[17] Y. Zheng, W. Chenshu, C. Tao, Z. Yiyang, G. Wei, and L. Yunhao, "Detecting outlier measurements based on graph rigidity for wireless sensor network localization," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 1, pp. 374–383, January 2013.

[18] Y. Chan and K. C. Ho, "A simple and efficient estimator for hyperbolic location," *IEEE Transactions on Signal Processing*, vol. 42, no. 8, pp. 1905–1915, August 1994.

[19] S. Monica and G. Ferrari, "Swarm intelligent approaches to auto-localization of nodes in static UWB networks," *Applied Soft Computing*, vol. 25, pp. 426–434, December 2014.

[20] S. Monica and G. Ferrari, "Impact of the number of beacons in PSO-based auto-localization in UWB networks," in *Proceedings of the International Conference on the Applications of Evolutionary Computation* (EvoApplications '13)*, track on Nature-inspired Techniques for Communication Networks and other Parallel and Distributed Systems* (EvoCOMNET '13), Vienna, Austria, April 2013, pp. 42–51.

[21] S. Monica and G. Ferrari, "Particle swarm optimization for auto-localization of nodes in wireless sensor networks," in *Proceedings of the* 11^{th} *International Conference on Adaptive and Natural Computing Algorithms* (ICANNGA '13), Lausanne, Switzerland, April 2013, pp. 456–465.

[22] S. Monica and G. Ferrari, "An experimental model for UWB distance measurements and its application to localization problems," in *Proceedings of the IEEE International Conference on Ultra Wide Band* (ICUWB 2014), Paris, France, September 2014, pp. 297–302.

[23] S. Monica and G. Ferrari, "A swarm intelligence approach to 3D distance-based indoor UWB localization," in *Proceedings of the International Conference on the Applications of Evolutionary Computation* (EvoApplications '15)*, track on Nature-inspired Techniques for Communication Networks and other Parallel and Distributed Systems* (EvoCOMNET '15), Copenaghen, Denmark, April 2015, pp. 42–51.

[24] K. C. Ho, X. Lu, and L. Kovavisaruch, "Source localization using TDOA and FDOA measurements in the presence of receiver location errors: Analysis and solution," *IEEE Transactions on Signal Processing*, vol. 55, no. 2, pp. 684–696, February 2007.

[25] G. Shen, R. Zetik, and R. S. Thomä, "Performance comparison of TOA and TDOA based location estimation algorithms in LOS environment," in *Proceedings of the* 5^{th} *Workshop on Positioning, Navigation and Communication* (WPNC 2008), Hannover, Germany, March 2008, pp. 71–78.

Mobile Agents with Recurrent Neural Networks-based Computing Model for Echo Cancellation Problem

F. Bonanno
Department of Electrical, Electronics and Informatics Engineering
University of Catania
Catania, Italy

G. Capizzi
Department of Electrical, Electronics and Informatics Engineering
University of Catania
Catania, Italy

G. Lo Sciuto
Department of Engineering
Roma Tre University
Rome, Italy

***Abstract*—The purpose of this paper is to draw attention to a novel solving method for AEC problem and to search for a solution based on mobile agents (MAs) technology. Several applications benefit already by the use of MAs technology and we now want to join a soft computing technology as recurrent neural networks (RNNs) for this problem's area in communication systems. In this paper we propose a MAs with RNNs-based, so as pipelined recurrent neural networks (PRNNs), computing model for AEC's problem in communication systems. Prediction of echo paths can be performed by RNNs and PRNNs based processing. The results about the faced problem in communication systems, where the use of MAs with RNNs implementation might be an improvement over a more conventional solutions, are here summarized, presented and evaluated. Satisfactory performance in echo cancellation were obtained. The two echo signals input, as echo remote and echo local are provided as inputs to the RNN and PRNN and the echo output prediction shows a relevant reduction evaluated as 60 db.**

I. Introduction

Several are the application of adaptive filtering algorithms in communication systems problems as for speech recognition, echo cancellation, interference suppression, noise cancellation and acoustic echo cancellation. There are also two main categories of audio analysis problems area that could be investigated as sound matching and speech recognition.

Real-time speech recognition is an important task in current digital communication systems such as mobile telephone systems, adaptive filtering, algorithm for echo cancellation etc... In this paper we deal with the use of mobile agents (MAs) technology with adaptive filtering for thefor echo cancellation problem. The agent concept has been widely adopted in many areas such as: control system, network management, information management, E-commerce.

A mobile agent is the composition of computer program and data which can travel from one computing platform to another. The agent technology becomes popular for the reasons such as: parallel performance of tasks, dynamic adaptation to changing conditions, easy deployment of new program and being able to exchange information. In a mobile agent network, agents can carry data and programs while moving from one computing platform to another and one task can be decomposed into several sub-tasks. These agents work cooperately and dynamically adapt themselves to the changing environment and as known scalability is one important feature of an agent network.

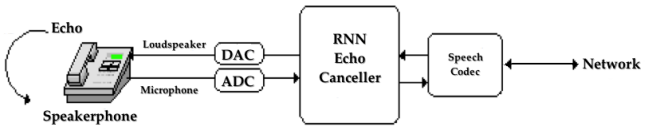


Fig. 1. A problem in Acoustic Echo Cancellation: Line/Network Echo Cancellation.

Several are the existing available agent platforms including (MOLE, Aglets, Concordia, Ara, TACOMA, and Mobile-C is an embeddable mobile agent system compliant with Foundation for Intelligent Physical Agents (FIPA) that is an internationally recognized agent standards.

We have implemented common RNNs and PRNNs for adaptive filtering but in recent studies appeared in literature, to reduce the computational complexity of the bilinear recurrent neural network (BLRNN), a novel low-complexity nonlinear adaptive filter with a pipelined bilinear recurrent neural network (PBLRNN) was presented by some researchers.

The present paper in this area includes development and implementation of new computing model for adaptive filtering by using MAs in conjunction with RNNs. MAs is one promising new paradigms for distributed application. A mobile agent consists of the program code and the program execution state. Initially, a mobile agent resides on a computer called home machine or dispatching server. The agent is then dispatched to execute on a remote computer called mobile agent host. When a mobile agent is dispatched, its entire code and the execution state are transferred to the mobile agent host. The host provides a suitable execution environment for the mobile agent.

A. The Acoustic Echo Cancellation Problem

Echo Cancellation is used to enhance speech for Radio, Mobile, VoIP applications are available for echo cancellation solutions including acoustic echo cancellers (AEC) and line or network echo cancellers. AEC is an essential part for providing voice quality enhancement in telephone communications (see Fig. 1).

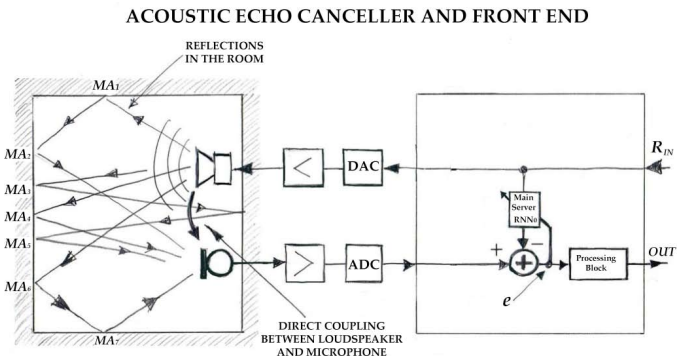


Fig. 2. Schematic of a proposed echo cancellation system based on MAs with RNNs and a main Server RNN0

Cancellation is the reduction of the reflected copies of a direct path wave in a signal. An AEC operates on the digitally sampled audio signals of the communication device. The transfer function of the acoustic environment from the loudspeaker to the microphone on the device is estimated to cancel the received echoes from the microphone signal then an AEC is required.

An adaptive filter is conventionally used in voice echo cancellation to accommodate the time varying nature of the echo path. The filter learns the path when the far-end speaker is talking and the near-end speaker is silent and adjusts its coefficients according to the algorithm optimization. For an adaptive filter to learn the echo path it must have an undisturbed reference signal to adapt to. Unfortunately in double-talk detection this filtering scenario cannot be admitted as the near-end speaker may want to interrupt the far-end speaker. In other words, the near-end and far-end speakers talking simultaneously or double-talk resulting modified reference signal.

Non-linear processing is the removal of residual echo left by the adaptive filter. Residual echoes are the un-modeled components of the echo path. Most used adaptive filters are linear and can only cancel the linear portions of the echo path. Thus the nonlinear portions cannot be removed via the adaptive filter and a residual echo removal follows the filter to handle nonlinear portions of the echo remaining. We have gained a large experience in the development and implementation of echo cancellation solutions and now we use MAs with RNNS as shown in Fig. 2.

B. Non-conventional Applications of Acoustic Echo Cancellation

The thinking of people about AEC is its application as a requirement of a conferencing speakerphones or using a wireless handset in hands-free mode. In these situations, the loudspeaker and microphone are enclosed in the same device. Therefore, the physical characteristics of the device help shape the echo path. AEC can be applied to any voice communication system requiring to achieve a high quality full-duplex conversation. For example, AEC can be applied to a drive-thru order post, home intercom systems, baby monitors, patient-care intercom systems in hospitals and imaging centers, VoIP communications on laptops, videophones, and human/machine interfaces. All of these applications present their unique set of challenges.

In drive-thru order posts, reflections off of curbs and other various buildings and structures create an unique acoustic environment to every deployment. The impulse response (echo path) will be very dissimilar to that of a typical office environment and potentially could have a long echo tail. Besides the uniqueness of the echo path, a drive-thru application also has to be able to handle the non-stationary aspects of the background noise. This makes a noise reduction algorithm a requirement with AEC. There are also applications in which the loudspeaker and microphone are not physically tied to the same device, as in distributed multimedia systems. This reduces the acoustic coupling between loudspeaker and microphone. In this situation assuring synchronization between the sampling rates of the loudspeaker and microphone becomes an additional burden.

At present, most teleconferencing systems involve a single full-duplex audio channel for voice communication. These systems usually employ an acoustic echo canceler to remove undesired echoes that result from coupling between the loudspeaker and microphone. As these systems evolve to transparent audio-video medium, the need for enhanced sound realism becomes more important.

This need leads to consideration of multichannel audio, which at a minimum involves two channels, i.e., stereophonic sound. However, before full-duplex stereophonic teleconferencing can be deployed, the AEC problem must be solved.

In this section we have reported these considerations in order to discuss preliminary some unsuccessful attempts in AEC problems area. Several applications benefit already by the use of mobile agents technology and we now want to use a soft computing technology as RNNs for this problem area in communication systems. In this paper we propose a MAs with RNNs and PRNN-based computing model for AECs problem in communication systems. Prediction of echo paths can be performed by MAs-RNNs based processing. The results about the faced problem in communication systems, where the use of MAs implementation coupled with RNNs might be an improvement over a more conventional solutions, are here summarized, presented and evaluated. As mentioned we do not know any satisfactory solution to the echo cancellation problems so basically in this paper a computing model and algorithm is presented for RNNs based adaptive filtering with application to AEC. This computing model can be called as generalized MAs with RNNs-PRNN and is derived as a MAs implementation. The basics of the implementation are then introduced so as the obtained results.

II. SOME TECHNIQUES FOR AEC

In applications such as acoustic echo cancellation the impulse response of the system often reaches over 100ms in length. This would require an adaptive FIR filter with over 1000 coefficients. The linear convolution and the update of the adaptive filter with this length creates a significant computational burden for applications that require low power processors. The application of the adaptive IIR filters often fail to produce the desired results despite their reduced complexity is because the adaptation of the IIR filter contains many local minima and instabilities. As the efficiency of Fast Fourier Transforms (FFTs) have improved, block processing and frequency domain adaptive filters (FDAF) were realized on low power DSPs.

FDAF provide several advantages over its time domain counterpart. Besides being able to perform the filter convolution by a multiplication in frequency domain, also the length of the adaptive filter are effectively decimated by the transformation. Thus, the computational complexity of the adaptive algorithm is reduced. In addition FDAF can also provide an increased convergence speed.

AEC must incorporate a sub-band adaptive filter whose adaptation speed should be superior against conventional solutions As mentioned earlier, applications such as acoustic echo cancellation can have long echo paths, resulting in a large delay and memory requirement. This disadvantage can be overcome by methods such as the multidelay adaptive filters. In this approach the block size can be smaller than the required time domain adaptive filter. In this paper an MAs based algorithm is presented for adaptive filtering in the frequency-domain.

A. Adaptive Filtering in Echo Cancellation

The Echo Cancellation based on Adaptive IIR Filtering is here summarized. Echo cancellation solutions are most often based on a linear FIR adaptive filtering approach. IIR adaptive filters typically use a much smaller number of coefficients to model a system but require additional complexity to control stability during adaptation. This conceptual simplicity has a cost: even a modest approximate model of the echo path has a large number of filter coefficients. A very good approximation of the same real echo path would have much smaller number of coefficients then we use RNNs and then PRNN for adaptive filtering. In literature is reported an outline of the IIR filter-based echo canceller solutions.

In the past the Kalman Filter and the Adaptive Kalman Filter for AEC was also widely used. The Kalman Filter was originally created in 1960 by Rudolf E. Kalman as a re-examination of the filtering and prediction problem using the Bode-Shannon formalism and state-space representation of dynamic systems. This means that the random signals to be worked with are represented as the output of linear systems excited by white noise, and such linear systems are themselves described by first order difference equations.

III. Mobile-Agent Based Computing Model

The most commonly used computing model is called client/server based, where individual sensors (the clients) send raw data or preprocessed data to a processing center (the server) and data integration is carried out at the center. It's use still resist today for distributed computing too. Some drawbacks are due to this computing model which might prevent it from being used in sensor networks. Firstly, client/server-based computing generally requires many round trips over the network in order to complete one transaction. The network connection needs to be alive and healthy the entire time of the transaction, otherwise the transaction has to restart if it can at all. Secondly, some kind of super-nodes in the sensors network, served as the processing centers, have bigger storage, higher computing capabilities, and more energy. However in some automatic and homogeneous sensor networks this is not always occur being the unreliability and low bandwidth of the wireless link used in sensor networks. The rising demand on processing power and the need to conserve bandwidth on large, slow networks claim for several new approaches appeared but possibly the most interesting among them is the mobile agents paradigm.

MAs are a special case of mobile code, i.e. processes that can move from one host to another and resume execution at the new host without actually restarting. There is no final definition of what is a MA, but attempts to classify autonomous agents led someone to a general definition. The difference between the client/server-based computing and mobile-agent-based computing models are well described in literature. In this latter model, instead of each sensor node sending raw data or preprocessed data to the processing center, the processing code is moved to the data locations through mobile agents. An agents decide their own course of action, within the bounds of the program in the context of software agents. This is the meaning of the sentence: Agents are autonomous.

In this section, we present the basics of the computing model based on the MAs and the following features that respond to the unique challenges posed by the sensor network.

A. Client/server-based computing versus Mobile-agent-based computing

Notable benefits over conventional distributed programming paradigms are provided by MAs. The mobile agent is a special kind of "software". Once it is dispatched, migrate from node to node performing data processing autonomously. The structure of a mobile agent has four attributes: identification, itinerary, data, and processing code. Identification uniquely identifies each mobile agent. Data is the agents data buffer, which carries a partially integrated result. Itinerary is the route of migration. It can be fixed or dynamically determined based on the current network status and the information gain. A processing code carries out the integration whenever the mobile agent arrives at a local sensor node.

While in a client/server-based model, data is the migration unit is in the MAs based model, the migration unit is "mobile agent". Therefore, the agent release results and terminate its itinerary any time the integration accuracy satisfies the requirement and this feature also saves both network bandwidth and computation time since unnecessary node visits and agent migrations are avoided. However, for client/server-based computing, there will be increased queuing delay as the number of clients increases so that result in longer processing delay and more potential drops at the server side. In sensor networks the number of nodes could be also hundreds or even thousands.

The main properties of MAs are:

1) Scalability: The performance of the network is not affected when the number of sensor nodes is increased. Agent architectures that support adaptive network load balancing could do much of a redesign automatically.

2) Reliability: Mobile agents can be sent when the network connection is alive and return results when the connection is restablished. Therefore, the performance of the mobile-agent-based computing model is not affected much by the reliability of the network.

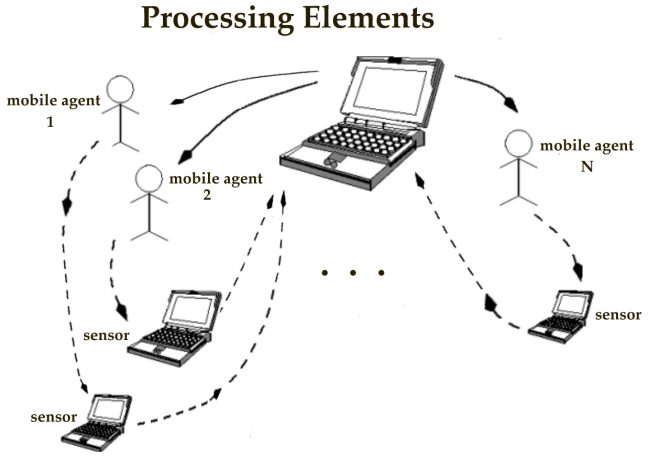


Fig. 3. General Schematic of MAs-RNNs based computing model for N sensors node.

3) Extensibility and task adaptivity: Mobile agents can be programmed to carry different task-specific integration processes which extends the functionality of the network.

4) Energy awareness: The itinerary of the mobile agent is dynamically determined based on both the information gain and energy constraints. It is tightly integrated into the application and is energy efficient.

5) Progressive accuracy: A mobile agent always carries a partially integrated result generated by nodes it already visited. As the mobile agent migrates from node to node, the accuracy of the integrated result is always improved assuming the agent follows the path determined based on the information gain.

The increasing in popularity of MAs led to the development of several programming languages specifically designed. Telescript was perhaps the first and most well known language. Java is currently the number one choice of mobile agents developers. It is Javas characteristics that favour for developing MAs; it is inherently platform independent and a de facto standard in platform independent computing.

IV. The development of the proposed MAs-RNNs and PRNNs based approach for AEC

This section documents the implementation and the simulation results with the devised MAs-RNNs and PRNNs based computing models and MAs were independent objects capable to achieve the AEC tasks. Based on users requests, the MAs start their journey and move autonomously among hosts. Figure 3 show the general schematic for N sensors in this scenario but overall behavior for AECwas considered in the paper.

A. Description

The prediction is performed at the listeners in a distribute manner using MAs-RNNs or PRNNS. The concept of the entire computing model is a distributed system, then the information is sent to the server using MAs based computing

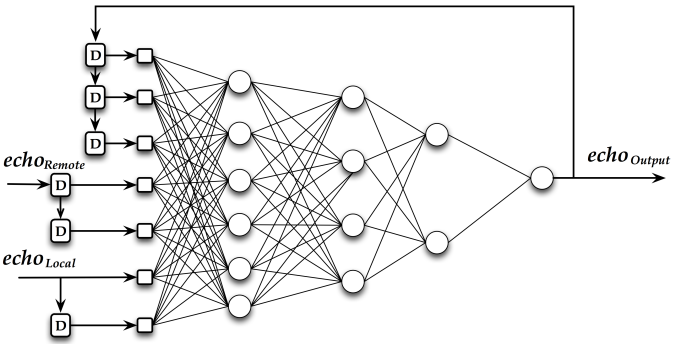


Fig. 4. The selected RNN as a MA.

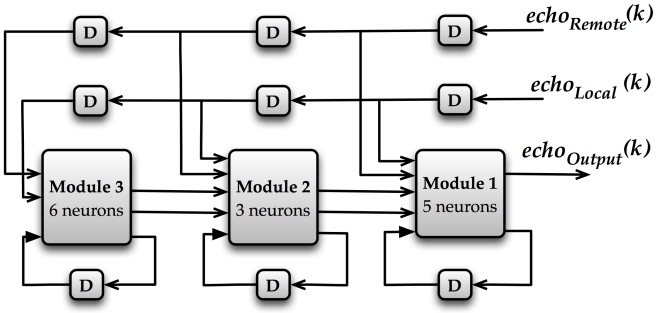


Fig. 5. The selected PRNN as a MA for improved performance in AEC.

methods.

However during the implementation of this computing model a few problems became apparent mainly due to the platforms to be adopted, but the use of an MAs-RNNs approach avoids the coarse approximation sometimes joined to imprecise information on the voice, echoes signals.

To our problem now, i.e. as to implement RNNs in the MAs computer hybrid model. Initially for RNNs implementation we used RNS (Recurrent Network Simulator) which is a simulator for recurrent neural networks.

RNSs features include:freely choosable connections, no restrictions besides memory or CPU constraints delayed links for recurrent networks, fixed values or thresholds can be specified for weights(recurrent) back-propagation, Hebb, differential Hebb, simulated annealing and more, patterns can be specified with bits, floats, characters, numbers, and random bit patterns with Hamming distances can be chosen for your user definable error functions, output results can be used without modification as input. However we mention as is wanted the use of RNNs as a guide to the agents, and then Repast (Recursive Porous Agent Simulation Toolkit) is a free and open source agent modeling toolkit under continual development by Argonne. It can be thought of as a specification for agent-based modeling services or functions. It provides an integrated set of libraries for neural networks so as genetic algorithms and other topics. Finally we favour the use of C++ for all implementedMAs in the current our application that will be shown automatically on the display. The system is started by creating a new agent of the Mobile Server and once created, the server shows the network set-up window.

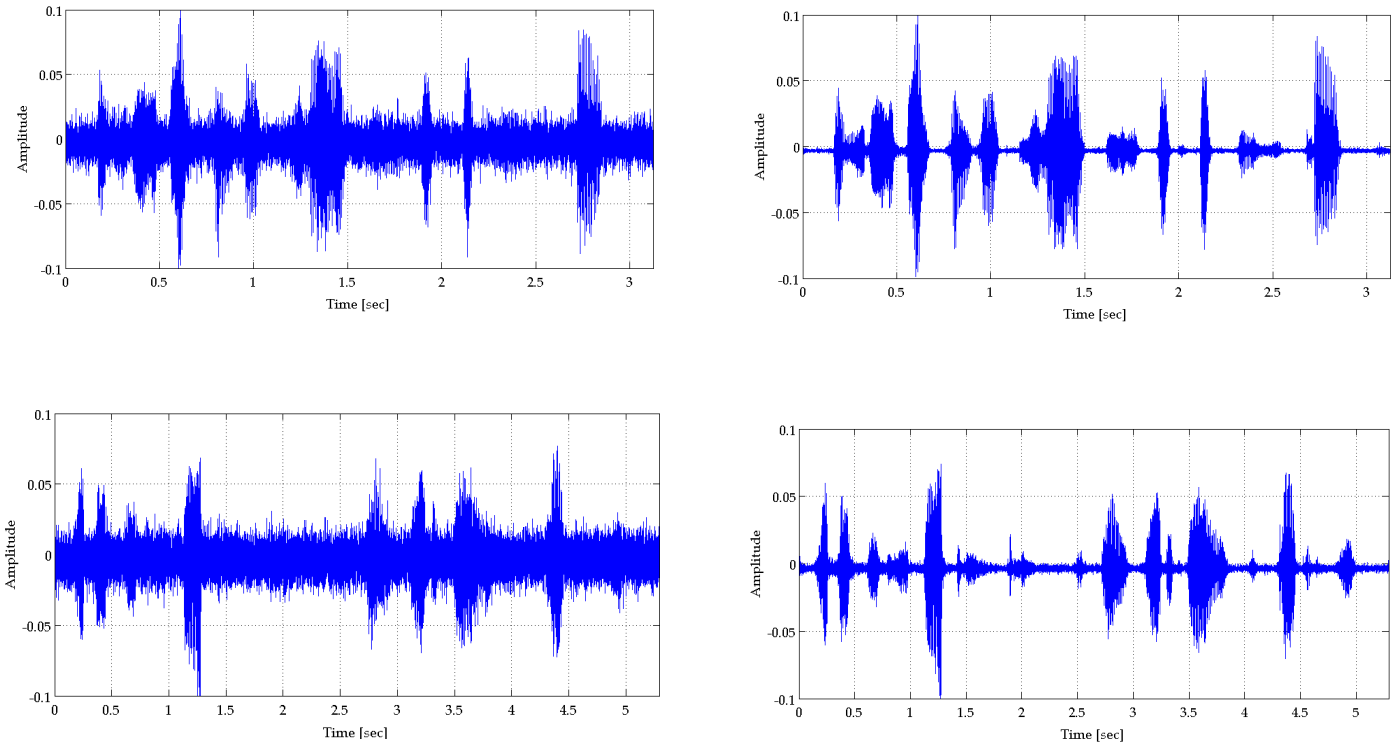


Fig. 6. Noisy signal on left and the signal after echo reduction on the right.

V. RESULTS

The authors have initially gained experiences by the use of MAs-RNNs and these were designed as shown in Fig. 4, but however in the present work better MAs-PRNNs based implementation and results are reported being several the advantages raised by their accurate implementation. An hybrid kind of neural network is the so called PRNN introduced by Haykin and Li in 1995. It consists of a modular nested structure of small scale fully connected RNN and a cascaded of FIR adaptive filter.

In the PRNN configuration the single module connection is well described and depicted in literature, joined with relative mathematical equations. PRNN is a modular network made of a number of RNNs as its modules with each module consisting of some neurons and the selected network for the present echo cancellation application consists of three modular layers, with relative neuroprocessing units, as shown in Fig. 5. For both MAs with RNNs, or PRNNs, the number of layers, modular layers and neurons were defined by extensive simulation tests in the corresponding implemented recurrent neural models.

The implemented MAs with RNNs, or PRNNs, have been trained, tested using several speech voice, echoes data samples collected from public speeches of different people and voices. This latter kind of network shows a better prediction capability of non linear behavior data and to model complex phenomena coming from several different physical contributes due to the modular structure which provides analysis and understanding of the basic contribution to reconstruct the overall behavior.

The signals processing is relative to signals registered at the University of Catania coming from the microphone array. For testing the MAs-PRNNs based computing algorithms a database of signals was available with a sampling rate of 8 kHz. They are divided in frames of ten thousand samples (corresponding to an analysis frame-interval of 3-5.3 ms).

In Fig. 2 is shown as the signal is corrupted by many types of noise, as the reflection in the room, or occurring in an environment, as shown in Fig. 2. For this reason the proposed approach should be highly reliable in presence of these noisy signals. The experiments show as the average echo reduction echo removing is about in the range 40 dB and 30 dB in a widely considered signals relative for the main speaker and at several SNR ratios, for a considered signal voices database registered at the University of Catania. By addressing the AEC problem by this approach, the time is not considered as a second dimension of the input space, but it is implicitly coded in the structure of a recurrent network topology or modular RNN.

The MAs-PRNNs based echoes output prediction is so obtained, but here the result is reported in Fig. 6 only for a node sensor. Satisfactory performance of echo cancellation were obtained also for the other nodes. The two echo signals input, as echo remote and echo local, so as the echo output prediction are shown in Fig. 6. This echo output prediction for the selected node is relative to the best result provided by MAs-PRNNs implementation.

A selective management of the temporal memory is also possible thanks to their computational flexibility. The individual weights of the three modules in Fig. 5 are adjusted in independent manner. Moreover PRNNs show a stronger time memory with respect to a standard RNNs with back propagation through time training algorithm. A large amount of data were processed in order to find a RNN and PRNN structure able to reproduce the behavior of a MAs soft-computing based echo cancellation solutions. The MAs based echoes output prediction in order to evaluate the echo-cancellation is so obtained. The result were calculated for the overall system

keeping in account the main speaker.

Satisfactory performance in echo cancellation were obtained. The two echo signals input, as echo remote and echo local are provided as inputs to the RNN and PRNN and the echo output prediction shows a relevant reduction evaluated as 60 db. This echo output prediction is relative to the best result coming from MAs-PRNNs implementation.

VI. Conclusions

Several applications benefit already by the use of mobile agents technology and we now want to use a soft computing technology as RNNs and PRNNs for this problem area in communication systems. In this paper we propose a MAs with RNNs-PRNNs-based computing model for AECs problem in communication systems.

Prediction of echo paths can be performed by RNNs based processing. The results about the faced problem in communication systems, where the use of MAs implementation with RNNs might be an improvement over a more conventional solutions, are here summarized, presented and evaluated.

As mentioned we do not know any satisfactory solution to the echo cancellation problems so basically in this paper a computing model and algorithm is presented for RNNs based adaptive filtering with application to AEC. The basics of the implementation are introduced so as the obtained results.

The experiences with the results, gained on the echo cancellation problems, by the implemented MAs-PRNNs show better performance versus MAs- RNNs computing model. However the performance of this approach can not be compared versus more conventional MAs applications or distributed computing system in AEC problems being a novel implementation in this field but the determination if rising any advantage of using other platform for the MAs with RNNS or PRNNs has to be investigated.

References

[1] M. Sondhi, D.R. Morgan, J.L. Hall, "Stereophonic acoustic echo cancellation-an overview of the fundamental problem," *IEEE Signal Processing Letters,* Vol. 2, no. 8, pp. 148-151 (1995).

[2] Haiquan Zhao, Xiangping Zeng, Zhengyou He, "Low-Complexity Nonlinear Adaptive Filter Based on a Pipelined Bilinear Recurrent Neural Network," *IEEE Transactions on Neural Networks,* vol.22, no. 9, pp.1494-1507, Sept. 2011.

[3] S. Haykin, "Adaptive Filter Theory," Englewood Cliffs, Prentice-Hall, New York (2001).

[4] R.H. Glitho, E. Olougouna, Pierre, Samuel, "Mobile agents and their use for information retrieval: a brief overview and an elaborate case study," *IEEE Network,* vol.16, no.1, pp.34-41, Jan/Feb 2002.

[5] L. Ismail and D. Hagimont "A Performance Evaluation of the Mobile Agent Paradigm" *ACM SIGPLAN NOTICES,* pp. 306–313, ACM Press (1999).

[6] S. Haykin, "Neural Networks and Learning Machines," Englewood Cliffs, Prentice-Hall, New York (2008).

[7] F. Bonanno, G. Capizzi, C. Napoli, "Some remarks on the application of RNN and PRNN for the charge-discharge simulation of advanced Lithium-ions battery energy storage" *2012 IEEE International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)* pp. 941-945, 20-22 June 2012.

[8] G. Capizzi, F. Bonanno, C. Napoli, "Recurrent neural network-based control strategy for battery energy storage in generation systems with intermittent renewable energy sources," *2011IEEE International Conference on Clean Electrical Power (ICCEP)* pp. 336-340, 14-16 June 2011.

[9] G. Capizzi, F. Bonanno, C. Napoli, "Hybrid neural networks architectures for SOC and voltage prediction of new generation batteries storage," *2011IEEE International Conference on Clean Electrical Power (ICCEP)* pp. 341-344, 14-16 June 2011.

[10] F. Bonanno, G. Capizzi, S. Coco, C. Napoli, A. Laudani, and G. L. Sciuto, "Optimal thicknesses determination in a multilayer structure to improve the SPP efficiency for photovoltaic devices by an hybrid FEM Cascade Neural Network based approach," *2014 IEEE International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)* pp. 355-362, 18-20 June 2014.

[11] F. Bonanno, G. Capizzi, G. L. Sciuto, C. Napoli, G. Pappalardo, E. Tramontana, "A novel cloud-distributed toolbox for optimal energy dispatch management from renewables in IGSs by using WRNN predictors and GPU parallel solutions," *2014 IEEE International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)* pp. 1077-1084, 18-20 June 2014.

[12] C. Napoli, F. Bonanno, G. Capizzi, "An Hybrid Neuro-Wavelet Approach for Long-Term Prediction of Solar Wind", *IAU Symposium No. 274, Advances in Plasma Astrophysics,* pp. 247-249, Giardini Naxos, Italy (2010).

[13] C. Napoli, F. Bonanno, G. Capizzi, "Exploiting Solar Wind Time Series Correlation with Magnetospheric Response by Using an Hybrid Neuro-Wavelet Approach", *IAU Symposium No. 274, Advances in Plasma Astrophysics,* pp. 250-252, Giardini Naxos, Italy (2010).

[14] G. Capizzi, F. Bonanno, and C. Napoli, "Hybrid neural networks architectures for soc and voltage prediction of new generation batteries storage, in Proc. *IEEE, International Conference on Clean Electrical Power (ICCEP)*, pp. 341344, 2011.

[15] G. Borowik, M. Woźniak, A. Fornaia, R. Giunta, C. Napoli, G. Pappalardo, E. Tramontana, "A software architecture assisting workflow executions on cloud resources", *International Journal of Electronics and Telecommunications*, vol. 61, no.1, pp. 17-23, 2015.

[16] G. Capizzi, F. Bonanno, C. Napoli, "A new approach for lead-acid batteries modeling by local cosine" in Proc. *IEEE Symposium on International Power Electronics Electrical Drives Automation and Motion (SPEEDAM),* pp.1074-1079, 2010.

[17] F. Bonanno, G. Capizzi, A. Gagliano, C. Napoli, "Optimal management of various renewable energy sources by a new forecasting method",in Proc. *International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM),* pp. 934-940, 2012.

[18] F. Bonanno, G. Capizzi, G. Lo Sciuto, C. Napoli, G. Pappalardo, E. Tramontana, "A cascade neural network architecture investigating surface plasmon polaritons propagation for thin metals in openmp," *Springer International Publishing*, pp. 22-33, 2014.

[19] C. Napoli, G. Pappalardo, E. Tramontana, Z. Marszałek, D.Połap, M. Woźniak, "Simplified firefly algorithm for 2D image key-points search", in Proc. *IEEE Symposium on Computational Intelligence for Human-like Intelligence,* pp.118-125, 2014.

Adaptive Hybrid Agents for Tactical Decisions in Pedestrian Environments

Luca Crociani, Andrea Piazzoni, Giuseppe Vizzari
CSAI - Complex Systems & Artificial Intelligence Research Center,
University of Milano-Bicocca, Milano, Italy
{luca.crociani,giuseppe.vizzari}@disco.unimib.it
a.piazzoni@campus.unimib.it

***Abstract*—This paper presents a hybrid agent architecture for modeling different types of decisions in a pedestrian simulation system. In particular, the present work focuses on *tactical level* decisions that are essentially related to the choice of a route to follow in an environment comprising several rooms connected by openings. These decisions are then enacted at the operational level by mean of a floor-field based model, in a discrete simulation approach. The described model allows the agent taking decisions based on a static a-priori knowledge of the environment and dynamic perceivable information on the current level of crowdedness of visible path alternatives. The paper presents the model formally, motivating the adopted choices with reference to the relevant state of the art. The model is also experimented in benchmark scenarios showing the adequacy in providing adaptiveness to the contextual situation.**

***Index Terms*—Pedestrian Simulation, Tactical Level, Hybrid Agents.**

I. Introduction

The simulation of complex systems is one of the most successful areas of application of agent-based approaches: even though models, mechanisms and technologies adopted by researchers in different disciplines are not necessarily up-to-date or in line with the most current results in the computer science and engineering area about agent technologies [1], the area still presents interesting challenges and potential developments for agent research.

The simulation of pedestrians and crowds is an example of already established yet lively research context: both the automated analysis and the synthesis of pedestrian and crowd behaviour, as well as attempts to integrate these complementary and activities [2], present open challenges and potential developments in a smart environment perspective [3].

Even if we only consider choices and actions related to walking, modelling human decision making activities and actions is a complicated issue: different types of decisions are taken at different levels of abstraction, from path planning to the regulation of distance from other pedestrians and obstacles present in the environment. Moreover, the measure of success and validity of a model is definitely not the *optimality* with respect to some cost function, as in robotics, but the *plausibility*, the adherence to data that can be acquired by means of observations or experiments. Putting together *tactical* and *operational* level decisions, often adopting different approaches (typically behaviour-based for operational decisions, and at knowledge level for tactical ones) in a comprehensive framework preserving and extending the validity that, thanks to recent extensive observations and analyses (see, e.g., [4]), can be achieved at the operational level represents an urgent and significant open challenge.

This paper presents a hybrid agent architecture for modeling different types of decisions in a pedestrian simulation system. In particular, the present work focuses on *tactical level* decisions that are essentially related to the choice of a route to follow in an environment comprising several rooms connected by openings. These decisions are then enacted at the operational level by mean of a floor-field based model, in a discrete simulation approach. The described model that integrates within an organised and comprehensive framework different spatial representations, types of knowledge and decision making mechanisms, allows the agent taking decisions based on a static a-priori knowledge of the environment and dynamic perceivable information on the current level of crowdedness of visible path alternatives.

The paper presents the relevant state of the art in the following Section. The tactical level part of the model is formally presented in Section III-B whereas its experimental application in benchmark scenarios showing the adequacy in providing adaptiveness to the contextual situation is given in Section IV.

II. Related Works

The research on pedestrian dynamics is basically growing on two lines. On the *analysis* side, literature is producing methods for an automatic extraction of pedestrian trajectories (e.g. [5], [4]), charactertization of pedestrian flows (e.g. [6]) automatic recognition of pedestrian groups [7], recently gaining importance due to differences in trajectories, walking speeds and space utilization [8]. The *synthesis* side – where the contributions of this work are concentrated – has been even more prolific, starting from preliminary studies and assumptions provided by [9] or [10] and leading to quite complex, yet not usually validated, models exploring components like panic [11] or other emotional variables. To better understand the model presented in the next section, the following will provide a brief description of related works on pedestrian dynamics modeling and simulation.

[12][1] provides a well-known scheme to model the pedestrian dynamics, describing 3 levels of behavior:

- **Strategic level**: the person formulates his/her abstract plan and final objective motivating the overall decision to move (e.g. "I am going to the University today to follow my courses and meet my friend Paul");
- **Tactical level**: the set of activities to complete the plan is computed and scheduled (e.g. "I am going to take the 8:00 AM train from station XYZ then walk to the Department, then meet Paul at the cafeteria after courses, then ..."");
- **Operational level**: each activity is physically executed, i.e., the person perform the movement from his/her position to the current destination (e.g. precise walking trajectory and timing, such as a sequence of occupied cells and related turn in a discrete spatial representation and simulation).

Most of the literature has been focussed on the reproduction of the physics of the system, so on the lowest level, where a significant knowledge on the fundamental diagram achieved with different set of experiments and in different environment settings (see, e.g, [14], [15]) allows a robust validation of the models.

Literature of this level can be classified regarding the scope of the modeling approach. Macroscopic models describe the earliest approach to pedestrian modeling, basing on analogies between behavior of dense crowds and kinetic gas [9] or fluids [10], but essentially abstracting the concept of individual. A microscopic approach is instead focused on modeling the individual behavior, effectively improving the simulations precision also in low density situations.

The microscopic approach is as well categorized in two classes describing the representation of space and movement: continuous models simulate the dynamics by means of a force-based approach, which finds its basis on the well-known social force model by [16]. These models design pedestrians as particles moved by virtual forces, that drive them towards their destination and let them avoid obstacles or other pedestrians. Latest models on this class are the centrifugal force model by [17] and the stride length adaptation model by [18]. Other examples consider also groups of pedestrians by means of attractive forces among person inside the group [19].

The usage of a discrete environment is mostly employed by the cellular automata (CA) based models, and describes a less precise approach in the reproduction of individuals trajectories that, on the other side, is significantly more efficient and still able to reproduce realistic aggregated data. This class derives from vehicular modeling and some models are direct adaptations of traffic ones, describing the dynamics with ad hoc rules (e.g. [20], [21]). Other models employs the well-know *floor field* approach from [22], where a *static* floor field drives pedestrians towards a destination and a *dynamic* floor field is used to generate a lane formation effect in bi-directional flow. [23] is an extension of the floor field model, introducing the *anticipation* floor field used to manage crossing trajectories and encourage the lane formation. [24] discussed methods to deal with different speeds, in addition to the usage of a finer grid discretization that decreases the error in the reproduction of the environment, but significantly impacts on the efficiency of the model. An alternative approach to represent different speeds in a discrete space is given by [25]. [26] is another extension of the floor-field model, where groups of pedestrians are also considered.

The tactical level has gained interest only recently in the literature of pedestrian dynamics modeling and simulation, despite its relevance for the simulation of a realistic behavior (especially by thinking to evacuation situations). Path planning algorithms have been widely investigated and proposed in the field of computer graphics and gaming by means of graph-based methods (e.g. [27], [28]), but with aims not necessarily matching the requirements of pedestrian simulation, since the point is mainly to reach a visual realism.

On the pedestrian dynamics side, a relevant recent work is the one from [29], mainly dealing with tactical level decisions during evacuation, providing an approach to find the *quickest* path towards the exit, i.e. the way that implies the fewest time. The approach is tested on the basis of four strategies for the route choice management, given by the combination of applying the shortest or quickest path, with a local (i.e., to go out from the room) or global (i.e., to reach the destination) strategy. The global shortest path is calculated with the well-known Floyd-Warshall algorithm, implying therefore a computational time that can become an issue by having a large number of nodes or by considering special features in the simulated population (i.e. portion of the path where the cost differs from an agent to another). The work in this paper will propose an alternative and efficient approach to find a global path, where each agent will be able to consider additional costs in sub-paths without adding particular costs to the computation.

III. A Model for Tactical Level of Pedestrians

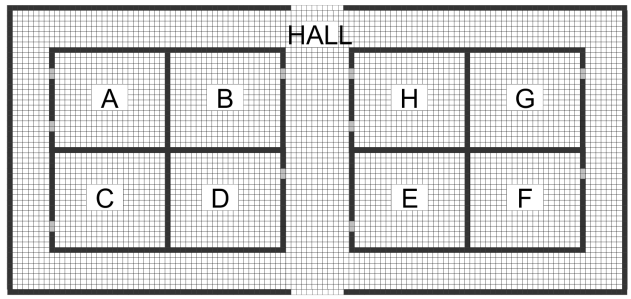
The model described in this work provides a methodology to deal with tactical choices of agents in pedestrian simulation systems. Due to constraints on the length of the paper, the description of the part of the model dedicated to the operational level, thoroughly described in [30], will not be provided.

A. A Cognitive Representation of the Environment for Static Tactical Choices

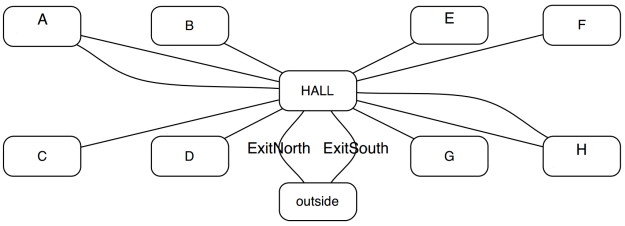
The framework that enables agents to perform choices on their plan implies a graph-like, topological, representation of the walkable space, whose calculation is defined in [31] and briefly reported in this section. This model allows agents to perform a static path planning, since dynamical information such as congestion is not considered in the graph. These additional elements will be considered in the extension that is presented in the next section and represent the innovative part of this paper.

The environment abstraction identifies *regions* (e.g. a room) as node of the labeled graph and *openings* (e.g. a door) as edges. This so-called *cognitive map* is computed starting from the information of the simulation scenario, provided by the

[1] A similar classification can be found in vehicular traffic modeling from [13].



(a)



(b)

Fig. 1. An example environment (a) with the resulting cognitive map (b), by applying the procedure from [31].

user and necessarily containing: (i) the description of the walkable space, that is, the size of the simulated environment and the positions of obstacles and walls; (ii) the position of final destinations (i.e. exits) and intermediate targets (e.g. a ticket machine); borders of the logical regions of the environment that, together with the obstacles, will define them. Approaches to automatically configure a graph representation of the space, without any additional information by the user, have been already proposed in the literature (e.g. [32]), but they are not leading to a cognitively logical description, i.e., a *topological* map. A cognitive definition of the space allows, in fact, a proper definition of portions of the environment where, for example, the behavior of a person is systematically different (e.g. the change of speed profile in stairs or ramps), or that contain relevant intermediate targets for the agent plan (e.g. a ticket machine).

The cognitive map is defined as a graph $\mathcal{CM} = (\mathcal{V}, \mathcal{E})$ generated with a procedure included to the floor field diffusion, starting from the statements that each user-defined opening generates a floor field from its cells and spread only in the regions that it connects, and that each region has a flag indicating its properties among its cells. The floor fields diffusion procedure iteratively adds to $\mathcal{CM}$ the couple of nodes found in the diffusion (duplicates are avoided) and labeled with the region id and the edge labeled with the id of the opening. Each *final destination*, different from the normal openings since it resides in only one region, will compose an edge linking the region to a special node describing the external *universe*. Intermediate targets will be mapped as attributes of its region node. An example of environment together with the resulting cognitive map is presented in Fig. 1.

To allow the calculation of the *paths tree*, that will be described in the following section, functions $Op(\rho)$ and $Dist(\omega_1, \omega_2)$ are introduced describing respectively: the set of openings accessible from the region ρ[2] and the distance between two openings linking the same arbitrary region. While the first one is trivial and outputs the edges linking ρ, the function $Dist(\omega_1, \omega_2)$ describes the distance that will be perceived by agents for their path planning calculation. To obtain a scalar from the sets of cells associated to ω_1 and ω_2, the value of the floor field in their center cell is used, defined as:

$$Center(\omega) = \left(\left\lfloor \frac{\sum x_i}{|\omega|} \right\rfloor, \left\lfloor \frac{\sum y_i}{|\omega|} \right\rfloor \right), (x_i, y_i) \in \omega.$$

The distance between ω_1 and ω_2 is then calculated as the average between the floor field values in the two center cells, i.e., the value of the floor field of ω_1 in $Center(\omega_2)$ and vice-versa.

B. Modeling Adaptive Tactical Decisions with A Paths Tree

To enhance the route choice and enable dynamical, adaptive, decisions of the agents in a efficient way, a new data structure has been introduced, containing information about the cost of *plausible* paths towards the exit from each region of the scenario.

Using the well-known Floyd-Warshall algorithm, in fact, can solve the problem but introduces issues in computational time: the introduction of dynamical elements in the paths cost computation (i.e. congested paths) implies a re-computation of the cost matrix underlying the algorithm every step. More in details, the penalty of a congested path is a subjective element for the agents, since they are walking with different desired velocities, thus the calculation cost increases also with the number of agents.

The approach here proposed implies an off-line calculation of the data-structure that we called *paths tree*, but is computationally efficient during the simulation and provides to the agents direct information about the travel times describing each path. The method is described in the following paragraphs.

1) The Paths Tree: We define the *Paths Tree* as a tree data-structure containing the set of "rational" paths towards a destination, that will be its root. Before describing what we mean with the attribute rational, that can be seen as a fuzzy concept, a general definition of path must be provided.

A path is defined as a finite sequence of openings $X \rightarrow Y \rightarrow \ldots \rightarrow Z$ where the last element represents the final destination. It is easy to understand that not every sequence of openings represents a path that is walkable by an agent. Firstly, a path must be a sequence of consecutive oriented openings regarding the physical space.

Definition III.1 (Oriented opening)**.** Let $E = R_1, R_2$ be a opening linking the regions R_1 and R_2, (R_1, E, R_2) and (R_2, E, R_1) define the oriented representations of E.

An oriented opening will therefore describe a path from an arbitrary position of the first region towards the second one.

[2]Its Id, described in the label of the edge mapped to it.

Definition III.2 (Valid path)**.** let C a sequence of oriented openings $X \rightarrow \ldots \rightarrow Z$. C is a valid path if and only if:

- $|C| = 1$
- $|C| = 2$: by assuming $C = X- > Y$, the third element of the triple X must be equal to the first element of Y
- $|C| > 2$: each sub-sequence S of consecutive openings in C where $|S| = 2$ must be a valid path.

The last oriented opening in the path leads to the *universe* region.

Given a set of paths, the agent will consider only complete paths towards its goal, starting from the region where the agent is located.

Definition III.3 (Start and Destination of a path)**.** Given p a path $(R_1, E, R_x) \rightarrow ... \rightarrow (R_y, O, \text{universe})$, the function $RS(p) = R_1$ will return the region R_1 where an agent can start the path p. $S(p) = E$ and $D(p) = p$ will respectively return the first opening (E) and the destination (O) of the path.

Definition III.4. Let p a path, T(p) is the function which return the expected travel time from the first opening to the destination.

$$T(p) = \sum_{i \in [1,|p|-1]} \frac{Dist(opening_i, opening_{i+1})}{speed} \quad (1)$$

Another basic rule is that a path must be loop-free: by assuming the aim to minimize the time to reach the destination, a plan passing through a certain opening more than once would be not rational.

Definition III.5 (opening loop constraint)**.** A path $X \rightarrow \ldots \rightarrow Z$ must not contain duplicated openings.

This will not imply that an agent cannot go through a certain opening more than once during the simulation, but this will happen only with a change of the agent plan.

By assuming to have only convex regions in the simulated space, we could easily achieve the set of rational paths by extending III.5 as to let a path not containing duplicated regions. However, since the definition of region describes also rooms, concave regions must be considered. Some paths may, thus, imply to pass through another region and then return to the first one to reduce the length of the path.

As we can see by the Figure 2 both paths starts from r_1, go through r_2, and then return to r_1. However, only the path represented by the continuous line is rational, even if the constraint III.5 is respected by both of them. Before the definition of the constraint that identifies the correct paths, the concept of *sub-path* has to be defined.

Definition III.6 (Sub-path)**.** Let p a path, a sub-path p' of p is a sub-sequence of oriented openings denoted as $p' \subset P$ which respects the order of appearance for the openings in p, but the orientation of openings in p' can differ from the orientation in p. p' must be a valid path.

The reason of the orientation change can be explained with the example in Fig. 3: given the path $p = (r1, o2, r2) \rightarrow (r2, o1, r1) \rightarrow end$, the path $p' = (r2, o2, r1) \rightarrow end$ is a valid path and is considered as a sub-path of p, with

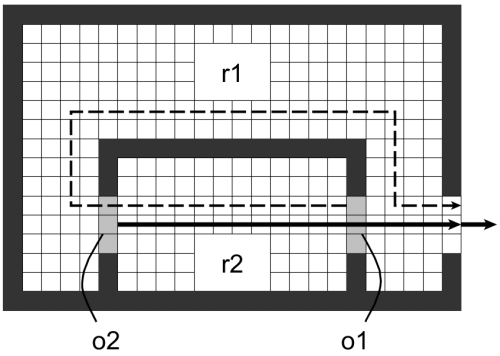


Fig. 2. A concave region can imply the plausibility of a path crossing it twice, but its identification is not elementary.

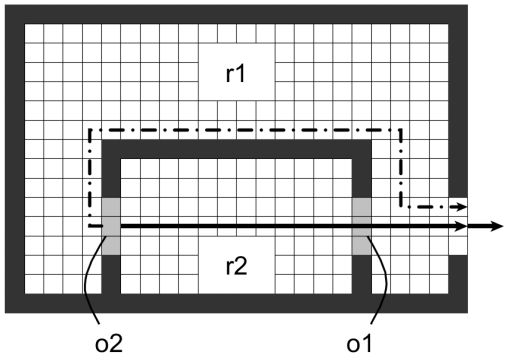


Fig. 3. The correct paths for this environment. Inside $r2$ the choice between the two openings is also determined by the congestion.

a different orientation of $o2$. In addition, given the path $p_1 = (r_2, o_2, r_1) \rightarrow end$, the path $p_2 = (r_1, o_2, r_2) \rightarrow (r_2, o_1, r_1) \rightarrow end$ is as well a minimal path if and only if the travel time of p_2 is less than p_1. It is easy to understand that this situation can emerge only if r_1 is concave. As we can see, the starting region of the two paths is different, but the key element of the rule is the position of the opening o_2. If this rule is verified in the center position of the opening o_2, this path will be a considerable path by the agents surrounding $o2$ in $r1$.

In Figure 3 the correct paths for this example environment are shown. An agent located in $r2$ can reach $r1$ and then the destination D using both of the opening considering the congestions. An agent located in $r1$ can go directly to the exit or chose the path $o2 \rightarrow o1 \rightarrow D$.

Definition III.7 (Minimal path)**.** p is a minimal path if and only if it is a valid path and $\forall p' \subset p : S(p') = S(p) \wedge D(p') = D(p) \implies T(p') > T(p)$

The verification of this rule is a sufficient condition for the opening loop constraint III.5 and solve the problem on the region loop constraint independently from the configuration of the environment (i.e. convex or concave regions).

At this point the constraint that defines a minimal path has been provided. This can be used to build the complete set of minimal paths towards a destination before running the

simulation. It must be noted that an arbitrary path represents a set of paths itself, since it can be undertaken at any region it crosses. Indeed every path p provides also information about the sub-paths achieved by cutting the head of p with an arbitrary number of elements. So a minimal representation of the set is a tree-like structure defined as:

Definition III.8 (Paths tree)**.** Given a set of minimal paths towards a destination, the Paths-Tree is a tree where the root represents the final destination and a branch from every node to the root describes a minimal path, crossing a set of openings (other nodes) and region (edges). Each node has an attribute describing the expected travel time to the destination.

2) An Algorithm to Compute the Paths Tree: The algorithm we are proposing build the the Paths Tree in a recursive way, starting from a path containing only the destination and adding nodes if and only if the generated path respects the definition of minimality.

Formally the Paths Tree is defined as $PT = (N, E)$ where N is the set of nodes and E the set of edges. Each node n is defined as a triple (id, o, τ) where:

- $id \in \mathbb{N}$ is the id of the node
- $o \in \mathcal{O}$ is the name of the opening
- $\tau \in \mathbb{R}^+$ is the expected travel time for the path described by the branch.

Each edge e is defined as a triple (p, c, r) where:

- $p \in \mathcal{O}$ is the id of the parent
- $c \in \mathcal{O}$ is the id of the child
- $r \in \mathcal{R}$ is the region connecting the child node to its parent.

To allow a fast access to the nodes describing a path that can be undertaken from a certain region, we added a structure called M that maps each r in the list of $p : (p, c, r) \in E$ (for every c).

Given a destination $D = (r_x, \text{universe})$, the paths tree computation is defined with the following procedures.

Algorithm 1 Paths tree computation

1: add $(0, D, 0)$ to N
2: add 0 to $M[r_x]$
3: $\forall s \in \mathcal{O}$ ShortestPath[s] $\leftarrow \infty$
4: expand region$(0, D, 0, R_x, ShortestPath)$

With the first line, the set N of nodes is initialized with the destination of all paths in the tree, marking it with the id 0 and expected travel time 0. In the third row the set of $ShortestPath$ is initialized. This will be used to track, for each branch, the expected travel time for the shortest sub-path, given a start opening s. $ExpandRegion$ is the core element of the algorithm, describing the recursive function which adds new nodes and verifies the condition of minimality. The procedure is described by Alg. 2.

In line 2 a list of openings candidates is computed, containing possible extensions of the path represented by $parentId$. Selecting all the openings present in this region (except for the one labeled as $parentName$) will ensure that all paths eventually created respect the validity constraint III.2.

Algorithm 2 ExpandRegion

Require: input parameters $(parentId,\ parentName,\ parentTime, RegionToExpand, ShortestPath)$
1: $expandList \leftarrow \emptyset$
2: $opList = Op(RegionToExpand) \setminus parentName$
3: **for** $o \in opList$ **do**
4: $\tau = parentTime + \frac{D(o, parentName)}{speed}$
5: **if** $CheckMinimality(ShortestPath, o, \tau)$ == True **then**
6: add (id, o, τ) to N
7: add $(parentId, id, r)$ to E
8: $ShortestPath[o] \leftarrow \tau$
9: $nextRegion = o \setminus r$
10: add id to $M[nextRegion]$
11: add $(id, o, \tau, nextRegion)$ to $expandList$
12: **end if**
13: **end for**
14: **for** $el \in expandList$ **do**
15: $ExpandRegion(el, ShortestPath)$
16: **end for**

At this point, the minimality constraint III.7 has to be verified for each candidate, by means of the function $CheckMinimality$ explained by Alg. 3. Since this test requires the expected travel time of the new path, this has to be computed before. A failure in this test means that the examined path – created by adding a child to the node $parentId$ – will not be minimal. Otherwise, the opening can be added and the extension procedure can recursively continue.

In line 6, id is a new and unique value to identify the node, which represents a path starting from the opening o and with expected travel time τ; line 7 is the creation of the edge from the parent to the new node. In line 8, $ShortestPath[o]$ is updated with the new discovered value τ. in line 9 the opening is examined as a couple of region, selecting the one not considered now. In fact, the element $nextRegion$ represents the region where is possible to undertake the new path. In line 10 the id of the starting opening is added to $M[nextRegion]$, i.e., the list of the paths which can be undertaken from $nextRegion$. In line 11 the node with his parameter is added to the list of the next expansions, which take place in line 13-14. This passage has to be done to ensure the correct update of $ShortestPath$.

Algorithm 3 CheckMinimality

Require: input parameter $(ShortestPath, o, \tau)$
1: **if** $ShortestPath[o] > \tau$ **then**
2: **return** True
3: **else**
4: **return** False
5: **end if**

To understand how the constraint of minimality is verified, two basical concepts of the procedure need to be clarified. Firstly, the tree describes a set of paths towards a unique destination, therefore given an arbitrary node n, the path described by the parent of n is a subpath with a different

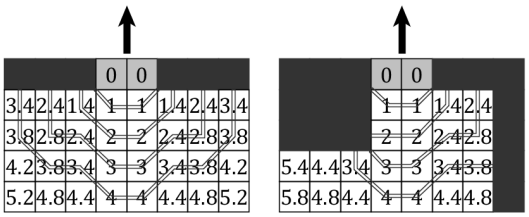


Fig. 4. Examples of surroundings of different sizes, for two configurations of the environment.

starting node and leading to the same destination. Furthermore, the expansion procedure implies that once reached a node of depth l, all the nodes of its path having depth $l-k$, $k>0$ have been already expanded with all child nodes generating other minimal paths.

Note that the variable $ShortestPath$ is particularly important since, given p the current path in evaluation, it describes the minimum expected travel time to reach the destination (i.e. the root of the tree) from each opening already evaluated in previous expansions of the branch. Thus, if τ is less than $ShortestPath[o]$, the minimality constraint III.7 is respected.

3) Congestion Evaluation: The explained approach of the paths tree provides information on travel times implied by each path towards a destination. By only using this information, the choice of the agents will be still static, essentially describing the shortest path. This may also lead to an increase of the experienced travel times, since congestion may emerge without being considered.

For the evaluation of congestion, we provide an approach that estimates, for each agent, the additional time deriving by passing through a jam. The calculation considers two main aspects: the size of the eventually arisen congestion around an opening; the average speed of the agents inside the congested area. Since the measurement of the average speed depends on the underlying model that describes the physical space and movement of the agents, we avoid to explain this component with full details, by only saying that the speed is estimated with an additional grid counting the *blocks* (i.e. when agents maintain positions at the end of the step) in the surrounding area of each opening. The average number of blocks defines the probability to move into the area per step, thus the speed of the agents inside. For the size of the area, our approach is to define a minimum radius of the area and (i) increase it when the average speed becomes too low or (ii) reduce it when it returns normal.

As we can see in Figure 4, the presence of an obstacle in the room is well managed by using floor field while defining the area for a given radius. If a lot of agents try to go through the same opening at the same time, a congestion will arise, reducing the average speed and letting the area to increase its size. When this one becomes too big and the farthest agents inside are not slowed by the congestion, the average speed will start increasing until the area re-sizing will stop.

During this measurement the average speed value for each radius is stored. Values for sizes smaller than the size of the area will be used by the agents inside it, as will be explained in the next section. Two function are introduced for the calculation:

- $size(o)$: return the size of the congestion around the opening
- $averageSpeed(o,s)$: return the average speed estimated in the area of size s around the opening o.

4) Agents Dynamical Path Choice: At this point we have defined which information an agent will use to make its decision:

- the Paths Tree, computed before the simulation, will be used as a list of possible path choice;
- the position of the agent, will be used to adjust the expected travel time considering the distance between the agent and the first opening of a path ($d(a,o)$);
- the information about congestion around each opening, computed during the simulation, will be used to estimate the delay introduced by each opening in the path.

The agent, who knows in which region R_x he is located, can access the Paths Tree using the structure $M[R_x]$. The structure returns a list of nodes, each representing the starting opening for each path. At this point the agent can compute the expected travel time to reach each starting opening and add it to the travel time τ of the path.

To consider congestion, the agent has to estimate the delay introduced by each opening in a path, by firstly obtaining the size of the jammed area.

$$size_a(o) = \begin{cases} size(o) & \text{if } d(a,o) \geq i(x) \\ d(a,o) & \text{otherwise} \end{cases} \tag{2}$$

At this point, the agent can suppose that for the length of the area it will travel at the average speed around the opening.

$$delay(o) = \frac{size_a(o)}{averageSpeed(o)} - \frac{size_a(o)}{speed_a} \tag{3}$$

If the agent is slower than the average speed around an opening, the delay will be lower than 0. In this case it is assumed that the delay is 0, implying that the congestion will not increase his speed.

At this point the agent can estimate the delay introduced by all openings.

$$pathDelay(p) = \sum_{o \in p} delay(o) \tag{4}$$

This is an example of omniscient agents, since they can always know the status of each opening. Another option is to suppose that the agent can only see the state of the opening located in the same region of the agent. In this situation the agent must be able to remember the state of the opening when it left a region, otherwise the information used to estimate the travel time at each time will not be consistent during the execution of the plan.

$$Time(p) = \tau_p + \frac{d(a,S(p))}{speed_a} + pathDelay_a(p) \tag{5}$$

Where:

- τ_p : the expected travel time of the path p

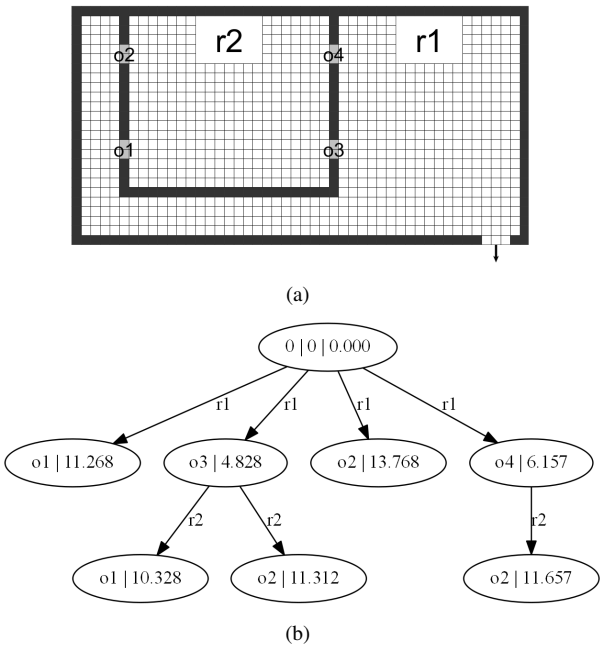


Fig. 5. Example scenario with the respective paths tree.

- $\frac{d(a,S(p)}{speed_a}$: the expected time to reach $S(p)$ from the position of the agent
- $pathDelay_a(p)$: the estimation of the delay introduced by each opening in the path, based on the memory of the agent (which may or may not be updated for each opening).

IV. Applications in Example Scenarios

In order to show the reliability and potentials of the proposed approach, results of two example scenarios will be presented in this section.

The aim of the first experimental scenario is to show the output of the *paths tree* generation algorithm. Figure 5 shows the environment and the associated data structure. The tree contains two information on each node, describing the Id of the mapped opening and the estimated time associated to its path. In addition, each edge is labeled with the Id of the region crossed by the path. The result shows that the possible minimal paths have been represented in the tree. In particular, child nodes of $o1$ and $o2$, passing inside $r2$, have not been added since that would imply a path passing from $o3$ or $o4$ and describing a not rational way through $r2$ and the corridor at the bottom of $r1$. Paths like $p = o2 \rightarrow o4 \rightarrow end$ or $p = o2 \rightarrow o4 \rightarrow end$ have been considered instead, since they could be plausible for pedestrians being in the top left corner of the scenario.

Results of the second experiment are shown in Figure 6: the illustrated environment have been populated with 200 agents, generated in the *Start* object with an arrival frequency of around 7 pers/sec. The heat maps contain the *cumulative mean densities* of pedestrians, describing a cumulative value of density in each cell for a fixed time window (in this case the duration is 50 steps, equal to 12.5 seconds). It must be noted that at each step the values are accumulated only in pedestrians

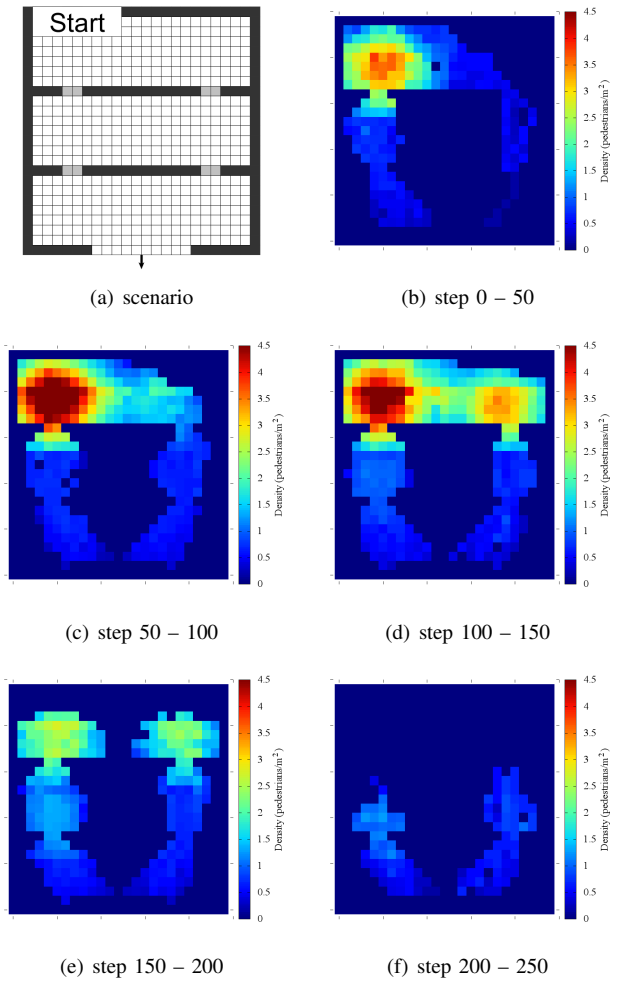


Fig. 6. The test scenario (a) with cumulative mean density maps of the simulation for various time windows (b – f).

positions, in order to give an idea of what pedestrians have perceived during the simulation. The stream of pictures shows that the arrival rate cannot be sustained from the opening at the top left of the environment, describing the shortest path towards the exit, thus a growing congestion arises in front of it. This increases the time perceived by the agents to employ the shortest path, leading a significant part of them to change their route preferring the other opening, that will get also congested in the third time window. The arrival rate stops at about step 150 and from this moment the environment starts to get empty.

V. Conclusions

The paper has presented a hybrid agent architecture for modeling tactical level decisions, related to the choice of a route to follow in an environment comprising several rooms connected by openings, integrated with a validated operational level model, employing a floor-field based approach. The described model allows the agent taking decisions based on a static a-priori knowledge of the environment and dynamic perceivable information on the current level of crowdedness of visible path alternatives. The model was experimented in benchmark scenarios showing the adequacy in providing adaptiveness to the contextual situation. The future works, on

one hand, are mainly aimed at defining requirements and an approach for the validation of the results achieved through the model: this represents an open challenge, since there are no comprehensive data sets on human tactical level decisions and automatic acquisition of this kind of data from video cameras is still a challenging task [33]. Even the mentioned [29], like most works on this topic, just explores the different alternatives that can be generated with distinct modeling choices, whereas a constrained form of validation was described in [34], although reported data are not sufficient for a quantitative cross validation of our approach. On the other hand, the addition of specific area actions (e.g. wait after reaching a certain point of interest indicated by a marker) and events (e.g. the arrival of a train) triggering agents' actions and, more generally, allowing the elaboration of more complicated agents' plans is also a planned extension on the side of model expressiveness.

REFERENCES

[1] S. Bandini, S. Manzoni, and G. Vizzari, "Agent based modeling and simulation: An informatics perspective," *Journal of Artificial Societies and Social Simulation*, vol. 12, no. 4, p. 4, 2009.

[2] G. Vizzari and S. Bandini, "Studying pedestrian and crowd dynamics through integrated analysis and synthesis," *IEEE Intelligent Systems*, vol. 28, no. 5, pp. 56–60, 2013.

[3] D. Pianini, M. Viroli, F. Zambonelli, and A. Ferscha, "HPC from a self-organisation perspective: The case of crowd steering at the urban scale," in *International Conference on High Performance Computing & Simulation, HPCS 2014, Bologna, Italy, 21-25 July, 2014*. IEEE, 2014, pp. 460–467.

[4] M. Boltes and A. Seyfried, "Collecting pedestrian trajectories," *Neurocomputing*, vol. 100, pp. 127–133, Jan. 2013.

[5] M. Boltes, J. Zhang, A. Seyfried, and B. Steffen, "T-junction: Experiments, trajectory collection, and analysis," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, Nov. 2011, pp. 158–165.

[6] S. D. Khan, G. Vizzari, S. Bandini, and S. Basalamah, "Detecting dominant motion flows and people counting in high density crowds," *Journal of WSCG*, vol. 22, no. 1-2, pp. 21–30, 2014.

[7] F. Solera and S. Calderara, "Social groups detection in crowd through shape-augmented structured learning," in *Image Analysis and Processing - ICIAP 2013 - 17th International Conference, Naples, Italy, September 9-13, 2013. Proceedings, Part I*, ser. Lecture Notes in Computer Science, A. Petrosino, Ed., vol. 8156. Springer, 2013, pp. 542–551.

[8] S. Bandini, A. Gorrini, and G. Vizzari, "Towards an integrated approach to crowd analysis and crowd synthesis: A case study and first results," *Pattern Recognition Letters*, vol. 44, pp. 16–29, 2014.

[9] L. F. Henderson, "The Statistics of Crowd Fluids," *Nature*, vol. 229, no. 5284, pp. 381–383, Feb. 1971.

[10] D. Helbing, "A fluid dynamic model for the movement of pedestrians," *arXiv preprint cond-mat/9805213*, 1998.

[11] T. Bosse, M. Hoogendoorn, M. C. A. Klein, J. Treur, C. N. van der Wal, and A. van Wissen, "Modelling collective decision making in groups and crowds: Integrating social contagion and interacting emotions, beliefs and intentions," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 52–84, 2013.

[12] A. Schadschneider, W. Klingsch, H. Klüpfel, T. Kretz, C. Rogsch, and A. Seyfried, "Evacuation Dynamics: Empirical Results, Modeling and Applications," in *Encyclopedia of Complexity and Systems Science*, R. A. Meyers, Ed. Springer, 2009, pp. 3142–3176.

[13] J. A. Michon, "A Critical View of Driver Behavior Models: What Do We Know, What Should We Do?" in *Human Behavior and Traffic Safety*, L. and Evans and R. C. Schwing, Eds. Springer US, 1985, pp. 485–524.

[14] J. Zhang, W. Klingsch, T. Rupprecht, A. Schadschneider, and A. Seyfried, "Empirical study of turning and merging of pedestrian streams in T-junction," *ArXiv e-prints*, p. 8, 2011.

[15] J. Zhang, W. Klingsch, A. Schadschneider, and A. Seyfried, "Ordering in bidirectional pedestrian flows and its influence on the fundamental diagram," *Journal of Statistical Mechanics: Theory and Experiment*, no. 02, p. 9, 2011.

[16] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, 1995.

[17] M. Chraibi, A. Seyfried, and A. Schadschneider, "Generalized centrifugal-force model for pedestrian dynamics," *Phys. Rev. E*, vol. 82, no. 4, p. 46111, 2010.

[18] I. von Sivers and G. Köster, "Dynamic Stride Length Adaptation According to Utility And Personal Space," Transportation Research Part B, vol. 74, pp. 104–117, 2015.

[19] F. Qiu and X. Hu, "Modeling group structures in pedestrian crowd simulation," *Simulation Modelling Practice and Theory*, vol. 18, no. 2, pp. 190 – 205, 2010.

[20] V. Blue and J. Adler, "Modeling Four-Directional Pedestrian Flows," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1710, no. -1, pp. 20–27, Jan. 2000.

[21] V. J. Blue and J. L. Adler, "Cellular Automata Microsimulation of Bi-Directional Pedestrian Flows," *Transportation Research Record*, vol. 1678, pp. 135–141, 1999.

[22] C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz, "Simulation of pedestrian dynamics using a two-dimensional cellular automaton," *Physica A: Statistical Mechanics and its Applications*, vol. 295, no. 3 - 4, pp. 507–525, 2001.

[23] Y. Suma, D. Yanagisawa, and K. Nishinari, "Anticipation effect in pedestrian dynamics: Modeling and experiments," *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 1-2, pp. 248–263, Jan. 2012.

[24] A. Kirchner, H. Klüpfel, K. Nishinari, A. Schadschneider, and M. Schreckenberg, "Discretization effects and the influence of walking speed in cellular automata models for pedestrian dynamics," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2004, no. 10, p. P10011, 2004.

[25] S. Bandini, L. Crociani, and G. Vizzari, "Heterogeneous Pedestrian Walking Speed in Discrete Simulation Models," in *Traffic and Granular Flow '13*, M. Chraibi, M. Boltes, A. Schadschneider, and A. Seyfried, Eds. Springer International Publishing, 2015, pp. 273–279.

[26] G. Vizzari, L. Manenti, and L. Crociani, "Adaptive Pedestrian Behaviour for the Preservation of Group Cohesion," *Complex Adaptive Systems Modeling*, vol. 1, no. 7, 2013.

[27] R. Geraerts and M. Overmars, "The Corridor Map Method: A General Framework for Real-Time High-Quality Path Planning," pp. 107–119, 2007.

[28] R. Geraerts, "Planning short paths with clearance using explicit corridors," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, May 2010, pp. 1997–2004.

[29] A. U. K. Wagoum, A. Seyfried, and S. Holl, "Modelling dynamic route choice of pedestrians to assess the criticality of building evacuation," *Advances in Complex Systems*, vol. 15, no. 07, p. 15, 2012.

[30] S. Bandini, L. Crociani, and G. Vizzari, "Heterogeneous speed profiles in discrete models for pedestrian simulation," in *Proceedings of the 93rd Transportation Research Board annual meeting*, 2014.

[31] L. Crociani, A. Invernizzi, and G. Vizzari, "A hybrid agent architecture for enabling tactical level decisions in floor field approaches," *Transportation Research Procedia*, vol. 2, pp. 618–623, 2014.

[32] T. Kretz, C. Bönisch, and P. Vortisch, "Comparison of Various Methods for the Calculation of the Distance Potential Field," in *Pedestrian and Evacuation Dynamics 2008*, W. W. F. Klingsch, C. Rogsch, A. Schadschneider, and M. Schreckenberg, Eds. Springer Berlin Heidelberg, 2010, pp. 335–346.

[33] S. D. Khan, G. Vizzari, and S. Bandini, "Identifying sources and sinks and detecting dominant motion patterns in crowds," *Transportation Research Procedia*, vol. 2, no. 0, pp. 195 – 200, 2014, the Conference on Pedestrian and Evacuation Dynamics 2014 (PED 2014), 22-24 October 2014, Delft, The Netherlands.

[34] M. Asano, T. Iryo, and M. Kuwahara, "Microscopic pedestrian simulation model combined with a tactical model for route choice behaviour," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 6, pp. 842 – 855, 2010, special issue on Transportation Simulation Advances in Air Transportation Research.

Outline of a Formalization of JADE Multi-Agent Systems

Federico Bergenti
Dipartimento di Matematica e Informatica
Università degli Studi di Parma
Parco Area delle Scienze 53/A, 43124 Parma, Italy
Email: federico.bergenti@unipr.it

Eleonora Iotti, Agostino Poggi
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma
Parco Area delle Scienze 181/A, 43124 Parma, Italy
Email: eleonora.iotti@studenti.unipr.it, agostino.poggi@unipr.it

***Abstract*—This paper proposes a formalization of JADE agents and multi-agent systems based on transition systems. The first section introduces the aims and scope of the research and it focuses the content of the paper. The second section enumerates the abstractions and the structures used in the formalization. Successively, third section presents the formal semantics of the parts of a JADE-based source code that are involved in the management of *(i)* the life cycle of agents and *(ii)* the behaviours of agents. Fourth section shows a very simple JADE agent and it exemplifies the use of the proposed transition system. Finally, a brief recapitulation of the work concludes the paper.**

I. Introduction

JADE [1] is today one of the most widely used tools for the development of multi-agent systems both in research and in the industry. It is a core component of a complex software system that helps managing one of the most penetrating telecommunication networks in Europe, serving millions of consumers daily [2]. It has been recently enhanced to support smart appliances [3], and it is the base of a recent initiative intended to revitalize the use of software agents to support social activities [4], both cooperative (see, e.g., [5], [6]) and competitive. Besides this, JADE was initially conceived primarily as a practical tool to help researchers experimenting with FIPA technology (www.fipa.org). JADE was recognized by FIPA community as the tool that most accurately implemented FIPA specifications, and it was often used for validating new specifications and for assessing the conformance of other tools to FIPA guidelines and specifications. Moreover, it was often selected to experiment potential enhancements to FIPA specifications (see, e.g., [7]), and to gather the interest of FIPA members on common projects (see, e.g., [8]). Such a close relationship with an official standardization body forced JADE architects to let the design of JADE APIs open, so that third-party developers could adopt JADE—and FIPA together with it—with minimal restrictions. For example, the agent model that JADE provides was intentionally left simple and not formally specified so that developers were not forced to adopt a specific agent model just because they wanted their agents to be FIPA compliant.

After more than 15 years of JADE development and use, this paper first proposes a formal semantics of JADE agents and multi-agent systems to support reasoning on JADE-based software systems. The proposed semantics is based on transition systems [9] and it closely follows the intended meaning of JADE APIs to ensure that properly written JADE agents can be reviewed in terms of the proposed semantics. No refinement of the JADE agent model is proposed and the semantics is ground on the semantics of Java classes. We always assume the availability of an underlying Java transition system and we refrain from formalizing it because it is a critical topic out of the scope of this paper (see, e.g., [10]).

For the sake of brevity, only an outline of the proposed semantics is presented in this paper, and interested readers are invited to consult an upcoming paper that provides a complete description of the formalization.

II. Definitions and Terminology

The proposed formalization considers only five main entities that collectively describe a JADE multi-agent system. The first of such entities is the multi-agent system itself—the MAS—which is seen as a sort of environment where agents live. Such an environment is subject to internal and external events that may modify its state and the state of each agent belonging to the MAS. Also, a MAS accounts for all entities that we need to formalize the JADE system: within a MAS we are able to see all agents and their respective behaviours. The second entity that we consider is the agent. An agent is a complex entity that has a state, detailed in Section II-C, and a list of behaviours. The behaviour is the third entity that we consider: it has a state, a type and an associated action. Actions are not entities in the proposed formalization, and they are always associated to behaviours. Actually, we may think of an action as a list of activites that an agent performs when it decides to activate the behaviour that encapsulates the action. The type of the behaviour changes the way the agent performs the associated action, e.g., a cyclic behaviour permits the agent to perform the action cyclically; conversely, a one-shot behaviour in meant to have the action performed only once. Each agent (respectively, behaviour) belongs to one agent class (respectively, behaviour class), and such classes are the last two of the five entities that we consider. An agent class is defined as a group of agents that have behaviours belonging to the same classes, and a behaviour class is a group of behaviours that have the same type and the same action. It is worth noting that the word class was chosen to identify the means that JADE provides to group agents and behaviours by specific properties, i.e., Java classes. This choice implicitly types agents, but this work does not address this issue and related issues (see, e.g., [11]).

A. Constants

In order to give a formal definition of the aforementioned five entities, we choose a first-order language whose alphabet consists of a finite set of constants, predicate symbols, function symbols, and the usual logical symbols (i.e., variables, connectives, and quantifiers). We let $\mathcal{C}$ be the finite set of constants and $\mathcal{V}$ the countably infinite set of variables.

In order to account for the life cycle state of an agent, we define specific constants:`initiated`, `deleted`, `active`, `suspended`, and `waiting`. Similarly, the life cycle state of a behaviour is described by constants `initiated`, `active`, `done`, and `blocked`. We call S_A the finite set of the agent life cycle state constants, and S_B the finite set of behaviour life cycle state constants. Each behaviour needs a type and a description of its internal state: we call T and S the sets of such constants, respectively. Moreover, we also need a finite set of constants that describes the state of the environment where agents live, which we call W.

In summary, the set of constants that we allow for the first-order language used to describe the proposed semantics is composed of the union of all aforementioned sets:

$$\mathcal{C} = S_A \cup S_B \cup T \cup S \cup W. \tag{1}$$

B. Variables and identifiers

In addition to constants and variables, we define a set of identifiers called I. Every time an entity is created, it gets a name from such a set. Each agent class and behaviour class is associated with a specific identifier in the set I. Therefore we need a semantic structure that associates the *name* of the class to the class itself. We call $\mathcal{F}_A$ the set of all agents classes and $\mathcal{F}_B$ the set of all behaviour classes. Such semantic structures are called *semantic contexts* of the classes, and they are

$$\rho_A : I \to \mathcal{F}_A, \tag{2}$$
$$\rho_B : I \to \mathcal{F}_B. \tag{3}$$

In JADE, each agent has a name and a global unique identifier (called *AID*, for agent identifier), which contains its local name and its platform name. So we can consider the AID as a structured identifier defined a function

$$\mathrm{aid} : \mathcal{V} \to I \tag{4}$$

that maps a variable a representing an agent to its AID. Moreover, each behaviour has a name, which is its unique identifier. We can access such an identifier through the function

$$\mathrm{bid} : \mathcal{V} \to I. \tag{5}$$

Each agent (respectively, behaviour) needs to be instantiated and maintained in a semantic structure, here generically called *heap*, which associates a concrete reference (i.e., an identifier) to the representation of the agent (respectively, behaviour). We call A the set of all agents and B the set of all behaviours, and we define a function α that accesses the state of an agent as

$$\alpha : I \to A. \tag{6}$$

Similarly, we define the function β that accesses the state of a behaviour as

$$\beta : I \to B. \tag{7}$$

The composition $\alpha \circ \mathrm{aid} : \mathcal{V} \to A$ associates an agent to a variable in the first-order language, while the composition $\beta \circ \mathrm{bid} : \mathcal{V} \to B$ does the same with respect to a behaviour. We may write $\alpha(a)$ instead of $\alpha(\mathrm{aid}(a))$ with a slight abuse of notation when the intended meaning is evident from the context.

C. Events

A MAS is constantly subject to two types of events: internal and external. Each agent is fed with a sub-sequence of such events in its life. Some of such events are particularly important because they can change the state of the MAS (these are the external events), and/or the state of agents and respective behaviours (these are the internal events). The set of such notable events that JADE manages, which we call E, contains the following events:

- $\mathrm{Create}(a, F)$ denotes the creation of the agent a of class F;
- $\mathrm{Kill}(a)$ denotes the destruction of the agent a;
- $\mathrm{Wait}(a)$ denotes the state change of the agent a to the state `waiting`;
- $\mathrm{Wake}(a)$ denotes the state change of the agent a from `waiting` to `active`;
- $\mathrm{Suspend}(a)$ denotes the state change of the agent a to the state `suspended`;
- $\mathrm{Activate}(a)$ denotes the state change of the agent a from `suspended` to the state it had before becoming suspended;
- $\mathrm{Create}(a, b, F)$ denotes the creation of the behaviour b of class F and agent a;
- $\mathrm{Block}(b)$ denotes the state change from `active` to `blocked` for the behaviour b;
- $\mathrm{Restart}(b)$ denotes the state change from `blocked` to `active` for the behaviour b;
- $\mathrm{Start}(\mathrm{MAS})$ denotes the first event that occurs in the MAS; and
- $\mathrm{End}(\mathrm{MAS})$ denotes the event that marks the termination of the MAS (with all agents and behaviours); it is the last event that occurs in a computation of the MAS.

The events in E that denotes changes in the state of an agent are related to each other as shown in Figure II-C (left). Similarly, Figure II-C (right) shows the finite state machine that represents the state changes of a behaviour.

In order to analyze the life cycle of agents and behaviours, we need to consider which event occurs in single steps of the computation: if $e \in E$ and $t \in \mathbb{N}$, the pair $\langle t, e \rangle$ means that the event e occurred at step t.

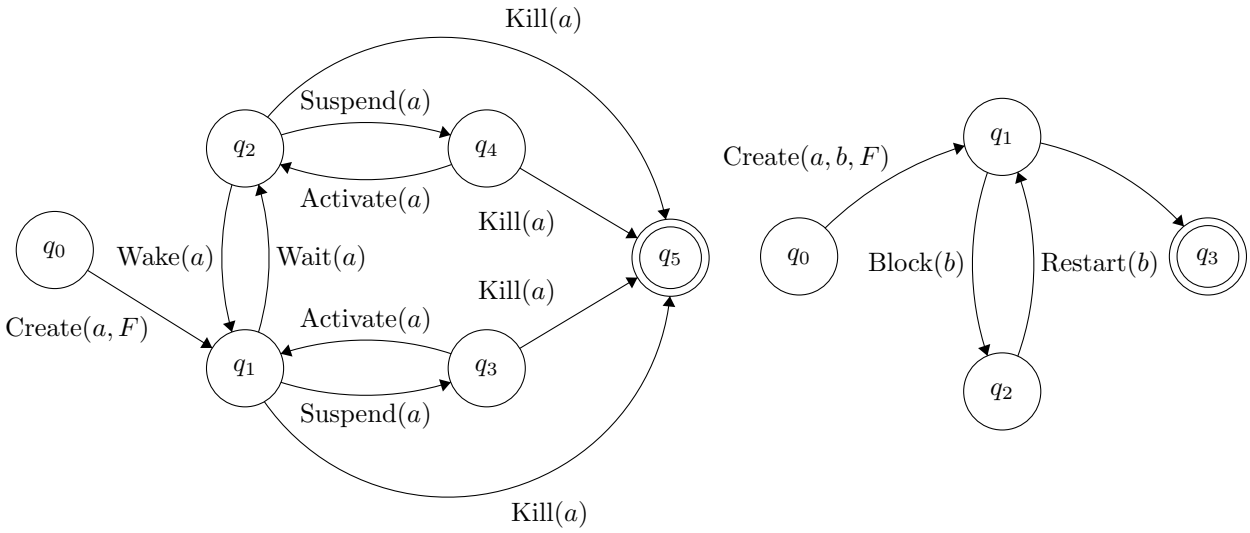


Figure 1. Agent (left) and behaviour (right) life cycle events and their relationships. For agents: q_0 = `initiated`, q_1 = `active`, q_2 = `waiting`, q_3 = `suspended_from_active`, q_4 = `suspended_from_waiting` and q_5 = `deleted`; for behaviours q_0 = `initiated`, q_1 = `active`, q_2 = `blocked` and q_3 = `done`.

D. Structures

An agent class is a pair $F_a = \langle L_B, \varphi \rangle$, where L_B is a list of possible behaviours and φ a function that maps the names of its methods (in the set I of identifiers) to the associated blocks of code, i.e., sequences of statements written in Java with JADE.

Similarly, a behaviour class is a pair $F_b = \langle t, \psi \rangle$, where $t \in T \subseteq \mathcal{C}$ fixes the type of the behaviour in the class (e.g., *cyclic* or *one-shot*), and ψ is a function that associates specific names to the related blocks of code.

In JADE, an agent is an entity that belongs to a particular class and that has a list of behaviours. So, an agent is completely described by the name of its class and by its state, as follows.

Definition 1 (Agent): An *agent* is a tuple

$$\langle \mathtt{F}, s_a, s_w, L_b, L_e \rangle \in A \tag{8}$$

where $\mathtt{F} \in I$ is the identifier of the class, $s_a \in S_A$ is the state of the agent, $s_w \in W$ the state of the environment seen by the agent, L_b is a finite list of behaviours and L_e a finite list of events.

Similarly, we can define a behaviour as a specific entity which belongs to a class, as follows.

Definition 2 (Behaviour): A *behaviour* is a tuple

$$\langle \mathtt{F}, \mathtt{myAgent}, s_b, s_w, s \rangle \in B \tag{9}$$

where $\mathtt{F} \in I$ is the identifier of the class, $\mathtt{myAgent} \in I$ is the unique identifier of the associated agent, $s_b \in S_B$ is the state of the behaviour, $s_w \in W$ is the state of the environment seen by the behaviour and $s \in S$ is the internal state of the behaviour. We call B the set of all behaviours.

In JADE, each agent lives within a MAS and therefore, for every time step T we can consider a global list of events L_E that occurred to all agents of that MAS since its activation, i.e., since event $\mathrm{Start(MAS)}$.

Definition 3 (MAS): A *MAS* is defined by a tuple

$$\mathrm{MAS} = \langle A', B', L_E \rangle_T \tag{10}$$

where $A' \subseteq A$ is a finite set of agents, $B' \subseteq B$ is a finite set of behaviours, L_E is the list of events occurred in this MAS up to time step $T \in \mathbb{N}$.

In addition, to support the semantics described in next section, we define $\mathcal{S}_b$, a selector for the behaviours of an agent, i.e., the *behaviour scheduler*. Such a function computes the current behaviour to execute, from the list L_b of an agent a as

$$\mathcal{S}_b : A' \to B'. \tag{11}$$

III. Semantics

The semantics of the life cycle of agents, given the events described in Section II-C, is described in the form of a *labelled transition system* (see, e.g., [9]). The proposed transition system, called *MAS transition system*, is a pair $\langle \Gamma, \to \rangle$ where Γ is a finite set of configurations and $\to$ is a binary relation on Γ. Once a MAS is fixed, the labels of such a transition system are the pairs time-event defined above.

Definition 4 (Configuration): A *configuration* in the transition system is described by a tuple

$$\langle C, \sigma, \alpha, \beta \rangle \in \Gamma \tag{12}$$

where C is a command, σ represents the state (stores and environments) of the underlying Java system, α and β are the heaps that represent the state of all agents and of all behaviours, respectively.

For the sake of clarity and when no ambiguity can arise, we may not enumerate all four elements of a configuration in the description of transition rules. Moreover, we use the symbol ε for a configuration with no commands. The abstract syntax used for a command (or statement) C is the Java syntax, as defined, e.g., in [10].

Each transition refers to an environment, or semantic context. In this case, we use the class contexts defined in Section II-D. So, for example, a transition may be written as follows

$$\rho_A, \rho_B \vdash_N \langle C, \sigma, \alpha, \beta\rangle \xrightarrow{\langle t,e\rangle} \langle C', \sigma', \alpha', \beta'\rangle. \quad (13)$$

where ρ_A and ρ_B are the class contexts and $N \subseteq I \cup \mathcal{V}$ is a finite set of *names*, i.e., identifiers or variables used in the transition.

A. Semantics of commands with events

We need a formal semantics to manage the events and the state changes that occur in the life cycle of an agent. To this extent, we use a subsystem of the main MAS transition system. Such a subsystem has the same configuration of the MAS system, but the transition relation is called $\rightarrow_{\text{com}}$. The execution of such a system may be interrupted by an event that may cause a state change: this is represented by taking the label on the transition relation that indicates the current execution step and by matching it with the event just occurred. Notably, at some execution step no event occurs, and we indicate them with $\langle t, \varpi\rangle$.

Let us consider a fixed MAS $\langle A', B', L_E\rangle_T$. A list of statements, indicated with $C_1; C_2$, is computed as usual from left to right taking care of the propagation of the current event, as shown by the rule

$$\frac{\langle C_1, \sigma, \alpha, \beta\rangle \xrightarrow{\langle t,e\rangle}_{\text{com}} \langle \varepsilon, \sigma', \alpha', \beta'\rangle}{\langle C_1; C_2, \sigma, \alpha, \beta\rangle \xrightarrow{\langle t,e\rangle}_{\text{com}} \langle C_2, \sigma', \alpha', \beta'\rangle}. \quad (14)$$

When no event occurs, as we can see in the rule (15), the statement C is computed by means of the underlying Java semantics system, which we call $\rightarrow_{\text{jstmt}}$

$$\frac{\langle C, \sigma\rangle \rightarrow_{\text{jstmt}} \langle \varepsilon, \sigma'\rangle}{\langle C, \sigma\rangle \xrightarrow{\langle t,\varpi\rangle}_{\text{com}} \langle \varepsilon, \sigma'\rangle}. \quad (15)$$

We work under the reasonable assumption that such a subsystem may change only the state σ and we do not consider changes in all stores and environments of Java.

We then define specific rules to update the state of an agent. When an event e related to the agent $a = \langle \mathtt{F}, s_a, s_w, L_b, L_e\rangle$ occurs, s_a is changed to a new state, and e is stored in the list of events L_e. Similarly, we define rules for the behaviour state changes. In this case, there is no need for a list of events inside the behaviour entity, as shown in Figure II-C. For example, the following rule describes the state change from `active` to `blocked`

$$\frac{e = \text{Block}(b) \quad \beta(b) = \langle \mathtt{F}, \mathtt{active}, s_w, s\rangle}{\langle C, \beta\rangle \xrightarrow{\langle t,e\rangle}_{\text{com}} \langle C, \beta[\langle \mathtt{F}, \mathtt{blocked}, s_w, s\rangle/b]\rangle}. \quad (16)$$

There is no explicit rule for the change from `active` to `done` because the latter is reached only after the execution of the action for a one-shot behaviour, and cyclic behaviours never change to `done`. Obviously the event $\text{End}(\text{MAS})$ stops all behaviours and agents in a MAS, forcing them to reach their respective final states.

B. Agent life cycle

We describe the life cycle of a generic agent, instantiated in the MAS $\langle A', B', L_E\rangle_T$, starting from its creation. When a new agent $a \in A$ of class $\mathcal{F}_a = \langle L_B, \varphi\rangle$ is created, it is memorized in a location `a` of the heap α and the function `setup` of its class is executed. We assume that the code of the method `setup` is accessible from the function φ of the agent class

$$\varphi(\mathtt{setup}) = \{\mathtt{B}\}. \quad (17)$$

In this case, the MAS system has the transition

$$\rho_A \vdash \langle \varepsilon, \alpha\rangle \xrightarrow{\langle t,e\rangle} \langle \mathtt{a.setup()}\{\mathtt{B}\}, \alpha'\rangle \quad (18)$$

where $\alpha' = \alpha[\langle \mathtt{F}, \mathtt{initiated}, \omega, [\,], [\text{Create}(a, F)]\rangle/\mathtt{a}]$. Note that the agent class context is crucial within this transition because it gives us the necessary information regarding the `setup` method.

The block `B` of the agent `setup` is computed by the commands transition system, as follows

$$\frac{\rho_A \vdash_{\mathtt{a}} \langle \mathtt{B}, \sigma, \alpha, \beta\rangle \xrightarrow{\langle t,e\rangle}_{\text{com}} \langle \varepsilon, \sigma', \alpha', \beta'\rangle \quad \dots}{\rho_A \vdash \langle \mathtt{a.setup()}\{\mathtt{B}\}, \sigma, \alpha, \beta\rangle \xrightarrow{\langle t,e\rangle} \langle \varepsilon, \sigma', \alpha'', \beta'\rangle} \quad (19)$$

where $\alpha'' = \alpha'[\langle \mathtt{F}, \mathtt{active}, s_w, L_b, L_e\rangle/\mathtt{a}]$.

During `setup`, agent a can add and/or remove behaviours from its list L_b, through the following statements

```
addBehaviour(b),  removeBehaviour(b).
```

It is worth noting that such statements can be used also inside the action of a behaviour, and not only in the `setup` method, to change the list of behaviours dynamically. Finally, after `setup`, the agent enters the `active` state and it becomes ready to select a behaviour in its list. The life of an agent continues by performing actions of behaviours until the event $\text{Kill}(a)$ occurs. When an agent a is killed, it computes the function `takeDown`. Only after the execution of `takeDown` the agent a is removed from the heap by assigning $\alpha(\mathtt{a}) = \varpi$.

C. Behaviour actions

If $\alpha(\mathtt{a}) = a = \langle \mathtt{F}', s_a, s'_w, L_b, L_e\rangle \in A'$ is an instantiated agent and $\beta(\mathtt{b}) = b = \langle \mathtt{F}, \mathtt{a}, \mathtt{active}, s_w, s\rangle \in B'$ is a behaviour that appears in the list of the agent, i.e., $b \in L_b$, then the behaviour may be selected by the behaviour scheduler of the agent to perform the relative action, i.e., $\mathcal{S}_b(a) = b$. The transition of the main system then calls the `action` method of the class of b

$$\rho_B \vdash \langle \varepsilon, \sigma, \alpha, \beta\rangle \rightarrow \langle \mathtt{b.action()}\{\mathtt{C}\}, \sigma, \alpha, \beta\rangle.$$

Just like for the method `setup` of the agent, the block of code `C` of `action` is executed by the subsystem $\rightarrow_{\text{com}}$.

A one-shot behaviour terminates after a single execution of its action, and it is removed from the list of behaviours of the agent, through the function `removeBehaviour`. So, if the agent is still alive, the behaviour scheduler chooses another behaviour. On the contrary, if the behaviour b is cyclic, then its action is performed cyclically and it is never automatically removed from the list of behaviours of the agent.

$$\frac{\rho_A, \rho_B \vdash_{\mathrm{a,b}} \langle \mathtt{C;removeBehaviour(b)}, \sigma, \alpha, \beta\rangle \xrightarrow{\langle t,e\rangle}_{\mathrm{com}} \langle \varepsilon, \sigma', \alpha', \beta'\rangle \quad \ldots \quad \beta'(\mathtt{b}) = \langle \mathtt{F, a, active}, s_w, s\rangle}{\rho_B \vdash \langle \mathtt{b.action()\{C\}}, \sigma, \alpha, \beta\rangle \xrightarrow{\langle t,e\rangle} \langle \varepsilon, \sigma', \alpha', \beta'[\langle \mathtt{F, a, done}, s_w, s\rangle/\mathtt{b}]\rangle} \tag{20}$$

$$\frac{\rho_A, \rho_B \vdash_{\mathrm{a,b}} \langle \mathtt{C;removeBehaviour(b)}, \sigma, \alpha, \beta\rangle \xrightarrow{\langle t,e\rangle}_{\mathrm{com}} \langle \varepsilon, \sigma', \alpha', \beta'\rangle \quad \ldots \quad \beta'(\mathtt{b}) = \langle \mathtt{F, a, blocked}, s_w, s\rangle}{\rho_B \vdash \langle \mathtt{b.action()\{C\}}, \sigma, \alpha, \beta\rangle \xrightarrow{\langle t,e\rangle} \langle \varepsilon, \sigma', \alpha', \beta'\rangle} \tag{21}$$

Figure 2. Rules governing the transition of a one-shot behaviour from the active to the blocked state. The behaviour scheduler of an agent can select only behaviours that are in the active state.

Note that a necessary condition for the scheduler $\mathcal{S}_b(a)$ to choose a behaviour is that it must be in the active state. A behaviour is normally in the active state, but it can reach the blocked state during the execution of its action. When this occurs, the execution continues up to the natural termination of the action and the blocked behaviour is not removed from the list of behaviours. Anyway, the scheduler cannot choose it until it returns to the active state, as detailed, for one-shot behaviours, in rules (20) and (21), summarized in Figure 2.

IV. A Didactic Example

This section discusses a simple example of JADE agent and shows its semantics according to the transition system outlined in previous sections. The simple agent has the following source code.

```
1  import jade.core.Agent;
2  import jade.core.behaviours.OneShotBehaviour;
3
4  public class HelloAgent extends Agent {
5    protected void setup() {
6      Greetings greetings = new Greetings(this);
7
8      addBehaviour(greetings);
9    }
10
11   private static class Greetings extends
         OneShotBehaviour {
12     public Greetings(Agent a) {
13       super(a);
14     }
15
16     public void action() {
17       System.out.println("Hello world");
18     }
19   }
20 }
```

Class `HelloAgent` defines an agent class and its static inner class a behaviour class. This is formalized by setting the pairs $\langle L_B, \varphi\rangle$ and $\langle t, \psi\rangle$ using the information contained in the source code. Then, the context is initialized to create a reference between the identifier of classes and their relative structures, as follows.

$$\rho_A(\mathtt{HelloAgent}) = \langle L_B, \varphi\rangle$$
$$\rho_B(\mathtt{Greetings}) = \langle \mathtt{one\text{-}shot}, \psi\rangle$$
$$L_B = [\mathtt{Greetings}]$$

$$\varphi(\mathtt{setup}) = \{$$
```
    Greetings greetings =
        new Greetings(this);
    addBehaviour(greetings);
}
```

$$\psi(\mathtt{action}) = \{$$
```
    System.out.println("Hello world");
}
```

Suppose that the MAS starts at execution step 0, and that the first event that occurs is the creation of agent *hello*, belonging to the class *HelloAgent*

$$e_0 = \langle 0, \mathrm{Start(MAS)}\rangle \quad \mathrm{MAS} = \langle \emptyset, \emptyset, [e_0]\rangle_T$$
$$\downarrow$$
$$e_1 = \langle 1, \mathrm{Create(hello, HelloAgent)}\rangle$$
$$A' = \{\mathrm{hello}\},\ L_E = [e_1, e_0].$$

Variable *hello* allows retrieving the information about the new agent, and we can access its AID by function $\mathrm{aid(hello)} = \mathtt{hello}$, and control its state thanks to the heap α.

The event $\mathrm{Create(hello, HelloAgent)}$ activates the first transition

$$\rho_A \vdash \langle \varepsilon, \alpha\rangle \xrightarrow{e_1} \langle \mathtt{hello.setup()\{B\}}, \alpha^i\rangle$$

where $\{\mathtt{B}\}$ is the block of code obtained by $\varphi(\mathtt{setup})$ and

$$\alpha^i = \alpha[\langle \mathtt{HelloAgent, initiated}, \omega, \mathtt{[]}, [e_1]\rangle / \mathtt{hello}].$$

Suppose that no event occurs at execution step 2: the main transition system enters in the subsystem $\rightarrow_{\mathrm{com}}$, as follows.

$$\rho_A \vdash_{\mathrm{hello}} \langle \mathtt{Greetings\ greetings = new\ Greetings()}, \sigma, \alpha^i, \beta\rangle$$
$$\downarrow e_3$$
$$\langle \mathtt{addBehaviour(greetings)}, \sigma^i, \alpha^i, \beta^i\rangle$$
$$\downarrow e_4$$
$$\langle \varepsilon, \sigma^{ii}, \alpha^{ii}, \beta^i\rangle$$

where σ^{ii} is obtained from a transition of the Java subsystem $\rightarrow_{\text{jstmt}}$, called at every execution step, and

$$\beta^i = \beta[\langle \texttt{Greetings}, \texttt{hello}, \texttt{active}, \omega, \omega\rangle /\texttt{greetings}]$$

$$\alpha^{ii} = \alpha^i[\langle \texttt{HelloAgent}, \texttt{initiated}, \omega, [\texttt{greetings}], [e_3, e_1]\rangle/\texttt{hello}].$$

This transition is necessary to apply rule (19)

$$\frac{\rho_A \vdash_{\text{hello}} \langle \texttt{B}, \sigma, \alpha^i, \beta\rangle \xrightarrow{e_2}_{\text{com}} \langle \varepsilon, \sigma^{ii}, \alpha^{ii}, \beta^i\rangle \quad \ldots}{\rho_A \vdash \langle \texttt{hello.setup()}\{\texttt{B}\}, \sigma, \alpha^i, \beta\rangle \xrightarrow{e_2} \langle \varepsilon, \sigma^{ii}, \alpha^{iii}, \beta^i\rangle}$$

where the state of the agent is eventually changed to active

$$\alpha^{iii} = \alpha^{ii}[\langle \texttt{HelloAgent}, \texttt{active}, \omega, [\texttt{greetings}], [e_3, e_1]\rangle/\texttt{hello}].$$

Then, because agent *hello* is active, the scheduler can choose a behaviour to execute. The only behaviour in the list of behaviours of the agent is *greetings* so $\mathcal{S}_b(\text{hello}) = \text{greetings}$

$$\rho_B \vdash \langle \varepsilon, \sigma^{ii}, \alpha^{iii}, \beta^i\rangle \downarrow e_5 \langle \texttt{greetings.action()}\{\texttt{B}\}, \sigma^{ii}, \alpha^{iii}, \beta^i\rangle.$$

The block of code $\{\texttt{B}\}$ of the action, accessed through the function ψ, is executed in the $\rightarrow_{\text{com}}$ subsystem, as follows.

$$\rho_A, \rho_B \vdash_{\text{hello,greetings}} \langle \texttt{System.out.println("Hello world")}\sigma^{ii}, \alpha^{iii}, \beta^i\rangle \downarrow e_6 \langle \varepsilon, \sigma^{iii}, \alpha^{iii}, \beta^i\rangle$$

where σ^{iii} is obtained by the Java transition system. According to rule (20), at the end of the computation of its action, the state of the behaviour is changed to `done`.

$$\frac{\rho_A, \rho_B \vdash_{\text{hello,greetings}} \langle \texttt{B}, \sigma^{ii}, \alpha^{iii}, \beta^i\rangle \xrightarrow{e_6}_{\text{com}} \langle \varepsilon, \sigma^{iii}, \alpha^{iii}, \beta^i\rangle \quad \ldots}{\rho_A \vdash \langle \texttt{greetings.action()}\{\texttt{B}\}, \sigma^{ii}, \alpha^{iii}, \beta^i\rangle \xrightarrow{e_6} \langle \varepsilon, \sigma^{iii}, \alpha^{iii}, \beta^{ii}\rangle}.$$

In fact:

$$\beta^{ii} = \beta^i[\langle \texttt{Greetings}, \texttt{hello}, \texttt{done}, \omega, \omega\rangle /\texttt{greetings}]$$

Let us now suppose that when $T = 7$ the event $\text{End}(\text{MAS})$ occurs, thus producing the following transition

$$\rho_A, \rho_B \vdash \langle \varepsilon, \sigma^{iii}, \alpha^{iii}, \beta^{ii}\rangle \xrightarrow{e_7} \langle \texttt{E}, \sigma^{iii}, \alpha^{iii}, \beta^{ii}\rangle$$

where `E` is a special command which denotes the termination of the execution. The event occurred are summarized below.

$$\begin{gathered} e_2 = \langle 2, \varpi\rangle \\ \downarrow \\ e_3 = \langle 3, \text{Create}(\text{hello}, \text{greetings}, \text{Greetings})\rangle \\ \downarrow \\ e_4 = \langle 4, \varpi\rangle \\ \downarrow \\ e_5 = \langle 5, \varpi\rangle \\ \downarrow \\ e_6 = \langle 6, \varpi\rangle \\ \downarrow \\ e_7 = \langle 7, \text{End}(\text{MAS})\rangle \end{gathered}$$

V. Conclusions

This paper provides an outline of the major ingredients of a semantics of JADE agents and multi-agent systems based on transition systems. The five main entities that we consider, namely the multi-agent system, agents and their classes, behaviours and their classes, are discussed. The paper provides an example of the transition system that can be obtained from a very simple JADE agent. The complete description of the semantics cannot fit the constraints of a workshop paper and interested readers are directed to an upcoming journal paper that also discusses the complete semantics of message passing.

References

[1] F. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*. Wiley Series in Agent Technology, 2007.

[2] F. Bergenti, G. Caire, and D. Gotta, "Large-scale network and service management with WANTS," in *Industrial Agents: Emerging Applications of Software Agents in Industry*. Elsevier, 2015, pp. 231–246.

[3] F. Bergenti, G. Caire, and D. Gotta, "Agents on the move: JADE for Android devices," in *Procs. Workshop From Objects to Agents*, 2014.

[4] F. Bergenti, G. Caire, and D. Gotta, "Agent-based social gaming with AMUSE," in *Procs. 5^{th} Int'l Conf. Ambient Systems, Networks and Technologies (ANT 2014) and 4^{th} Int'l Conf. Sustainable Energy Information Technology (SEIT 2014)*, ser. Procedia Computer Science. Elsevier, 2014, pp. 914–919.

[5] F. Bergenti and A. Poggi, "Agent-based approach to manage negotiation protocols in flexible CSCW systems," in *Procs. 4^{th} Int'l Conf. Autonomous Agents*, 2000, pp. 267–268.

[6] F. Bergenti, A. Poggi, and M. Somacher, "A collaborative platform for fixed and mobile networks," *Communications of the ACM*, vol. 45, no. 11, pp. 39–44, 2002.

[7] F. Bergenti and A. Poggi, "Ubiquitous information agents," *Int'l J. Cooperative Information Systems*, vol. 11, no. 34, pp. 231–244, 2002.

[8] F. Bergenti, A. Poggi, B. Burg, and G. Caire, "Deploying FIPA-compliant systems on handheld devices," *IEEE Internet Computing*, vol. 5, no. 4, pp. 20–25, 2001.

[9] G. D. Plotkin, "A Structural approach to Operational Semantics," *J. Log. Algebr. Program.*, vol. 60-61, pp. 17–139, 2004.

[10] J. Alves-Foss, *Formal syntax and semantics of Java*. Springer Science & Business Media, 1999, no. 1523.

[11] M. Baldoni, C. Baroglio, and F. Capuzzimati, "Typing multi-agent systems via commitments," *Post-Procs. 2^{nd} Int'l Workshop on Engineering Multi-Agent Systems (EMAS 2014), Revised Selected and Invited Papers*, pp. 388–405, 2014.

A Market Mechanism for QoS–aware Multi–Robot Task Allocation

Francesco Barile*, Alessandra Rossi†, Maricarla Staffa‡, Claudia Di Napoli§ and Silvia Rossi‡
* Dipartimento di Matematica e Applicazioni, Università degli Studi di Napoli "Federico II"
francesco.barile@unina.it
† Dipartimento di Fisica, Università degli Studi di Napoli "Federico II"
alessandra.rossi@unina.it
‡ Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione, Università degli Studi di Napoli "Federico II"
{ mariacarla.staffa, silvia.rossi}@unina.it
§ Istituto di Calcolo e Reti ad Alte Prestazioni, CNR
claudia.dinapoli@cnr.it

***Abstract*—Market mechanisms, such as auctions and negotiations, are often used for efficient task allocation in multi–robot domains where tasks are characterized by quality parameters that are related to the way the task is executed depending on the specific robot capabilities. In these cases, usually robots negotiate their respective assignments in order to optimize task distribution according to their own utility function. In this work, a market–based negotiation mechanism is proposed to allocate tasks to a set of robots by taking into account end–to–end requirements that the complete allocation should meet in terms of the considered quality parameters. Negotiation takes place on these parameters that are considered goods to be traded by the individual robots, depending on their strategies, so that they can successfully negotiate to obtain the task allocation.**

I. Introduction

Networked robotic devices, such as mobile robots, drones, unmanned vehicles and position sensors, are starting to be concretely employed in many real contexts, especially in emergency and rescue activities, where navigation and search operations take place both in indoor and outdoor environments with the human supervision. In this context, the use and the arrangement of multiple robots has been proven to help in achieving the task, both in terms of execution speed, increase of robustness, reliability, quality and performance of solutions in general. However the displacement of multiple robots comes with the cost of coordination to allocate resources and tasks among them in a way that enables them to accomplish their mission efficiently and reliably.

Researchers have recently applied the principles of market economies to multi–robot coordination [1]. Market mechanisms, such as auctions and negotiations, are often used for efficient task allocation in multi–robot domains [2]. More specifically, they are used to determine the optimal allocation (distribution) of tasks to robots by considering it as an optimization problem with some cost functions, such as, for example, distance to travel, battery consumption or battery autonomy. These functions depend on the specific application, and they are usually expressed as a minimization of the robots individual costs, or the total cost of the mission, although others are possible. In any case, the optimal task allocation problem is known to be NP–hard.

In this work, our assumption is that, in a more broad context, ubiquitous robots can be represented as heterogeneous self–interested agents that can provide different services (e.g., to execute a specific task) depending on their own capabilities, which are a priori defined. According to their capabilities, robots may provide services with different performance levels, which can be evaluated depending on dynamic information and that can be traded, so allowing robots to dynamically adjust the performance they execute a task with, in order to be assigned the task. Of course, it is not possible to obtain an optimal allocation, but any allocation of tasks that meets global constraints on the complete allocation is considered an acceptable solution.

II. Multi-Robot Task allocation

Mobile robot teams can fulfill a goal more efficiently than a single robot by sharing the workload and by optimizing the use of available resources. In fact, a given system–level (or global) task can be divided into m sub–components $\{T_1, \ldots, T_m\}$ which can be assigned to individual n robots $\{R_1, \ldots, R_n\}$ that can execute them. How a global task is decomposed is a crucial problem addressed by the planning methods, while the distribution of subtasks to robots is known as task allocation.

A task decomposition algorithm has to consider both the tasks nature, and restrictions in order to accommodate them among the agents for the execution. Tasks can be long-term (e.g. monitoring an environment) or transient (e.g. looking for objects), they can have different complexity and specificity, and they can be performed by a single robot or multiple robots. Tasks have a well–defined set of constraints depending on the problem domain, the use of limited technologies, the scarcity of resources, and environment conditions. The most common ones are:

- Partial ordering, i.e. a task has to be completed before or after one or more others;
- Coupling, i.e. two or more tasks have be executed concurrently;
- Incompatibility, i.e. two or more tasks can not be concurrently executed;

- Time windows, i.e. a task has a duration condition (i.e. a deadline) to be met;
- Mobility interferences, i.e. robot's ability to perform a task is limited (e.g., it can not move in a narrow space due to its dimensions).

Such constraints model the functional relationships among tasks that can be expressed using a particular data structure. There are several planning approaches to generate such constrained decompositions that can be expressed by task trees. In [3], Doherty et al. presented a task specification language and an abstract distributed data structure, called Task Specification Tree (TST). Each node of a TST represents a task. A TST has a constraint network formed by a constraint model for each node and tree constraints, expressing the relations between the nodes. Hierarchical Task Networks (HTN) are used to represent various levels of task semantics, that have been extensively used for planning in AI domains, and have been imported to the robotic domain in abundant researches and applications.

In the literature [4], tasks may be also characterized by different parameters that are related to the way the task is executed, i.e. to non–functional attributes whose values are determined by the specific robot able to execute it. The most common parameters are:

- Cost, i.e., a measure of the effort made by the robot to execute a task, in terms of time to reach a goal, energy and resources consumed, and so on [5];
- Accuracy, i.e., a characterization of goodness of the task execution [6] performed by a robot (e.g. the map accuracy produced using a laser range-finder);
- Reward, i.e., a measure of the profit gained by the robot for performing a task [7];
- Priority, i.e., the task urgency required for its execution (higher priority tasks have to be executed before lower priority ones [8]).

When a complex task has to be executed by a team of robots with specific values of these non–functional parameters, the allocation of subtasks to the suitable robots, is an NP–hard decision problem. Market–based approaches for task allocation have received significant attention within the robotics research community [1], [9], in order to efficiently produce sub–optimal allocations [2].

Generally, in reply to a task request, robots may submit bids based on their abilities to perform the tasks and the highest (lowest) bid wins the assignment. Moreover, when more than one task can be assigned to each robot (and the evaluation of the robot non–functional parameters depends on its complete assignment), the robots can negotiate their respective assignments in order to optimize the task distribution. Typical allocation approaches adopt optimization algorithms based on some utility function. Indeed, the measure of utility is considered as a combination of these non–functional parameters, and it is used for evaluating the impact of the chosen tasks execution on the system performance. Utility functions can be based on different parameters, such as sensors-based metrics [10] or sophisticated planner–based measures [11] (multi–attribute utility theory). Utility represents a trade–off between accuracy and costs able to obtain an approximated result [10].

III. QoS Market–based Negotiation for Task Allocation

The problem of allocating subtasks composing a complex task with specific non–functional constraints to a team of robots is similar to select services for delivering a composition of services with Quality of Service (QoS) constraints. So, service–based computing technologies can be effectively integrated and utilized into robotic applications [12], [13].

In this work, we propose the use of automated negotiation as a mechanism to allocate subtasks to a set of robots, by allowing the individual robots to negotiate on the values of the non–functional parameters characterizing the complex task. These values may depend on dynamic conditions of each robot, such as its computational and physical resources at the time the allocation process starts, the number of tasks it was already allocated, the reward it gets, and so on. It is crucial to adopt allocation mechanisms that can take into account of such a variability. In addition, robots may decide to change the values of these non–functional parameters to accommodate some global requirements specified for the complex task in order to have the subtask assigned.

Here, it is assumed that a complex task is represented as a task tree, we refer to as an *Abstract Task Tree* (ATT), whose leaves are the subtasks composing it, we refer to as *Abstract Task*s (Ts). The complex task is required to be executed with specific QoS end–to–end requirements, referring to non-functional attributes of the complex task. The request is managed by an agent, we refer to as the *Task Allocator Agent* (TAA), responsible for finding an allocation of these subtasks to a team of robots providing them with values of the considered attributes that, once aggregated, meet the end–to–end requirements. Robots are modeled as task providers, i.e. "market vendors" of tasks characterized by QoS values accounting for these non–functional attributes. They negotiate these values with the TAA, so that a successful negotiation determines an allocation of subtasks to the robots able to execute them with suitable values of the non–functional attributes. Robots, referred to as *Task Provider Agent*s (TPAs), aim to win the negotiation so to obtain the allocation of the task, and they may change the QoS values they provide during negotiation according to their own negotiation strategies. In fact, while some values may depend only on the robot capabilities, others can be modified proactively by the robot. For example, it can dynamically modify the execution speed of a task or the accuracy of a provided sampling.

More specifically, each robot is modeled as composed of:

- a *deliberative* layer responsible for negotiating with the TAA the allocation of tasks;
- a *dispatching* layer responsible for scheduling the tasks allocated to the robot;
- an *execution* layer responsible for the actual execution of the allocated tasks.

In the following only the deliberative layer is addressed, while the other two layers are outside the scope of the work.

The negotiation mechanism adopted in this work is based on the one proposed in [14], a one–to–many iterative protocol, used to select services when a composition of services is required with specific end–to–end requirements. It allows a composer agent, acting on behalf of the consumer, to negotiate QoS attribute values of the functionality requested in the composition, both with different providers of the same functionality, and with the providers of different functionalities in a coordinated way, since it is not always possible to decompose these requirements in individual requirements for each functionality of the composition.

Here the same negotiation protocol is adopted, while new negotiation strategies are provided, specifically designed for the multi–robot task allocation domain.

The negotiation protocol consists of a number of *negotiation rounds* proceeding until either the negotiation is successful (i.e., all subtasks are allocated), or a *deadline* (i.e., a maximum number of allowed rounds) is reached. The deadline can be set according to the nature of the required tasks, or other criteria depending on the considered scenario. At each negotiation round, the TAA sends n *call for proposal*s (cfps), one for each available TPA, specifying the subtasks in the ATT to be allocated ($\{T_1, \ldots, T_m\}$), with $n \geq m$, and it waits for replies for a given time, known as the *expiration time* of a negotiation round. Each TPA_j provides an offer o_j containing the k QoS values $q_{i,k}$ for each of the T_i it is able to perform, and $m-k$ $NULL$ values for the T_i it is unable to perform. Such QoS values represent measures of non–functional parameters characterizing the way the robot can execute the task T_i (such as for example battery consumption, the speed, the accuracy, and so on). Hence, the TAA receives a set of offers $\{o_1, ..., o_n\}$, with n the number of TPAs.

At the first negotiation iteration, the TAA checks if there are offers for each required subtasks specified in the ATT. If there are no offers for all the required subtasks, it declares a failure since it is not possible to find a set of TPAs for all required T_i. Otherwise, it evaluates the received offers and the iteration number, and, according to the result of such evaluation, it performs one of the following actions:

1) if the aggregated QoS values of the received offers do not meet the global QoS requirements and the deadline is not reached (not final iteration), it asks for new offers by sending again n cfps, so starting another negotiation round;
2) if the aggregated QoS value of the received offers meets the global QoS requirements (final iteration), it selects the best set of offers, in terms of its own utility, and it accepts such offers and rejects the others, so ending the negotiation successfully.
3) if the deadline is reached without a success, it declares a failure to all robots that took part in the negotiation (final iteration).

A. Negotiation strategies

In order to decide whether to accept a set of offers, the TAA evaluates first if there is a combination of offers satisfying the global end–to–end requirements, intended as upper bounds for the aggregated offers values. The evaluation function used by the TAA is a solver of an Integer Linear Programming problem formulated as follows:

$$\sum_{j=1}^{n} x_{i,j} = 1, \forall i = 1, \ldots, m \tag{1}$$

$$aggr_i(\sum_{j=1}^{n} x_{i,j} \cdot q_{i,j,k}) \leq Q_k, \forall k = 1, \ldots, r \tag{2}$$

$$\sum_{j=1}^{n} x_{i,j} \cdot q_{i,j,k} \neq NULL, \forall k = 1, \ldots, r \tag{3}$$

Hence, there are $n \cdot m$ decision variables $x_{i,j}$, where i identifies one of the m Ts, and j identifies one of the n TPAs that replied to the cfp. Such variables assume value 1 if the $j-th$ TPA is selected for the $i-th$ T, 0 otherwise. Equation 3 verifies that agent j is able to execute the task corresponding to the $i-th$ T. Equation 1 verifies that exactly one TPA has to be selected for each T (such value can be changed in case task replication is required). Equation 2 evaluates that the global constraint for each considered non–functional parameter is met. Typically, additive (e.g., price and execution time) and multiplicative (e.g., reliability and availability) parameters are considered [15], so $aggr$ is either a sum or a multiplication over the number of Ts.

Once that combinations of offers that satisfy the ILP are found, the TAA selects the one that maximizes its own utility, that is evaluated as follows:

$$U_{TAA} = \frac{1}{r} \sum_{k=1}^{r} \frac{Q_{max'(k)} - aggr_i(\sum_{j=1}^{n} x_{i,j} \cdot \overline{q}_{i,j,k})}{Q_{max'(k)} - Q_{min'(k)}} \tag{4}$$

where, $Q_{max'(k)} = aggr_k(max(q_{i,j,k}))$ aggregates the local maxima of the offers received for the $i-th$ task, $Q_{min'(k)} = aggr_k(min(q_{i,j,k}))$ aggregates the corresponding local minima, and $\overline{q}_{i,j,k}$ are the values of one offer of the found combinations.

The adopted negotiation mechanism is one–sided, since the TAA cannot make counterproposals, but it only evaluates offers in an aggregated manner, while TPAs can send new offers. The TAA could formulate counterproposals relying on heuristics methods to decompose the end–to–end requirements into individual requirements. But this approach is not followed in the present work, since it increases the constraints to be satisfied by individual offers. In fact, when an aggregated value is required, it is possible that individual offers are acceptable when aggregated with the others, while they are not acceptable according to the chosen decomposition criteria.

TPAs are provided with strategies and tactics to generate offers to send at each negotiation round to the TAA for the subtasks they are able to execute. Several strategies and tactics have been proposed in the literature for different types of negotiation [16]. Here, a stochastic monotonic concession strategy is adopted for the TPAs, that models a cooperative behavior coming from the robot objective to obtain the allocation of the task.

IV. A Practical Example

In our reference scenario, we assume that robots may execute different type of tasks, and that they have the goal of maximizing the number of tasks to be executed according to the resources they have. For simplicity, we consider as a working example a global task ATT that consists in covering a specific total distance, and it is requested to be executed with a global completion time TCT_{req}, that represents the only constraint to be met. The task is decomposed in 3 subtasks $\{T_1, T_2, T_3\}$, each one characterized by a pair $< x_i, dir_i >$, where x_i is a distance in a specific direction dir_i. The tasks have to be executed sequentially. At each negotiation, the 3 subtasks will be assigned to 3 different robots if the negotiation is successful, i.e. if the sum of the execution times tct_i proposed by each triple of robots does not exceed the upper bound TCT_{req}:

$$\sum_{i=1}^{3}(\sum_{j=1}^{n} x_{i,j} * tct_{i,j}) \leq TCT_{req} \quad (5)$$

where n is the number of available robots. Hence, the negotiation is single–issue and the issue is additive.

It is assumed that each robot is able to cover a distance only in a specific direction, so it is able to execute only one of the 3 tasks, i.e. each subtask has to be allocated to a different robot.

The TAA initiates the negotiation by issuing the n call for proposals, one for each robot (or TPA), containing the reference to the 3 subtasks to be executed. Each robot R_j will reply providing an offer characterized by a triple of values, with a $NULL$ value for the subtasks it is unable to execute, and a task completion times $tct_{i,j}$ the robot R_j offers for the subtask T_i it is able to execute. The offered time $tct_{i,j}$ depends on the velocity $v_{i,j}$ the robot j decides to cover the distance of subtask T_i with.

Each robot R_j is represented by a state s affecting the offers it provides to the TAA for a specific task T_i. In particular, it is assumed that the state s of each robot is a n–tuple composed of the current battery level $b_s \in [bmin, bmax]$, depending on the current task allocation, a function f determining the battery consumption according to the velocity the robot offers to cover the distance of a task T_i with, the tasks $\{T_k, \ldots, T_h\}$ that it committed to execute and that are in its agenda, and a minimum allowed velocity $vmin$ that is the minimum functioning robot velocity (i.e., it depends on the robot and not on the task).

The initial state is defined as $s_0 =< b_{s0}, f, \{\}, vmin >$, and it is updated at the end of each negotiation. The maximum level of battery b_{s0} can be different for each robot, and it determines the maximum velocity the robot can execute a task with, when it is fully charged (a maximum initial value when the robot is not committed to execute any task), i.e. $b_{s0} = \alpha \cdot vmax_{i,j} \cdot x_i$. Hence, if the robot executes the task at the maximum velocity, it will use all its battery.

In order to reply to a cfp, the $j-th$ robot evaluates its current state in terms of the current battery level b_s, to determine the maximum velocity it can execute a task with $(vmax_{i,j})$, where

$$vmax_{i,j} = b_s / T_i. \quad (6)$$

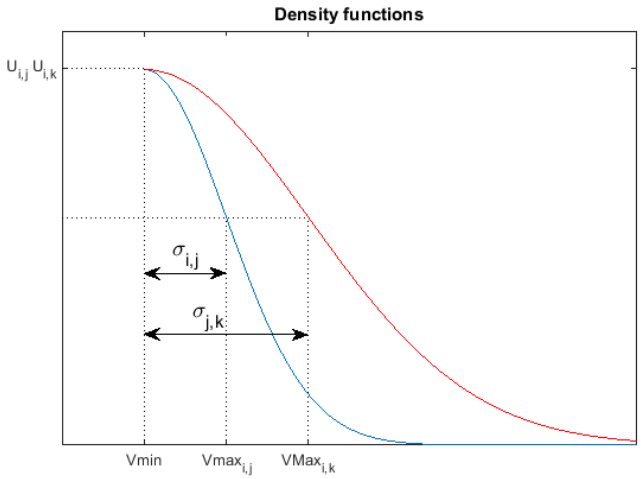


Fig. 1. An example of Gaussian functions for two robots j and k.

To model a stochastic behavior of robots taking part in the negotiation, the offers generation strategy is given by a Gaussian function, representing the probability distribution of the offers in terms of the robot utility, as proposed in [17]. In particular, the Gaussian function depends on the specific task T_i, and it accounts for the robot utility obtained when covering the distance of the task T_i at velocity $v_{i,j}$ ($tct_{i,j} = x_i / v_{i,j}$). As shown in Figure 1, the mean value of the Gaussian $U_{i,j}(vmin)$ represents the best offer the R_j may propose in terms of its own utility with the highest probability to be selected, and it corresponds to the maximum $tct_{i,j}$ it may provide the T_i with, i.e. the one obtained by executing the task at the minimum velocity possible. The rationale of this choice is that the robot prefers to use the less battery possible when providing the task, so that it can try to maximize the number of tasks it may be assigned, so its most convenient offer is the one that minimizes the battery discharge. The standard deviation σ represents the attitude of the R_j to concede during negotiation, and it is given by $\sigma_{i,j} = vmax_{i,j} - vmin$, if $vmax_{i,j} > vmin$, 0 otherwise. It takes into account the computational load of the robot in terms of the number of tasks it was assigned to execute. In fact, at each new negotiation the σ is decreased according to the battery consumption due to an assigned task since the $vmax_{i,j}$ depends on the remaining battery level, so generating a new Gaussian function to be used in the new negotiation. The value $vmax_{i,j}$ represents the reservation value for the robot, since it is the maximum velocity it can execute the task with its current level of battery. Furthermore, the robot can make an offer only if the remaining battery level allows it to cover the distance of the task T_i at the minimum velocity. The parameter σ varies from robot to robot providing the same task T_i, in such a way that the lower its computational load (in terms of available battery) is, the more it is available to concede in utility, and the lower its reservation value is.

The battery consumption is assumed to be linearly dependent on the velocity the robot can cover the distance x_i of the task T_i, according to the following equation:

$$f = \alpha \cdot v_{i,j} \cdot x_i \quad (7)$$

Whenever the robot R_j is selected for executing task T_i at the end of a negotiation, it adds it to its agenda and evaluates the remaining battery as follows:

$$b_{s'} = b_s - \alpha \cdot v_{i,j} \cdot x_i \quad (8)$$

The new state will be $s' =< b_{s'}, f, \{T_i\}, vmin >$.

In Figure 1, the functions associated to two different robots for the same T_i are reported. The best offer is the same for both robots (i.e., $U_{i,1}(vmin) = U_{i,2}(vmin)$), since it is assumed that the robots have the same minimum velocity, while their concession strategies are different according to their workload when the negotiation takes place. In fact, $\sigma_{i,1}$ is greater than $\sigma_{i,2}$ meaning that R_1 has a lower computational load than R_2, so it concedes more in utility than R_2.

At each negotiation round, each robot generates, following its Gaussian distribution, a new utility value corresponding to a new offer. If this value is lower than the one offered in the previous round and within the negotiation set, then the robot proposes the new value. The negotiation set is $[U_{i,j}(vmin); U_{i,j}(vmin + \sigma_{i,j}]$. If the new utility value generated is higher than that offered in the previous round, or it is outside the negotiation set, the robot proposes the same value offered in the previous round. This strategy allows to simulate different and plausible behaviors of robots that prefer not having a consistent loss in utility, even though by increasing the number of negotiation rounds the probability for the robot to move towards its reservation value increases.

A. Results

In our testing scenario, we evaluate the behavior of a set of negotiations carried out with 9 TPAs to allocate the complex task ATT composed of 3 subtasks T_1, T_2, T_3. There are 3 robots able to perform one of the three tasks.

Multiple negotiations are carried out since the TAA aims to allocate as many complex tasks as possible with the available robots. In this preliminary analysis it is assumed that each $x_i = 3m$, $dir_i = north, south, west$, and each robot can perform the task with a velocity v_j in a range between $vmin_j = 0.05m/s$ and $vmax_j = 0.3m/s$. The initial battery for each robot is $b_{s0} = vmax_j * x_i$, and the battery consumption is computed as defined in Equation 8 with $\alpha = 1$.

In such scenario, if a robot performs a single task with its minimum velocity ($vmin_j = 0.05m/s$), each task is completed in $tct_{i,j} = 60$ seconds, while the complex task requires $TCTreq = 180$ seconds. Hence, at the minimum velocity, each robot could execute 6 subtasks before consuming all its battery. Without considering the global constraint, 18 complex tasks (ATT) could be allocated to the robots. So, we set the maximum number of negotiations at 18. If we assume that robots perform the tasks at maximum velocity, each robot can execute only 1 task, so 3 complex tasks could be allocated to the available robots.

We simulate the set negotiations with 3 different global time constraints for the complex task ($TCT_{req1} = 100s$, $TCT_{req2} = 130s$ and $TCT_{req1} = 160s$), and, for each configuration, we perform 100 runs. The deadline of each negotiation is 100 rounds. Table I reports for each configuration the following information: average values, standard deviation, maximum and minimum values of the number of allocated complex tasks, the number of tasks allocated to each robot, the remaining battery at the end of the all negotiations, the time to complete each complex task, and the number of rounds for successful negotiations (i.e., the ones terminated with a complete allocation).

When a global constraint is required, the possibility to offer different values of the constraint parameter, allows to allocate a greater number of complex tasks with respect to a static allocation at a fixed velocity for each robot, in fact, at a fixed maximum or medium velocity, only 3 complex tasks could be allocated. As expected, when increasing the required TCT for the task execution, the number of complex tasks allocated increases, with a consequent increase in battery consumption. Moreover, let us note that the probabilistic distributions of offers lead to a global time value that is lower than the set constraint. Hence, different concession strategies may further improve these results on the constraints satisfaction. Finally, the number of rounds necessary to reach a complete allocation is smaller in the first negotiations, while it drastically increases in the final ones since it is more difficult to find a complete allocation once the battery level of the robots decreases. This is also shown by the high values of standard deviation for the *Rounds* number reported.

More specifically, this behavior is shown in Figures 2 and 3, reporting respectively, for one run of a complete set of negotiations with the $TCT_{req} = 130s$, the battery levels for each robot and the number of rounds for each negotiation. As shown in Figure 3, in the first four negotiations, robots easily find an agreement in 1 round; while, from the forth to the seventh negotiation the number of rounds in order to reach an agreement increases considerably leading to a lack of agreements from the seventh negotiation onward. This behavior is explained in Figure 2 where the trends of the battery level for each robot show that after the seventh negotiation the battery levels decreased for each robot. In addition the stochastic behavior of the offers generation leads to different final battery levels for different robots.

V. Conclusion

In this paper, we investigated the possibility to adopt heuristic approaches to find allocations of tasks to teams of robots when tasks are characterized by non–functional attributes, and the complete allocation has to meet global constraints expressed as end–to–end requirements of these non–functional attributes. A market–based negotiation mechanism where robots are modeled as task providers with negotiable non–functional parameters is proposed as an heuristic method to address the NP–hard task distribution problem.

This preliminary study showed that the adopted negotiation mechanism is a promising approach when trying to optimizing the number of complete allocations given a fixed number of available robots in the team.

The approach relies on the possibility for the robots to change the values of the parameters characterizing the tasks

Limit	**100s**				**130s**				**160s**			
	AVG	**SD**	**MIN**	**MAX**	**AVG**	**SD**	**MIN**	**MAX**	**AVG**	**SD**	**MIN**	**MAX**
Allocated Tasks	5.02	0.43	4	6	6.27	0.55	5	7	8.06	0.51	7	9
Tasks for each robot	1.67	0.48	1	3	2.08	0.55	1	3	2.68	0.58	1	4
Remaining Battery	37%	9%	2%	64%	30%	7%	19%	50%	21%	6%	7%	39%
Time	83	15	70	93	99	27	85	109	116	38	106	126
Rounds	9	16	1	97	5	10	1	97	4	7	1	100

TABLE I. NEGOTIATION RESULTS WITH TCT_{req} =100S, 130S AND 160S.

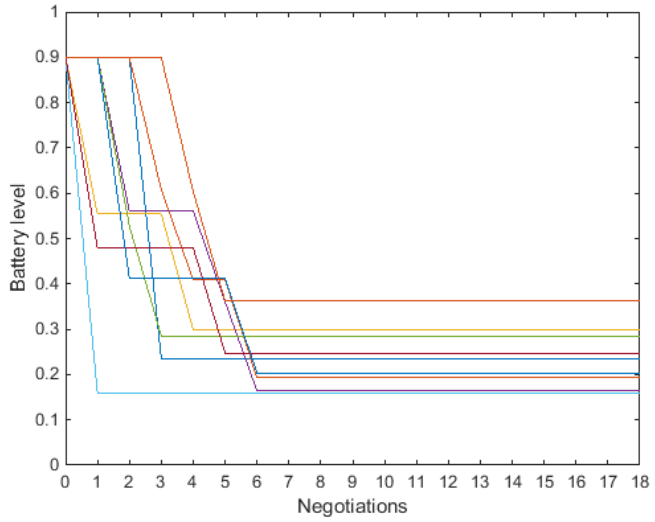


Fig. 2. Robot battery consumption trends for one set of negotiations.

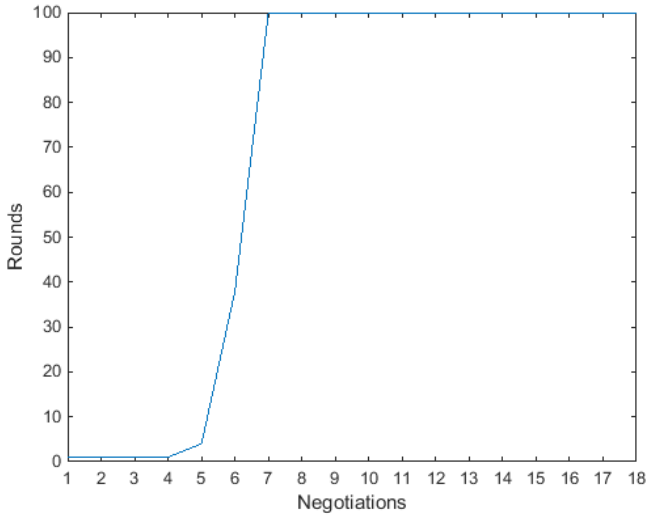


Fig. 3. Negotiation rounds for one set of negotiations.

they can execute dynamically, so modeling an innovative behavior of robots that can make decisions on how to perform a task.

Further investigation is necessary to optimize the battery level consumption of each involved robot that allows to still meet the constraints so leading to an increased number of complete allocations.

REFERENCES

[1] M. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, July 2006.

[2] R. Cui, J. Guo, and B. Gao, "Game theory-based negotiation for multiple robots task allocation," *Robotica*, vol. 31, pp. 923–934, 9 2013.

[3] P. Doherty, F. Heintz, and J. Kvarnström, "High-level mission specification and planning for collaborative unmanned aircraft systems using delegation," *Unmanned Systems*, vol. 1, no. 1, pp. 75–119, January 2013.

[4] L. M. Alejandro R. Mosteo, "A survey of multi-robot task allocation," Aragon Institute of Engineering Research, Tech. Rep., 2010.

[5] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and S. Jain, "Auction-based multi-robot routing," *Robotics: Science and Systems*, 2005.

[6] P. Stone and M. M. Veloso, "Task decomposition and dynamic role assignment for real–time strategic teamwork," in *Agent Theories, Architectures, and Languages*, 1998, pp. 293–308.

[7] A. Stentz and M. B. Dias, "A free market architecture for coordinating multiple robots," Tech. Rep., 1999.

[8] F. Antonelli, G.; Arrichiello and S. Chiaverini, "The null–space-based behavioral control for mobile robots," in *IEEE Computational Intelligence in Robotics and Automation (CIRA'05)*, 2005, pp. 15–20.

[9] R. M. Zlot and A. T. Stentz, "Market-based multirobot coordination for complex tasks," *International Journal of Robotics Research, Special Issue on the 4th International Conference on Field and Service Robotics*, vol. 25, no. 1, pp. 73–101, 2006.

[10] B. P. Gerkey, M. J. Matari, and M. robot Systems, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. of Robotics Research*, 2004.

[11] S. Botelho and R. Alami, "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement," in *IEEE International Conference on Robotics and Automation*, vol. 2, 1999, pp. 1234–1239 vol.2.

[12] S. Mokarizadeh, A. Grosso, M. Matskin, P. Kungas, and A. Haseeb, "Applying semantic web service composition for action planning in multi-robot systems," in *Internet and Web Applications and Services, 2009. ICIW '09. Fourth International Conference on*, May 2009, pp. 370–376.

[13] Y.-G. Ha, J.-C. Sohn, and Y.-J. Cho, "Service-oriented integration of networked robots with ubiquitous sensors and devices using the semantic web services technology," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, Aug 2005, pp. 3947–3952.

[14] C. Di Napoli, P. Pisa, and S. Rossi, "Towards a dynamic negotiation mechanism for qos-aware service markets," in *Trends in Practical Applications of Agents and Multiagent Systems*. Springer, 2013, vol. 221, pp. 9–16.

[15] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311–327, may 2004.

[16] P. Faratin, C. Sierra, and N. R. Jennings, "Negotiation Decision Functions for Autonomous Agents," *Robotics and Autonomous Systems*, vol. 24, pp. 3–4, 1998.

[17] S. Rossi, D. Di Nocera, and C. Di Napoli, "Normal distributions and multi-issue negotiation for service composition," *Advances in Soft Computing*, vol. 293, pp. 1–8, 2014.

Self-Organising UAVs for Wide Area Fault-tolerant Aerial Monitoring

Massimiliano De Benedetti, Fabio D'Urso, Fabrizio Messina, Giuseppe Pappalardo, Corrado Santoro
University of Catania – Dept. of Mathematics and Computer Science
Viale Andrea Doria, 6 — 95125 - Catania, ITALY
EMail: {m.debenedetti1983,fabiod}@gmail.com, {messina,pappalardo,santoro}@dmi.unict.it

***Abstract*—This paper describes an algorithm for the coordination of a flock of multirotor UAVs which cooperate in performing a wide area monitoring mission. The key characteristics of the proposed approach are the self-organising ability and the decentralisation. The flock has the basic task of covering a certain area of terrain and is able to self-organise in order to *(i)* find a coverage of the area which minimises mission time, and *(ii)* identify possible faults in one or more multirotors, taking care of performing a re-scouting of proper terrain regions in order to avoid loss of data. The algorithms to ensure flocking formation and area coverage are described. They are also validated by means of simulation approach, which is performed using an ad-hoc software tool.**

I. Introduction

The recent advent of small aerial vehicles, such as *multirotors*, made them an interesting solution also for professional applications rather than entertainment. Such multirotors have the great advantage of being easy to build; no particular skills in mechanics or avionics are required if compared with a small aeroplane which, instead, needs a particular care in wing profile, wing size and shape, tail size, position of the centre of gravity, etc. Indeed, these multirotors can be easily built by purchasing parts off-the-shelf (i.e. motors, propellers, frame, electronics) and then assembling them properly. They are basically controlled remotely by a human pilot, but they can also have the ability of flying autonomously using GPS hardware and suitable control algorithms.

Given the said characteristics, one of the most interesting applications of multirotors is the *aerial monitoring* of certain areas which cannot be accessed with other conventional vehicles: as a reference example, the authors are involved in a research project whose objective is the control of *landslips* and, since one of the monitoring strategies will be the gathering and analysis of *aerial images* in order to evaluate terrain movement, the employed solution will be indeed based on multirotors.

In addition to the cited advantages, multirotors have some drawbacks: the most critical one is the *autonomy*, which is in the order of 5 to 15 minutes (on average), not so high to make them suitable for wide areas. But another sensible problem is the *fault-tolerance*: should a aircraft fail during mission, it would be lost, together with its set of monitoring data.

To deal with the problems above, a classical solution is to employ a *set of several multirotors*, each one entailed with the objective of covering a specific portion of the monitored area; but, even if this solution solves the autonomy issue by parallelising the activities, it is not fault-tolerant, since a problem in one or more multirotors would provoke, in any case, a loss of a part of monitored data.

With these aspects in mind, the authors designed an algorithm, which is described in this paper, for coordinating a *flock of multirotors* during a wide area monitoring mission. The key characteristics of the proposed approach are the *self-organising ability* and the *decentralisation*. The flock has the basic task of covering a certain area of terrain and, to this aim, is able to self-organise in order to *(i)* find a coverage of the area which minimises mission time, and *(ii)* identify possible faults in one or more multirotors, taking care of performing a re-scouting of proper terrain regions in order to avoid loss of data. The proposed approach has been validated by using a software simulator implemented ad-hoc.

The paper is structured as follows. Section II introduces the minimum background behind this work and discusses related work. Section III introduces the use scenario, highlighting the basic characteristics and the requirements that in automated system must be met. Section IV deals with the proposed algorithms, which include flocking formation and area coverage. Section V describes the simulator implemented to test the proposed approach. Section VI reports the conclusions.

II. Background and related work

The *Area Coverage* research topic has the objective of studying solutions to cover a specific area with a certain number of robots, under a set of constraints, e.g. mission time minimisation, path optimisation and frequency cover for some point of interest. Area coverage can be classified into two main categories[6]:

- **Single Coverage:** the goal is to cover the target area until all the accessible points in the selected environment have been visited at least once.
- **Repeated Coverage:** the aim is to cover a set of points of interest in the selected environment more than once, for the entire mission.

The differences between the two aforementioned problems are closely related to the involved parameters [6]. The main parameters of Single Area Coverage are time, distance and number of visits for each point of interest. In other words, coverage must be completed while minimising the mission time, the distance covered by the robot and the frequency of visits. In Repeated Area Coverage the main parameters are the frequency of visits – to be maximised – , the length of paths

– to be minimised – and the workload distribution among the set of available robots, which has to be balanced.

The first concern of Area Coverage is the decomposition technique adopted to divide the area into fundamentals units which, in turns, are associated with a set of sub-tasks or paths. A significant approach is the Approximate Cell Decomposition [9], [11], in which the environment is divided into cells of equal size and shape, apart those cells that are partially occluded by obstacles or placed at the boundary of the area. A second approach is called Exact Cell Decomposition [10]; here the area is divided into a set of non-overlapping regions that represent the whole environment. The Boustrophedon Decomposition is a particular Exact Cell Decomposition on which the slice is a line, and the robot follows the intersection of the slice and the area to be covered with a simple back-and-forth motions. A cell in Boustrophedon decomposition is a region on which slice connectivity is constant and can be swept by a specific direction called Sweep Direction. As we discuss in the other sections of this work, our approach is similar to Boustrophedon decomposition.

Another aspect of Area Coverage is the algorithm used to cover the cells obtained from the area decomposition. As proposed in [3], the cooperation and coverage can be completely decoupled. In other words a coverage algorithm for a single robot can be extended to a cooperative setting by adding an overseer algorithm which takes incoming data from other robots and integrates it into the cellular decomposition [8]. The core of cooperation behaviours is the overseer algorithm that provides a task allocation among the robots. The computation of a specific function containing all the parameters involved in the global task [14] is thus distributed among the robots.

There are many different approaches for real time task allocation, and most of them are drawn from biology or from market-based approaches [4], [5]. Market-based approaches rely on a leader/auctioneer (a robot or an external operator) that calls an auction for each task to be assigned. The other robots will bids for each single task with an offer that represents a cost or an utility gain, and the best bidders are selected. Market-based approaches have gained popularity because they include an interesting, although implicit, fault management strategy. Indeed, whenever the communication channel between the leader (auctioneer) and one or more robots becomes unavailable, the auctioneer will call another bid for the execution of the incomplete task [1], [11]. Furthermore, if each robot broadcasts its information to the other robots, the system will be able to complete the goal even if the leader/auctioneer is affected by a fault. This additional strategy allows the remaining robots to choose another leader without loss of functionality.

The loss of communication among robots can occur in case of fault or when the distance between the robots became too large for the communication system. This second aspects becomes critical when the robots used for area coverage are UAVs or the coverage task is performed on indoor environments. In this kind of systems (UAVs fleet) a designated area must be covered with a certain level of efficiency and, at the same time, a constant connection among the UAVs must be maintained. Assuming that every UAV is equipped with a wireless transceiver (to broadcast and receive message from/to others UAV) a tree-based overlay network can be built and the area coverage tasks can be performed within network connectivity constraints [13]. Indeed, maintaining a complete knowledge of the environment requires a continuous broadcasting process among the robots that reduces the overall available bandwidth. Therefore, one of the important issues in this kind of environments (Area Coverage - UAVs fleet) is to find a trade-off between the broadcasting process and the minimum bandwidth required to perform/assign all the pending tasks.

III. System Model and Mission Requirements

We consider a system made of *(i)* a certain *physical area* that must be monitored, and *(ii)* a set of *mobile entities* with the ability of autonomously moving in the considered area and equipped with the sensors needed for the monitoring activity.

We assume that an inertial reference system exists, given in a 3D space, which includes, together with XYZ coordinates, also the *heading*. In the Earth reference system, these coordinates will correspond to *latitude*, *longitude*, *altitude* and *heading*.

On this basis, the area to be monitored can be identified by a polygonal in the XY plane of the considered reference system. Without loss of generality, we can assume that the monitored area has a *rectangular shape*[1] and therefore is identified by the coordinates of four vertices $Area = ((x1, y1), (x2, y2), (x3, y3), (x4, y4))$.

In this environment the monitoring entities live; we'll simply call them henceforward *agents*, even if, in the real scenario, they will be multirotor VTOL[2] machines. Each agent has the ability of moving in the 3D space with *four degrees of freedom*, one for each element of the coordinate system; we can define the *pose* of the i^{th} agent at time t as a 4-elements tuple:

$$pose_i(t) = (X_i, Y_i, Z_i, Head_i)$$

Each agent knows its pose by means of proper position sensor (e.g. GPS); it has also the ability of interacting with other agents by means of a *communication system* which, however, could be limited in the range (e.g. a low-power wireless system); therefore, two agents can interact to each other only if the (Euclidean) distance, in the given space, is less than the range supported by the communication system. Each agent i has also assigned a unique ID_i, which is mission-static, i.e. it does not change during a mission (for example the MAC address of the wireless card).

Each agent is also equipped with a certain *monitoring sensor*, which, given a certain Z altitude, is able to "monitor" (cover) a certain rectangle of the XY plane. As an example, if the sensor is a camera, we can determine the size of the captured pictures given the characteristics of the lens and the mission altitude. We can model this aspect by a function

$$sensor(z) = (ws_z, hs_z)$$

which takes a certain altitude z as argument and returns a tuple indicating the size of the rectangular area (the *sensor area*), in the XY plane, perceived by the sensor.

[1] If the real shape is different we can always consider the smallest rectangle which include the real area.

[2] Vertical Take-Off and Landing

In such a system, a *mission* is defined as:

$$Mission = (Area, \overline{Z}, N), N > 0$$

where:

- $Area$ is the specific area to be covered, defined as indicated above.
- $\overline{Z}$ is the altitude at which the agents must operate.
- N is the number of agents employed.

The agents must perform the mission in a complete autonomy, by cooperating with one another, and with the overall objective of

1) *to ensure the covering*, with their sensors, of the given area;
2) *to minimise* the mission time; and
3) *to avoid/minimise* the amount of overlapped data (area *over-covering* avoidance).

If we suppose that each agent i, at the end of the mission, has covered the area $SArea_i \subset Area$, the requirements above can be expressed as:

$$\cup_{i=1}^{N} SArea_i = Area \quad (1)$$

$$\cap_{i=1}^{N} SArea_i \ is \ minimal \quad (2)$$

IV. The Area Coverage Algorithm

Given the system model described so far, the area coverage approach proposed in this paper is composed of two basic parts: *(i)* a strategy to make agents fly in a certain *flocking formation*, and *(ii)* a technique to make the flock cover the given area, according to the aforementioned requirements.

A. Overlay Construction Algorithm

To support flock formation and information spreading, an overlay network among the agents is built. The aim is to let each agent know who are the other agents of the set and to make them able to exchange information about their position.

The algorithm used to construct the overlay network is based on a simple mechanism which exploits gossiping. Each agent $\overline{k}$ maintains an *agent database*, called $ADB = \{(ID_i, pose_i, HOP_i^{(\overline{k})}, TS_i)\}$ which stores, for each other agent $i \neq \overline{k}$, the agent identifier ID_i, which is also a primary key for the database[3], the current agent position $pose_i$, the number of hops $HOP_i^{(\overline{k})}$ which separates agent $\overline{k}$ to agent i, and the timestamp TS_i at which the pose information $pose_i$ has been generated.

Each agent, with a given periodicity, samples its position $pose_{\overline{k}}$, retrieves the sampling timestamp[4] $TS_{\overline{k}}$, and sends, in broadcast, the sampled information together with its ADB:

$$ADB \cup \{(ID_{\overline{k}}, pose_{\overline{k}}, 0, TS_{\overline{k}})\}$$

Since the wireless system has a limited range, the broadcast message will reach only some of the agents of the set. Each agent, on the basis of the reception of such a message, performs a comparison between its local ADB and the database received (henceforward called $RADB$); in particular, the following operations are performed:

- first the *hop count* field of each tuple of the $RADB$ is incremented by 1;
- then, for each tuple of the $RADB$, the tuple with the same ID is searched into the local ADB;
- if an associated tuple is **not found**, the $RADB$ tuple is added to the ADB;
- if the associated tuple is found, the timestamps are compared;
- if the $TS^{(RADB)} > TS^{(ADB)}$, the received information is *fresher*, so the local information in the ADB is updated;
- otherwise, the information present into ADB is kept and the received one is discarded.

This process is executed iteratively and periodically on each agent therefore, after some steps, each agent will know who are the other agents and what their "distance" is, in terms of *hop counts* and according to the transmission range of the wireless system.

Since agents always send their ADB, there could be the case in which, at a certain time instant, two different pose information, but relevant to the same agent $\overline{k}$, are travelling in the network, e.g. the one freshly generated by the agent $\overline{k}$ itself and the one previously owned by other agents and sent in their ADB: in this case, the use of timestamps permits an agent to accept the fresh information and discard the old ones, thus ensuring data consistency.

B. Ideal Flock Shape and Flocking Formation

Flock formation approaches are traditionally based on algorithms which, by running on each agent of the set, continuously perform the following steps:

1) retrieve the position of all (or a set of) the other agents;
2) evaluate a *speed* and the *direction* to follow, on the basis of some precise *rules*;
3) apply the computed speed and direction.

According to the literature [2], [7], [12], the basic rules which can drive a flock formation have to ensure the properties of *separation*, *alignment* and *cohesion*.

The first rule ($R1$, *separation*) says that if two agents are too close to each other, they must head to opposite directions. The second rule ($R2$, *alignment*) makes an agent orientate towards the average heading of the neighbour agents. The third rule ($R3$, *cohesion*) drives an agent towards a position which is the average of the positions of the neighbour agents. In addition, when the flock is used in the exploration of an area, a fourth rule is included ($R4$, *exploration*) which lets an agent drive towards the nearest unexplored zone.

All of these rules are properly weighted on the basis of their importance in forming the flock and also in order to avoid

[3] two tuples with the same ID cannot exist

[4] Timestamps are *local*, that is, they do not need a global synchronisation mechanism.

collisions and partitioning, and they must be properly adapted when a certain flock shape is desired (indeed the basic rules reported in literature often provoke flock partition and do not ensure a specific flock shape [7]). Such aspects are particularly important for our application: partition avoidance is indeed an obvious feature, while ensuring a certain flock shape means to avoid that area portions are not over-covered.

On this basis, the ideal flock shape we consider is based on a linear placement of agents along a *formation line* which is perpendicular to direction of flight, as it is shown in Figure 1. In this way, if the distance between two close agents is kept—as equal as possible—to the width of the sensor view $ws_{\overline{z}}$, the flock can monitor the area by means of slices of size $(N \cdot ws_{\overline{z}}, hs_{\overline{z}})$; since no agent is placed behind another one (in the direction of forward flight), this flock shape ensures that, during a linear flight, a certain area is not over-covered.

Fig. 1. Ideal flock shape

To make agents form the desired flock shape, our algorithm proceeds as follows.

At each step of the iteration, the agent elects a *leader*; it will be the agent with the *lowest* known ID present in the ADB[5]. Since we suppose that each agent always receives the IDs and positions of all the other agents with the overlay construction algorithm, we can assure that all agents will select the same leader. Indeed, when the overlay network construction is not in a steady-state (this transient condition occurs when the system is started and when an agent has a failure and thus abandons the flock), not all the agents may choose the same leader; however, as it will be clear in the following, this does not influence the validity of the flock formation: also flock shape will be affected by the same transient which will lead to a regime when overlay construction is a steady-state. We have to also highlight that the presence of a leader *is not* a central point of failure and does not affect the fault-tolerance of the proposed schema: should the leader fail, a new leader is elected soon and the flock can thus continue its task; as it will be detailed in Section IV-D, the transient between leader failure and new leader election does not have a significant influence on the correctness and completion of the coverage task.

The leader has the role of establishing the *formation line*, which is set as the line perpendicular to the current heading of the leader itself. The following flocking rules are applied.

The first rule, $R1$, *separation*, behaves as follows:

- The distance d_i to each other neighbour agent i is computed.
- If d_i is less than a *hard threshold* DH, no motion is applied (the UAV remains in hovering for the current iteration); this is required to avoid any possible collision.
- If d_i is less than a *soft threshold* $DS > DH$, a repulsion force is generated for the agent, whose intensity is proportional to d_i but with an heading *always parallel* to the formation line.

The *alignment* rule, $R2$, behaves by making the agent to orientate with the same heading of the neighbour which is *the nearest to the leader* (w.r.t the overlay network known). The alignment is performed by means of a yaw rotation.

The *cohesion* rule, $R3$, behaves by making the agent to move closer to the neighbour which is, also in this case, the nearest to the leader in the flock. This motion is performed by means of a translated flight. Obviously, if the agents are *too close*, rule $R1$ will ensure the right separation.

The behaviour of these rules is shown in Figure 2.

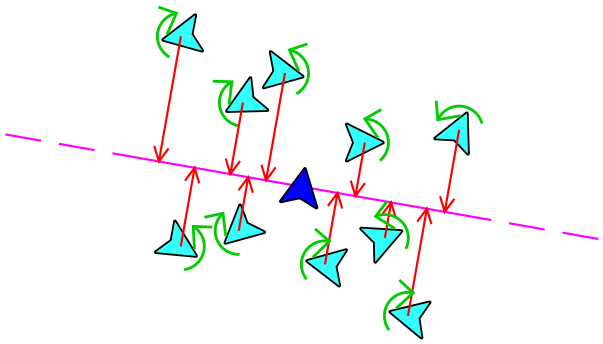


Fig. 2. Flock formation with rules R1, R2 and R3

C. Flock Driving and Area Coverage

Once the flock has been formed, its main task is to fly over the given area in order to ensure its coverage. To this aim, a *path planning* algorithm is designed. The algorithm is mainly driven by the leader and is based on the following schema:

1) All the agents spread, over the flock, the information related to the area parts already covered by each of them.
2) The leader gathers such information and computes an *optimal path*.
3) The leader starts to follow such a path thus driving the overall flock.

Such steps are executed iteratively, so that the path can be continuously adjusted on the basis of the real conditions and situations which can occur during the flight, e.g. wrong positioning of an agent due to too much wind, failure of a certain agent, etc. The key aspect of the approach lies in the way in which information on covered parts are represented, sent and processed by the agents.

As modelled in Section III, the overall area to be covered is represented by a rectangle, as well as the single portion which can be monitored by a sensor. Without loss of generality, we can perform a change of reference system and consider the rectangle area always hooked at the origin of a XY Cartesian system and with the proper width and height. In this rectangle,

[5]If the ID is not a numeric type, we suppose that a metric can be always determined in order to apply the concept of "lowest".

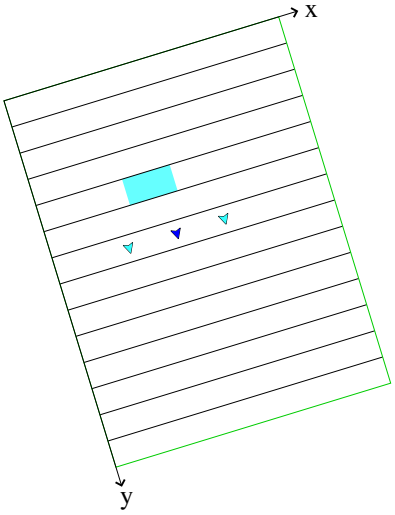


Fig. 3. Subdivision in stripes of the area to be covered

flight will be performed by keeping the *formation line* parallel to the X axis.

We subdivide the area into *stripes* by discretising the Y axis[6] (see Figure 3); the height of each stripe is equal to $hs_{\overline{Z}}$ that is the height of the rectangle covered by the sensor at the mission altitude $\overline{Z}$. We represent each stripe with a natural number, called $StripeID$ starting from 0. Since the area to cover is known at start-up time, stripe subdivision and numbering can be done before starting the mission.

Each agent, during its mission, holds and updates an *Area Parts Database* containing all the area parts already monitored. Since the parts are grouped in stripes, the database will contain a set of tuples structures as:

$$(StripeID, \{(XS_1, XE_1), (XS_2, XE_2), \ldots, (XS_n, XE_n)\})$$

Here, $StripeID$ identifies the stripe, while XS_i and XE_i indicates the stripe parts which has been already covered. The set of stripe parts is *ordered*, i.e. $XS_1 < XE_1 < XS_2 < XE_2 \ldots$ The size of each part $XE_i - XS_i$ is not forced to be equal to $ws_{\overline{Z}}$, i.e. the width of the sensor area: indeed, when several *close* sensor areas are covered, a *stripe union operation*, for such areas, is performed, as depicted in Figure 4.

The path planning approach is based on such stripe coverage data. The leader periodically starts a *distributed aggregation query* by sending, in broadcast, a data packet made of:

- Its node identifier ($LeaderID$);
- A *sequence number*, which increments each time a new query is started;
- An *agent ack field*, represented as a bitmap in which each bit indicates whether the correspondent agent has answered;
- The Area Part Database with the stripe parts already covered by the leader.

[6]We could also use X axis for discretisation, instead of Y, this choice does not affect the validity of the algorithm but only the way in which the path is then computed by the leader.

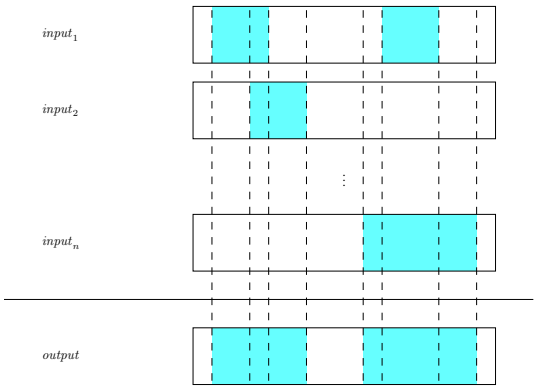


Fig. 4. Stripe Union Operation

Each agent receiving this query[7] performs the following operations:

- It first check that the $LeaderID$ correspond the its leader knowledge; if there is a mismatch, the packet is discarded (this may happen when a leader fails and a new leader is elected).
- The packet is also discarded if its sequence number is less than the last sequence number known by the agent. This is required to avoid duplicated/old data.
- The Area Part Database present in the packet is merged (through a stripe union operation) with the same database held by the agent and the result is replaced in the relevant packet field.
- The proper bit in the *agent ack field* is set.

The so-modified packet is then re-transmitted in broadcast. Re-transmission is however not performed when an agent identifies that it has *the same* knowledge of all its neighbours about the covered stripe parts (unless a new aggregated query, with a new sequence number, is started).

Sooner or later, the aggregation query packet will return to the leader, containing an updated list of stripes: if the *agent ack field* signals that *all nodes* have been answered, the list of covered stripes received by the leader can be considered completed and the next step of the path planning can be performed. To this aim, the leader computes two possible paths by following the external borders of the areas not yet covered (see Figure 5) and then chooses the one which is the closer to the flock. The chosen path is then *smoothed* (see Figure 6) and then *followed* by the leader itself: all the other agents will behave in consequence, by following the leader according to the rules $R1$, $R2$ and $R3$ illustrated above.

[7]Notice that not all the nodes receive the query, even if it is transmitted in broadcast: remember that we are considering a wireless transmission system, on each node, which could not have the adequate power to reach all the agents of the flock

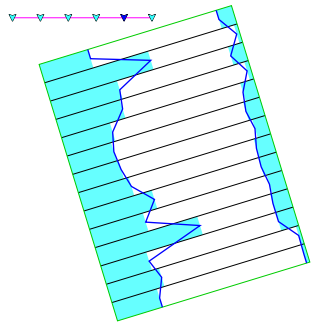


Fig. 5. Possible Paths Computed by the Leader

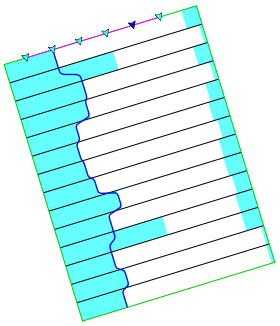


Fig. 6. Path Smoothing

In performing its flight, each agent will activate its sensor each time it will pass over a not yet monitored area part, properly updating its Area Parts Database.

As the mission proceeds, the various parts of the area to be monitored will be properly covered and the mission will end when the leader will recognise that the *whole area* has been covered: in this case, the agents can return to the base station and delivered the acquired data.

As it can be understood from the description, the proposed solution is indeed a mixture of a distributed and centralised approach. From the overall point of view, the solution is completely distributed, since knowledge of the entire system, as well as mission state, is not held by a single agent but properly spread over the various entities. Surely certain activities, like path planning, are instead centralised; indeed, this is a choice by design: in order to ensure total area coverage, flight time minimisation and over-coverage avoidance, the execution of path planning in a *centralised entity* can guarantee that an optimal solution can be found. However, such a centralised entity is not prefixed but continuously chosen during the flight on the basis of the said strategies. Therefore, as it has been introduced in Section IV-B, the use of a central entity does not have to be considered as a reliability issue, but the fault-tolerance is assured by the mechanism described below.

D. Leader Role and Fault Tolerance Aspects

When the leader fails, it stops transmitting its packets and thus the following situations will occur.

First of all no packet related to overlay construction, and coming from the (faulty) leader will transit over the network. After a certain timeout, each agent will recognise that the entry, in the ADB, relevant to the faulty leader *is no more updated*: so it can delete the entry, assuming that the leader is no more working, and elect a new leader. This process will be executed by all the agents, but using different reaction times tied to the fact that information spreading on the network, based on gossiping, is not immediate; therefore, it will be exist a *temporary interval* in which all agents *do not agree* on which node is the leader; however, this condition is not a problem: basically, agents do not directly follow the leader but their neighbours, therefore they will continue to stay in formation, naturally compensating the latency required to elect the new leader and thus continuing the mission.

A similar situation occurs when a non-leader agent fails. Also in this case, the relevant entry in the ADB will, sooner or later, become *too old* and thus discarded. The new overlay network will be built without the faulty agent so that a consistent knowledge of the new formed system is obtained, and the system itself can continue to behave correctly.

V. Simulation Results

To validate the approach proposed in this paper, a graphic software simulator has been developed, a screen-shot of which is reported in Figure 7. The simulator has been implemented in C++ using the Qt graphic libraries and allows a researcher to setup a mission, inspect its evolution and evaluate its correctness. The initial setup consists of establishing the number of agents and the area to be covered, as well as mission parameters, such as the sensor area size and the maximum range of the transmission system. After setup, the simulation can be started: first the agents self-organise in the flock and then they start the area monitoring mission. During the mission, a user can:

- dynamically add and remove agents (agent removal can be used to simulate faults);
- modify the area to be explored;
- inspect the databases of each agent;
- add/modify packet-loss percentage in order to simulate problems in the wireless system.

Simulation is time-driven: at each "time tick", a service procedure is executed for each agent, which performs all the activities described in this paper. Subsequently, the graphic display is updated by drawing the agents in their proper position and heading, together with some lines which represent the overlay network created; the area portions covered during the mission are then progressively coloured.

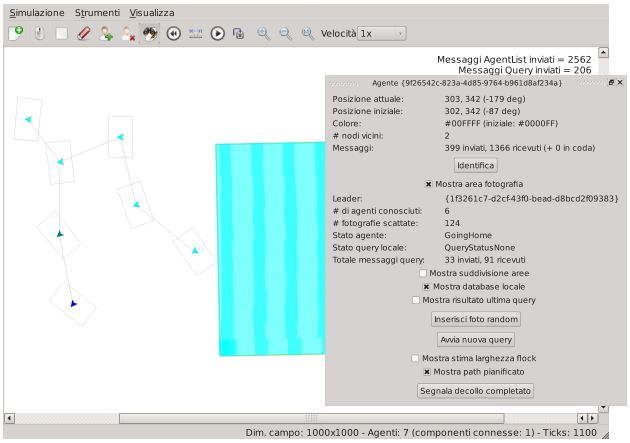


Fig. 7. Screenshot of the simulator

VI. Conclusions

This paper has described a decentralised approach to drive a set of UAVs in performing an *aerial monitoring mission* of a wide area. The approach is based on two algorithms. The first one drives agents to form a *flock* with certain given characteristics. The second one allows agents to follow a certain path which ensures the overall coverage of the area to be monitored.

The approach has been validated by means of a simulation study, performed using an ad-hoc software tool. The simulation campaign performed showed that the approach is able to drive UAVs in executing the right monitoring task, ensuring fault tolerance and area coverage, also minimising over-covering and thus mission duration.

Future work will aim at making the simulator more realistic by including the knowledge of some real physical aspects, such as the real area size, flight speed and UAV dynamics, in order to perform a proper evaluation of the real mission time. The implementation on real UAVs will be then a further step.

VII. Acknowledgements

This work is partially supported by projects PRISMA PON04a2 A/F and CLARA funded by the Italian Ministry of University.

References

[1] M. Bernardine Dias, M. Zinck, R. Zlot, and A. Stentz, "Robust multirobot coordination in dynamic environments," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4. IEEE, 2004, pp. 3435–3442.

[2] N. Bouraqadi and A. Doniec, "Flocking-based multi-robot exploration," in *Proceedings of the 4^{th} National Conference on Control Architectures of Robots*, Toulouse, France, 2009.

[3] Z. J. Butler, A. A. Rizzi, and R. L. Hollis, "Complete distributed coverage of rectilinear environments," 2000.

[4] M. B. Dias and A. Stentz, "A market approach to multirobot coordination," DTIC Document, Tech. Rep., 2000.

[5] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[6] P. Fazli, A. Davoodi, and A. K. Mackworth, "Multi-robot repeated area coverage," *Autonomous robots*, vol. 34, no. 4, pp. 251–276, 2013.

[7] Gbor Vsrhelyi, Csaba Virgh, Gerg Somorjai, Norbert Tarcai, Tams Szrnyi, Tams Nepusz and Tams Vicsek, "Outdoor flocking and formation flight with autonomous aerial robots," in *Submitted for publication to the IEEE/RSJ International Conference on Intelligent Robots and Systems*, ser. IROS '14, Chicago, IL, USA, 2014.

[8] C. S. Kong, N. A. Peng, and I. Rekleitis, "Distributed coverage with multi-robot system," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 2423–2429.

[9] J.-C. Latombe, "Approximate cell decomposition," in *Robot Motion Planning*. Springer, 1991, pp. 248–294.

[10] ——, "Exact cell decomposition," in *Robot Motion Planning*. Springer, 1991, pp. 200–247.

[11] I. Rekleitis, A. P. New, E. S. Rankin, and H. Choset, "Efficient boustrophedon multi-robot coverage: an algorithmic approach," *Annals of Mathematics and Artificial Intelligence*, vol. 52, no. 2-4, pp. 109–142, 2008.

[12] C. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14^{th} Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '87. New York, NY, USA: ACM, 1987, pp. 25–34.

[13] J. Schleich, A. Panchapakesan, G. Danoy, and P. Bouvry, "Uav fleet area coverage with network connectivity constraint," in *Proceedings of the 11th ACM international symposium on Mobility management and wireless access*. ACM, 2013, pp. 131–138.

[14] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *AAAI/IAAI*, 2000, pp. 852–858.

A Case Study on Goal Oriented Obstacle Avoidance

Pasquale Caianiello and Domenico Presutti
DISIM
Università dell'Aquila
Italy
Email: pasquale.caianiello@univaq.it, domenico.presutti@gmail.com

***Abstract*—We report on several test experiments with a mobile agent equipped with an artificial neural net control to achieve a basic route direction goal reflex in a 2-dimensional environment with obstacles. A real assembled *4tronix Initio* robot kit agent is reproduced with its sensor and motor characteristics in a virtual environment for experimenting and comparing the behavior of its artificial neural net control with two different learning approaches: A standard supervised error back propagation training with examples, and an unsupervised reinforcement learning with environmental feedback.**

I. Introduction

Obstacle avoidance is a basic task for mobile agents. Commercial and research applications address the problem by using state of the art adaptive techniques and mathematical modeling. In this work we confronted with the task of using a simple neural net architecture to equip a real *4tronix Initio* [22] robot kit agent with a basic obstacle avoidance control. Its neural net would take inputs from a pre-processing of the sensor information of the robot, and provide an output to control its motor actuators. The preliminary aim of our project, as reported in this paper in its start-up phase, is to construct the virtual counterpart of the *4tronix Initio* robot agent, that will let us perform fast and reliable test experiments of possible control strategies at the reflex proactive level with real time response.

As a result we identified a simple artificial neural net architecture and a pair of training strategies that would let the agent show simple adaptive/reactive capabilities in avoiding obstacles, while achieving a given target position goal. The agent's control neural net model is deliberately kept at a reflex level state, the perceptron basic input/output transduction, with no high level onthologies or primitives for describing the environment, no environment model or map acquisition capabilities, and no planning ability of any sort. The environment only exists for the pattern that it induces on the agent sensors at a given position and orientation. The agent will have to react by issuing control settings to its motor components, taking into account its given goal.

II. Tools and Methods

A. The agent

The agent hardware is a *4tronix Initio* [22] robot kit controlled by a Raspberry Pi B+ [25] computer that emulates the artificial neural net and performs low level sensor acquisitions and issues motor control primitives to a PiRoCon v.2 motor controller. The agent is equipped with a pan-tilt HC-SR04 ultrasonic sensor that acquires a panoramic of the environment, two infrared front mounted close range proximity sensors and a GY-80 multi-chip module that integrates a 3-axis gyroscope, a 3-axis accelerometer, and a 3-axis digital compass used for determining the agent absolute orientation.

B. The virtual agent and environment

The virtual environment is constructed as a bit matrix of dimension $m \times n$. Each bit represents a virtual position for the agent. A bit is set to 1 if the position is blocked, there is an obstacle, and the position cannot be occupied by the agent. The *goal* area of the environment is a circle identified by a center position and a boundary radius. The virtual agent emulates the real hardware in performing its basic control commands to the hardware controllers. The basic functioning cycle has three steps: Sample sensors, Compute orientation and velocity, Run for a time unit.

C. The artificial neural net control module

The artificial neural net (ANN) is emulated through the use of the Neuroph [24] Java framework used to implement a 3-layer fully connected feed-forward net with 31 input units, 62 hidden nodes, and 10 output units as depicted in fig 1. All artificial neurons in the net are sigmoid. The 31 input units collect the infrared proximity information, the distances in 9 predefined pan positions measured by the ultrasonic sensor (normalized in the unity interval), the distance from the agent to the center of the goal area, and the computed orientation angle with respect to the given direction goal, and represented into 18 binary direction intervals of 20° to comprise the whole 360° range. The output units consist of 9 binary orientation directions (the front 180°) and one real in the unity interval to represent the distance to be covered. So the control protocol will interpret the ANN output by setting the agent to the orientation given, and let it run for the given distance.

The experiments are described when the ANN is trained according to two different protocols: Supervised error Back Propagation (BP), environment reinforcement learning (RL).

1) The Supervised error back propagation protocol: The BP protocol was experimented with training sets (TS) constructed in two different ways, the first (BPp) by sampling the control choices of a human pilot, the second (BPh) by recording the behavior choices of a heuristic evaluation function in a wide range of enumeration of input sensor patterns. The use of BPh allowed the easy construction of much larger virtual TS, while TS construction with BPp required synchronized reading

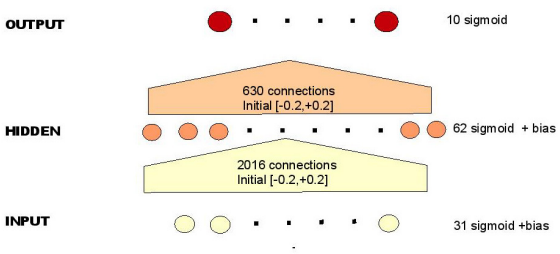


Fig. 1. The perceptron architecture for proactive reflex input/output transduction

the agent sensors and sampling the human pilot, who on the other hand was allowed to comprise higher level cognitive faculties, and was allowed to look at the environment map while making his decision.

To preserve generalization capabilities and avoid overfitting of the ANN, the training process was stopped at predetermined network mean square error limit values. For this purpose, each training sets were split in two subsets of training subsets and test subsets, containing respectively 90% and 10% of the original TS samples. Optimal limit network error values were determined by training the network on training subsets and testing network response on the test subsets: when the error on the test subsets became stationary or increasing, training was paused and finally completed by using the full TS and the minimum network error limit.

After the training process, the trained ANN's were tested on the agent control system and some critical aspects emerged as of the dimensional insufficiency of the TS obtained with BPp when compared to the dimension of the inputs state configurations. It did, in fact, bring about insufficient network output response polarization on new input patterns and a random-like behavior of the agent in specific configurations. On the other hand the ANN's trained with TS constructed with BPh was often trapped in stationary or cyclic behavior in sub-optimal positions with respect to the navigation goal. In consideration of the complementary critical aspects described for the BPp and BPh cases, a third instance of the ANN was trained with an incremental training process that combined both pilot driving and heuristic evaluation TS. In this case the learning process consisted in two training phases. In the first phase, the BPh TS was used for training the ANN for a small number of training epochs in order to give the network base response capabilities in covering a wide range of input configurations. In the second phase, the BPp TS was used until training completion. As the incremental trained network was finally tested on the robot, the critical behaviors were relieved.

All the test results reported in the following are obtained by running the net configuration obtained at the end of the training process as described, with a sample environment problem.

2) The Reinforcement Learning protocol: The same net architecture has been used with an unsupervised reinforcement training protocol Q-learning [15] with reward/reinforce function taking into account distance from goal, runs length, and route declination from goal direction. Reward was corrected by the Q function [15] to take into account future effects of actions

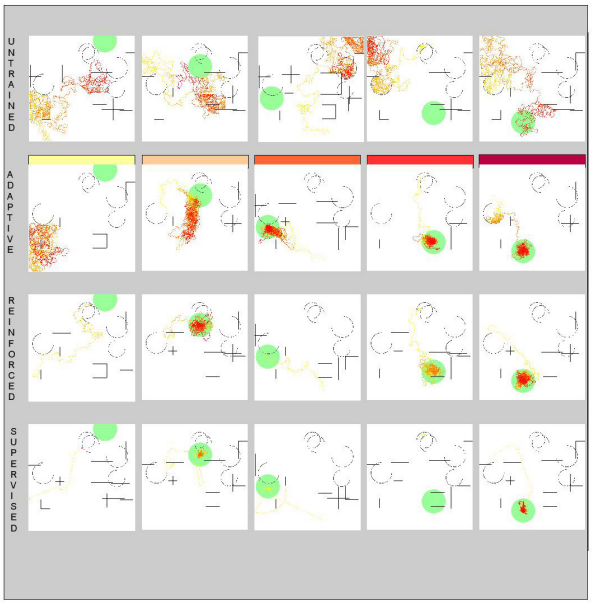


Fig. 2. Simulation Results, Environment complexity 3. Each snapshots records the trajectory of the corresponding row agent. Trajectory position points have a time scale color starting from yellow and going to darker red as time passes. Goal area is green. See text.

and to loosen excessive local/opportunistic behavior. The net was trained on a collection of problem samples with random selection of obstacles configuration, starting position and goal area to obtain the trained net that was used in the comparative experiments where its behavior is sampled at different levels of maturation.

III. Experimental Results

The experimentation was performed after training the ANN both with the BP protocol and with the Q-learning protocol leading to two net configurations named SUPERVISED (SU) and REINFORCED(RE) respectively.

The RE network was trained for 4000 learning sessions of 100 base cycles. For each session a random start and target is assigned in a randomly generated environment. Main learning parameters are set by an empirical optimization process to the following values: Learning rate= 0.036, Future actions discount factor g= 0.24, and Stochastic action selector temperature T=4. A high temperature T value is necessary during the learning process to maximize reinforcements. The T value is subsequently lowered on tests to 0.4 value to appreciate neural network response and control system behavior.

On the other hand, SU is trained for one learning epoch on the BPh TS, containing 1.152.000 training samples, and resulting in a 0.14 mean square error after training. The following 2076 training epochs are performed with the BPp TS, with 400 training samples. Mean square error at the end of the training process is 0.07. Both BPh and BPp training sets are generated on several base generation sessions of 50 iterations each. Network weights are initalized with random values in the [-0.02, 0.02] interval.

After training the nets are tested on a battery of several tests

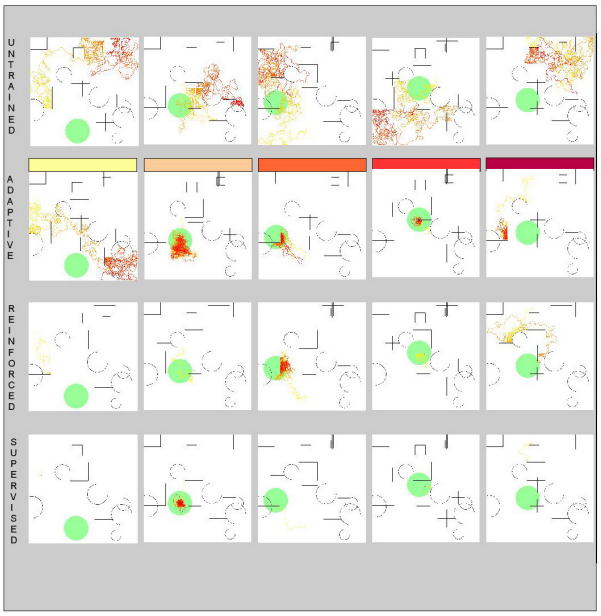


Fig. 3. Simulation Results, Environment complexity 7. Each snapshots records the trajectory of the corresponding row agent. Trajectory position points have a time scale color starting from yellow and going to darker red as time passes. Goal area is green. See text.

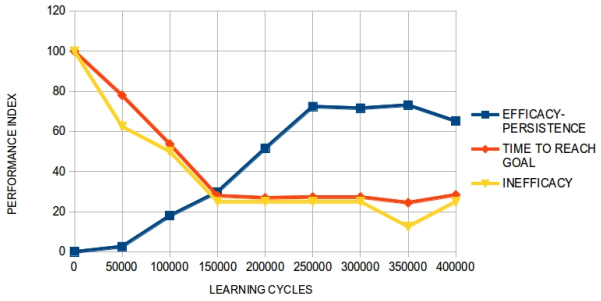


Fig. 4. Performance of RE while training

on the same environment, organized in groups of increasing environment complexity.

Their behavior, while trying to achieve the target area goal, is sampled as reported in fig. 2 and in fig.3. The whole experiment ranges over 8 batteries of 10 random problems in the same environment, for 10 different random choices of start/target positions. In the figures we show the outcome of just two test batteries, where each row collects an order preserving under-sampling of five out of the ten snapshots in the battery, each representing the trajectory of the agent behavior in the same environment. Snapshots of RE and SU behavior in test problems are in the bottom two rows. The top two rows records the agent behavior when controlled by an UNTRAINED (UN), randomly selected net configuration, and when controlled by a Q-learning ADAPTIVE (AD) net. AD is always in learning phase, it is randomly initialized at the first test in the battery, and retains its net configuration through subsequent tests in the battery. As AD gets trained while testing, it is expected to converge to the one of RE.

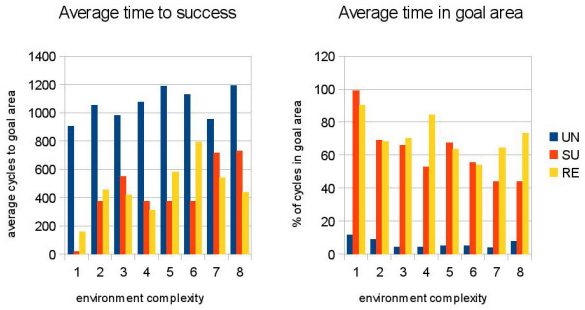


Fig. 5. Comparative statistics of reflex effectiveness in reaching the goal

Each snapshots records the trajectory of the corresponding row agent. Trajectory position points have a time scale color starting from yellow and going to darker red as time passes. Test problems in a row are presented in the same order as they were performed. The order is irrelevant for UN, RE, and SU, but AD's behavior changes (and improves) while solving a problem. For subsequent tests in the battery, AD's behavior change gives an idea of how a Q-learning net evolves to a finally trained RE from an UN.

Figure 5 reports a statistics over the tests performed in a single test time of 1600 iterations. The first index indicates *ineffectiveness* on task achievement, obtained by measuring the time taken for the agent to reach the goal area. High values of ineffectiveness are generally associated to wandering behavior or stationary dead ends encountered during navigation. The second index computes effectiveness and persistence, by measuring time percentage spent inside the goal area after the area is reached. Low values of the index are generally associated with excessive random behavior or fortuitous goal area achievements. The tests are performed on increasing environment complexity levels, and then growing time needed to success. UN network shows negative performances on all tests conditions, while SU network shows the best performances especially into low complexity environments, with a low number of obstacles, moving straight to the goal area in a few number of cycles, basically focusing on target. RE shows best performances particularly in high complexity environments, proving better explorations capabilities and abilities to overcome stationary configurations.

Figure 4 reports performance statistics indexes over increasing learning time of AD neural network from 0 to 400.000 learning cycles, increasing by a 50.000 interval. The progressive trend is evident and demonstrates the adaptive capabilities of the reinforcement learning protocol. Positive performance indexes show a clear increasing trend, while negative performance indexes show a decreasing trend. Performance charts show acceleration between 100.000 and 200.000 learning cycles, with inflection points in this interval, and a final stabilization after 250.000 learning iterations.

A.

IV. Conclusion

We presented simulations of the behavior of a mobile agent equipped with a neural net reflex-like control in avoiding ob-

stacles and achieving a given target position goal. At this stage of the project we use no high level onthologies or primitives for describing the environment, no environment model or map acquisition capabilities, and no planning abilities. We implemented the artificial neural net control with two different learning approaches: a standard supervised error back propagation training with examples, and an unsupervised reinforcement learning with environmental feedback. We constructed both a real robot agent and a virtual agent-environment simulation system, in order to perform fast and reliable test experiments. The virtual environment let us perform advanced integrated training and test sessions with progressive complexity levels and random configurations, leading to a high grade of generalization for the neural net control. We collected statistical data on several test experiments and compared the performance of the two learning approaches. The analysis of the control system critical aspects and capabilities, as observed in the simulations, favored fixing and improving data presentation in the training protocol.

REFERENCES

[1] Anvar A.M., Anvar A.P. (2011). *AUV Robots Real-time Control Navigation System Using Multi-layer Neural Networks Management*, 19th International Congress on Modelling and Simulation, Perth, Australia.

[2] Awad H.A., Al-Zorkany M.A. (2007). *Mobile Robot Navigation Using Local Model Networks*, World Academy of Science, Engineering and Technology.

[3] Bing-Qiang Huang, Guang-Yi Cao, Min Guo (2005). *Reinforcement Learning Neural Network to the Problem of Autonomous Mobile Robot Obstacle Avoidance*, Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou.

[4] Chen C., Li H.X., Dong D., (2008). *Hybrid Control for Robot Navigation - A Hierarchical Q-Learning Algorithm*, Robotics and Automation Magazine, IEEE, 15(2), 37-47.

[5] Floreano, D., Mondada, F. (1994). *Automatic creation of an autonomous agent: Genetic evolution of a neural network driven robot*, Proceedings of the third international conference on Simulation of adaptive behavior: From Animals to Animats 3 (No. LIS-CONF-1994-003, pp. 421-430), MIT Press.

[6] Floreano D., Mondada F. (1996). *Evolution of homing navigation in a real mobile robot*, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 26(3), 396-407.

[7] Janglova D. (2004). *Neural Networks in Mobile Robot Motion*, International Journal of Advanced Robotic System, Institute of Informatics SAS, vol. 1, no. 1, pp. 15-22.

[8] Glasius, R., Komoda, A., Gielen, S.C. (1995). *Neural network dynamics for path planning and obstacle avoidance*, Neural Networks, 8(1), 125-133.

[9] Medina-Santiago A., et. al. (2014). *Neural Control System in Obstacle Avoidance in Mobile Robots Using Ultrasonic Sensors*, Instituto Tecnolgico de Tuxtla Gutirrez, Chiapas, Mxico. pp. 104-110

[10] Michels J., Saxena, A., Ng, A. Y. (2005). *High speed obstacle avoidance using monocular vision and reinforcement learning*, Proceedings of the 22nd international conference on Machine learning (pp. 593-600), ACM.

[11] Milln J. (1995). *Reinforcement Learning of Goal-Directed Obstacle-Avoiding Reaction Strategies in an Autonomous Mobile Robot*, Robotics and Autonomous Systems, Volume 15, Issue 4. pp. 275299.

[12] Na, Y.K., Oh, S.Y., (2003). *Hybrid control for autonomous mobile robot navigation using neural network based behavior modules and environment classification*, Autonomous Robots, 15(2), 193-206.

[13] Pomerleau D.A. (1991). *Efficient Training of Artificial Neural Networks for Autonomous Navigation*, in Neural Computation3: 1. pp. 88-97.

[14] Rogers T.T. , McClelland J.L, (2014). *Parallel Distributed Processing at 25: Further Explorations in the Microstructure of Cognition*, Cognitive Science 38, 10241077.

[15] Rummery G.A., Niranjan M. (1994). *On-Line Q-Learning Using Connectionist Systems*, Cambridge University.

[16] Tsankova D.D. (2010). *Neural Networks Based Navigation and Control of a Mobile Robot in a Partially Known Environment*, Mobile Robots Navigation, Alejandra Barrera (Ed.), ISBN: 978-953-307-076-6, InTech.

[17] Ulrich I., Borenstein J., (2000). *VFH*: Local Obstacle Avoidance with Look-Ahead Verification*, International Conference on Robotics and Automation, San Francisco, CA, 28, 2000, pp. 2505-2511

[18] Yang G.S., Chen E.K., An C.W., (2004). *Mobile robot navigation using neural Q-learning*, Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on (Vol. 1, pp. 48-52), IEEE.

[19] Yang S.X., Luo C. (2004). *A neural network approach to complete coverage path planning*, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 34(1), 718-724.

[20] Yang S.X., Meng M. (2000). *An efficient neural network approach to dynamic robot motion planning*, Neural Networks, 13(2), 143-148.

[21] Floreano D., Mattiussi C. (2002). *Manuale sulle reti neurali*, Il Mulino, Bologna.

[22] 4tronix website, $http://4tronix.co.uk/$

[23] *HC-SR04 Ultrasonic Ranging Module*, Iteadstudio, $http://wiki.iteadstudio.com/Ultrasonic\_Ranging\_Module\_HC-SR04$

[24] *Neuroph Framework*, Neuroph website, $http://neuroph.sourceforge.net/$

[25] RaspberryPi website, $http://www.raspberrypi.org$

A game-based model for human-robots interaction*

Aniello Murano and **Loredana Sorrentino**
Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione
Università degli Studi di Napoli Federico II, Italy
{aniello.murano,loredana.sorrentino}@unina.it

I. Abstract

Game theory has exhibited to be a fruitful metaphor to reason about multi-player systems. Two kinds of games are mainly studied and adopted: turn-based and concurrent. They differ on the way the players are allowed to move. However, in real scenarios, there are very simple interplays among players whose modeling does not fit well in any of these settings.

In this paper we introduce a novel game-based framework to model and reason about the interaction between robots and humans. This framework combines all positive features of both turn-based and concurrent games. Over this game model we study the reachability problem.

To give an evidence of the usefulness of the introduced framework, we use it to model the interaction between a human and a team of two robots, in which the former tries to run away from the latter. We also provide an algorithm that decides in polynomial time whether at least one robot catches the human.

II. Introduction

In recent years, *game theory* has exhibited to be a fruitful metaphor in multi-agent system modeling and reasoning, where the evolution of the entire system emerges from the coordination of moves taken by all agents being involved [1], [20], [21], [22], [15], [23], [7], [8]. In the simplest setting, we consider finite-state games consisting of just two players (or agents), conventionally named *Player* 1 and *Player* 2. Depending on the possible interactions among the players, games can be either *turn-based* or *concurrent*. In the former case, the moves of the players are interleaved. In the latter case, instead, the players move simultaneously. In a turn-based game, the states of the game are partitioned into those belonging to Player 1 and those belonging to Player 2. When the game is at a state of Player i, only Player i determines the next state. In a concurrent game, instead, the two players choose, simultaneously and independently, their moves and the next state of the game depends on the combination of such moves.

A game consists of two main objects, the *arena* and the *winning condition*. The arena is used to describe the players, the game states (configurations), and the possible evolution of the game in accordance to the moves the players can take. The winning condition is used to express the objective the players aim to achieve.

* This work is partially supported by the FP7 EU project 600958-SHERPA.

Solving a game corresponds to check whether a designed player has a winning strategy in the game, i.e. a sequence of moves that let him satisfy the winning condition no matter how the other players behave. In the literature, we distinguish between the case the condition is given as an "external entity", for example via a formula of a logic [1], [14], or internally as a condition over the states of the arena. While external conditions offer some modularity and allow to formalize very intricate targets, they require solutions with a very high complexity [12]. Internal conditions instead offer a good balance between expressiveness and complexity and this is the setting we consider in this paper.

A very simple and largely used (internal) winning condition consists of labeling some states of the arena as "*good*" and then consider as target the *reachability* of at least one of them. Properly speaking, the resulting setting is called a *reachability game*. These games have been exploited in both the (two-player) turn-based and (multi-player) concurrent settings and fruitfully applied in several interesting real scenarios. However, there are specific situations that cannot be casted in any of these settings. In particular, this happens when we want to model the interplay among agents with a different essence. This is the case, for example in human-robot interaction. To give an evidence of this necessity, we discuss along the paper a scenario involving the interaction between a human and two robots. The shape of the arena is a maze and the three players are initially placed in three different positions. The goal of the human is to run away from the robots, while the robots have the opposite goal. Therefore, looking at the game from the robots side, the good states are those in which at least one of the robots and the human sit in the same place in the same moment. We assume that whenever the human decides to move, none of the robots can interfere and vice-versa. In other words the human uses the game on its side as turn-based while the robots use it as being concurrent. We introduce a novel model framework that is able to represent efficiently such a scenario and provide a solution algorithm that can decide in polynomial time whether the robots have a winning strategy, i.e., they have a sequence of moves that, no matter how the human behaves, they reach a desired state.

Related work. Robotic technology is quickly advancing and this rapid progress inevitably is having a huge effect over the people. The interaction between human and robots is a complex issue that challenges both engineering and cognitive science. In several settings, such an interaction has been modeled in terms of a suitable interplay between all

actors involved (see [5] for a survey). Several models in this context take inspiration from the field of open-system formal verifications [9], [19], [11], [16]. A system is considered *open* if it has an ongoing interaction with an external unpredictable environment and the system behavior fully depends on this interaction. To verify an open system means to check that the system is correct no matter how the environment behaves. Several models based on two-player games (system vs. environment) have been proposed in order to model such an interaction as well as suitable algorithms to check whether the system is correct (i.e. wins the game) [3], [24]. In this context, multi-agent games have been also proposed in order to model and reason about multiple players able to play in a cooperative or adversarial manner [1], [14].

The game setting we consider in this paper has several connections with planning problems as well [2], [4], [6]. Indeed, planning can be rephrased as the problem of synthesizing a strategy (the *plan*) for an agent to achieve a determined task within an environment. Often the environment is hostile and consists of an aggregation of several entities working together. By casting such a scenario in our model setting one can see the environment as the team of cooperative agents working against the one aiming for the planning.

Since the environment can be seen as an adversarial player the correlation the our setting follows

III. Case Study

In this section we introduce a simple but effective human-robot interaction scenario that will be used along the paper as an application for the game-model framework we introduce. The scenario is described in the following and depicted in Figure 1.

The interaction takes place in a maze and involves tree players: a human H and two robots R_1 and R_2. The goal of the human is to escape from both the robots. The robots work in team and aim just the opposite. For simplicity we assume the maze divided in square rooms and we start by considering that the players sit all in different rooms. All players from every room can access to an adjacent one unless there is a wall (drawn with a bold line in the figure). We assume to have an interleaving of moves between the human and the team of robots. Hence, the robots move simultaneously. We assume that all players can move in the four directions *up*, *down*, *left*, and *right*, unless the shape of the maze forbids it.

Let us now discuss how such an human-robot interaction can be rephrased in terms of a game. We make this reasoning more formal in the next section. Starting from an initial position in the maze, all players by taking the allowed moves can change their position. Each possible placing of the players can be seen as a *configuration* of the game and the starting one is usually denoted the *initial configuration*. Clearly, we can move in one step from one configuration to another only if we have moves that allow it. In particular, as seen from the human, moves are interleaved in a *turn-based* fashion, while they are taken in a *concurrent* way as seen by the robots. All legal moves can be collected by an opportune data structure or a table. Following the target of the robots, we have that the human loses if along

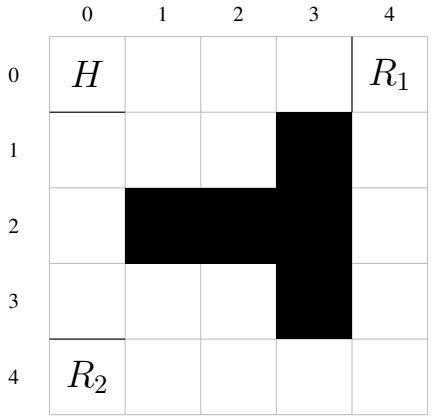


Fig. 1. Maze Game.

an interplay, the game reaches a configuration in which both he and one of the robots sit in the same room. Accordingly, we label all such configurations as good ones (as seen by the robots). Thus, deciding whether the team of robots can defeat the human corresponds to solve a *reachability game*.

It is important to note that the scenario we have discussed is neither (just) turn-based, as the robots move simultaneously, nor concurrent, as the human moves independently from the robots. Moreover, the discussed game involves three players and it is not trivially reducible to a two-player one because of the particular target: at least one of the robots has to catch the human.

To model this scenario, we introduce in the next section a novel game model in which all players, except a designed one, work in team and move simultaneously. The designed one instead will move alternately and independently from the other players.

IV. The conceived model

In this section we introduce a novel multi-player game-based framework that is suitable to represent, under a unique structure, both the players moving turn-based and those acting concurrently. This framework, which we call *hybrid*, opportunely combines and extends the main features behind *concurrent game structures* [1] and two-player turn-based games (see [18] for an introduction).

Definition 1 (Hybrid Game models): A *hybrid reachability game structure* is a tuple $G =< Ag, Ac, St, s_0, tr, St_f >$, where Ag is a finite non-empty set of *agents*, partitioned into two teams Ag_0 and Ag_1. Ac and St are enumerable non-empty sets of *actions* and *states*, respectively, and $s_0 \in St$ is a designated *initial state*. The set of states is partitioned in St_0 and St_1 as those states belonging to the teams Ag_0 or Ag_1, respectively. For $i \in \{0,1\}$, let $Dc_i = Ag_i \rightharpoonup Ac$ to be the set of *decisions* of team i, i.e., partial functions describing the choices of an action by all agents in Ag_i. Then, $tr : Dc_i \times St_i \rightarrow St_{1-i}$ denotes the *transition function* mapping every pair of decisions $\delta \in Dc_i$ and state $s \in St_i$ for the team i to a successive state over the deterministic graph belonging the the adversary team. Finally, we define $St_f \subseteq St$ the subset of *terminal* (or *accepting*) states.

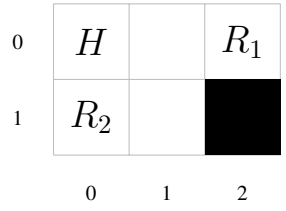


Fig. 2. A Reduced Maze Game.

We now show how the case study we have presented in the previous section can be easily and formally described by means of a hybrid game model G.

For a sake of clarity, instead of considering the scenario depicted in Figure 1, we consider a reduced one as reported in Figure 2. Also, we allow the robots to move only horizontally (left and right), while the human is still able to move in all directions. While this new scenario may look too naive, it will avoid a bunch of tedious calculations in the sequel. We model the human-robots interaction over this maze by setting Ag_0 as the team consisting of the unique player *human* and Ag_1 as the team of *robots* R_1 and R_2 . We consider the maze as a grid-board made by $K \times J$ positions, with $K = \{0, 1\}$ and $J = \{0, 1, 2\}$. In each position, zero, one, or more than one player can sit. States are just a set of positions of the three players plus a flag we use to recall which team is playing in that state. Formally, we have as set of states $St = ((K \times J)^3) \times \{0, 1\}$. St_i contains those states having the flag equal to i. The initial state is just the initial position of the players.

Accordingly to the picture depicted in Figure 2, assuming R_1 and R_2 are the next to move, the state is $((0, 0), (1, 0), (0, 2), 1)$. We assume in our example that this is the initial state. The possible actions for the robots are set to r for *right* and l for *left*. For the possible actions of the human we set u for *up*, d for *down*, l for *left*, and r for *right*. Decisions are defined accordingly and must respect the limitations imposed by the shape of the maze. As far as the set of accepting states concerns, recall that the target of the robots is to reach a configuration where at least one of them catches the human, being both siting in the same position. This means that St_f must contain all those states in which the first pair of coordinates (corresponding to the position of the human) is equal to the second or third pair. Formally, $St_f = \{((a, b), (c, d), (e, f), i) \mid (a = c \text{ and } b = d) \text{ OR } (a = e \text{ and } b = f)\}$. Finally, it remains to define the transition relation. For the sake of exposition, we only report the part corresponding to the team Ag_0. Note that this is coherent with the shape of the maze.

$$tr(\delta, ((i, j)(k, l)(m, n)), 0) =$$

$$\begin{cases} (((i-1, j)(k, l)(m, n)), 1), & \text{if } \delta = u \text{ and } i > 0; \\ (((i, j-1)(k, l)(m, n)), 1), & \text{if } \delta = l \text{ and } j > 0; \\ (((i, j+1)(k, l)(m, n)), 1), & \text{if } \delta = r \text{ and } j < 2; \\ (((i+1, j)(k, l)(m, n)), 1), & \text{if } \delta = d \text{ and } i < 1. \end{cases}$$

To better clarify the meaning of the above formalization, let us discuss some examples over the maze. From the initial state, which belongs to the team Ag_1, by using the decision lr, the game moves to the state $(((0, 0)(0, 1)(1, 1)), 0)$. In accordance with the flag, this is now a state belonging to the team Ag_0 and thus this team (the human) takes the turn to move. From this state, the human agent has two available moves, that are d and r. In the first case the game moves in the state $(((1, 0)(0, 1)(1, 1)), 1)$ and in the second case it moves in the state $(((0, 0)(0, 1)(1, 1)), 1)$, both belonging to the players in the coalition Ag_1. And so on. In Figure 3, we report the computations of the game. It is easy to observe that the accepting states are $((1, 0), (0, 2)(1, 0)), 0)$, $((1, 0), (0, 0)(1, 0)), 0)$ and $((0, 1), (0, 1)(1, 1)), 1)$ since one of the two robots and the human are both in the room.

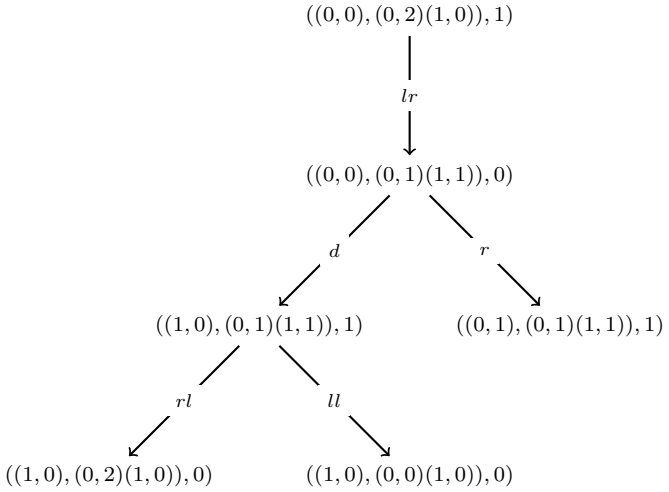


Fig. 3. A game model G for the simplify maze in Figure 2.

V. The proposed solution

Under the conceived model, we can handle all possible targets that can be represented in terms of *reachability*, i.e., the players in the coalition Ag_1 set some configurations of the game as "good" and aim to reach them. These configurations are those represented by states St_f in the model. The coalition Ag_1 *wins* the game if its players have a *winning strategy*, i.e., they can force the game, by means of a sequence of moves, to reach a state in St_f, *no matter* how the players in the team Ag_1 behave. *Deciding a game* means deciding whether the coalition Ag_1 wins the game.

In this section we provide an algorithm to decide a game under the hybrid framework we have introduced. This algorithm aims to find the set of states of the model from which the players in the coalition Ag_1 win the game, that is the set of states $reach_1(St_f)$. As the complement of this set contains the states from which players in the coalition Ag_0 win the game, as a corollary we obtain that our game model is zero-sum (i.e. from each node only one team can win the game).

The algorithm proceeds as follows. We start from the set St_f containing all winning states for players in Ag_1. Then, in a backward manner, we recursively build the set

$reach_1^i(St_f)$ containing all states $s \in St$ such that players in Ag_1 can force a visit from s to a state in the set St_f in less than i steps. Formally, we have that $reach_1^i(St_f) = \{s \in St \,|\, Ag_1$ can force a visit from s to St_f in less than i moves$\}$.

Formally, we have the following.

$$reach_1^0(St_f) = St_f;$$

$$reach_1^{i+1}(St_f) = reach_1^i(St_f) \cup$$
$$\{s \in St_1 | \exists s' \in St : tr(Dc_1, s) = s' \wedge s' \in reach_1^i(St_f)\} \cup$$
$$\{s \in St_0 | \forall s' \in St : tr(Dc_0, s) = s' \wedge s' \in reach_1^i(St_f)\}.$$

In words, from the set $reach_1^i(St_f)$ we select all states that have incident edges in this set. From each of these states, say s, if it belongs to the coalition Ag_1, then, this state is immediately added to $reach_1^{i+1}(St_f)$ (as we *may* move from s to $reach_1^i(St_f)$ and thus reach St_f). Conversely, if s is a state belonging to the coalition Ag_0, then it is added to $reach_1^{i+1}(St_f)$ only if all outgoing edges from s fall in $reach_1^i(St_f)$ (i.e., from s, players in Ag_0 are forced to move to $reach_1^i(St_f)$).

Finally we have that

$$reach_1(St_f) = reach_1^{|St|}(St_f).$$

As the calculation of $reach_1(St_f)$ requires a number of iterations linear in number of states St, we have that the whole algorithm requires at most quadratic time in the size of the model, as reported in the following theorem.

Theorem 1: Given a hybrid reachability game, it can be decided in quadratic time.

By a matter of calculation, one can see that by applying the algorithm above over our reduced case study, the coalition Ag_1 wins the game from every state. In fact, for each state of the model, there exists always a winning strategy for the players in the team Ag_1.

VI. Discussion and Future Works

In the last years, human-robots interaction is receiving large attention in several research fields. An important aspect in this study is to come up with appropriate models to design and reasoning about how such interactions can take place and how they affect the future behavior of the involved actors. In this setting, *game theory* is a powerful framework that is able to formalize the interplay between the human and the robots in a very natural way.

In this paper, we have introduced a game model framework that allows to represent and reasoning about scenarios in which the interaction between the humans and the robots results in a hybrid two-team game: the game between the two teams is turn-based while all players in each team play concurrently among them. We have observed that classic turn-based and concurrent games are not powerful enough to model such a setting. Over the conceived model, we study the reachability problem and show that it is solvable in quadratic time. Therefore, with no extra cost with respect to classic turn-based and concurrent games. We give an evidence of the usefulness of the introduced framework by means of a case study.

This work opens to several future directions. First, it would be interesting to reasoning about quantitative aspects about the human-robots interaction. For example, to determine what is the best strategy to perform.

Another direction would be to consider other possible winning conditions. In particular, one could import some winning conditions studied in formal verification such as the *Büchi* and the *parity* ones (see [17], [10] for an introduction) or external winning conditions given in terms of formulas of a suitable logic. Some scenarios along these lines have been already investigated in the case of turn-based and concurrent game settings [9], [19] and showed to have some useful applications [13], [14]. We plan, as future work, to investigate them in the settings of multi-robots and human-robots interactions.

References

[1] R. Alur, T. Henzinger, and O. Kupferman. Alternating-Time Temporal Logic. *Journal of the ACM*, 49(5):672–713, 2002.

[2] D. Calvanese and G. De Giacomo and M.Y Vardi. Reasoning about actions and planning in LTL action theories. *KR*, 2:593–602, 2002.

[3] L. de Alfaro, T. Henzinger, and O. Kupferman. Concurrent reachability games. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 564–575. IEEE, 1998.

[4] G. De Giacomo and M. Y. Vardi. Automata-theoretic approach to planning for temporally extended goals. pages 226–238, 2000.

[5] M. A. Goodrich and A. C. Schultz. Human-robot interaction: a survey. *Foundations and trends in human-computer interaction*, 1(3):203–275, 2007.

[6] H. Geffner and B. Bonet. A concise introduction to models and methods for automated planning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(1):1–141, 2013.

[7] J. Gutierrez and M. Wooldridge. Equilibria of concurrent games on event structures. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, page 46. ACM, 2014.

[8] J. Van Benthem. *Logic games: From tools to models of interaction*. Institute for Logic, Language and Computation (ILLC), University of Amsterdam, 2007.

[9] W. Jamroga and A. Murano. On Module Checking and Strategies. In *Autonomous Agents and MultiAgent Systems'14*, pages 701–708. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[10] O. Kupferman, M. Vardi, and P. Wolper. An Automata Theoretic Approach to Branching-Time Model Checking. *Journal of the ACM*, 47(2):312–360, 2000.

[11] O. Kupferman, M. Vardi, and P. Wolper. Module Checking. *Information and Computation*, 164(2):322–344, 2001.

[12] F. Mogavero, A. Murano, G. Perelli, and M. Vardi. Reasoning About Strategies: On the Model-Checking Problem. *Transactions On Computational Logic*, 15(4):34:1–42, 2014.

[13] F. Mogavero, A. Murano, and L. Sorrentino. On Promptness in Parity Games. In *Logic for Programming Artificial Intelligence and Reasoning'13*, LNCS 8312, pages 601–618. Springer, 2013.

[14] F. Mogavero, A. Murano, and M. Vardi. Reasoning About Strategies. In *Foundations of Software Technology and Theoretical Computer Science'10*, LIPIcs 8, pages 133–144. Leibniz-Zentrum fuer Informatik, 2010.

[15] S. Parsons and M. Wooldridge. Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 5(3):243–254, 2002.

[16] P. Schobbens. Alternating-Time Logic with Imperfect Recall. 85(2):82–93, 2004.

[17] W. Thomas. Automata on infinite objects. *Handbook of theoretical computer science*, 2, 1990.

[18] W. Thomas. Monadic Logic and Automata: Recent Developments. 1998.

[19] W. Jamroga and A. Murano. Module checking of strategic ability. pages 227–235, 2015.

[20] M. Wooldridge. Intelligent Agents. In G. Weiss, editor, *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*. MIT Press: Cambridge, Mass, 1999.

[21] M. Wooldridge. Computationally Grounded Theories of Agency. In *International Conference on MultiAgent Systems'00*, pages 13–22. IEEE Computer Society, 2000.

[22] M. Wooldridge. *An Introduction to Multi Agent Systems*. John Wiley & Sons, 2002.

[23] Y. Jiang and J. Hu and D. Lin. Decision making of networked multiagent systems for interaction structures. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(6):1107–1121, 2011.

[24] W. Zielonka. Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.

INDEX OF AUTHORS