N-Mates Evaluation: a New Method to Improve the Performance of Genetic Algorithms in Heterogeneous Multi-Agent Systems.

Paolo Pagliuca[1,*,†], Alessandra Vitanza[1,†]

[1]Institute of Cognitive Sciences and Technologies, National Research Council (CNR-ISTC), Rome, Italy

Abstract

Multi-Agent Systems (MASs) are a widely used paradigm for modeling agents that interact with each other to solve problems. Genetic algorithms represent methods mimicking natural evolution and have been successfully applied in various domains, including MASs. While evolving controllers for homogeneous agents can be considered a relatively trivial task, evolving a collective ability in a group of heterogeneous agents strongly depends on the individual's characteristics. In a genetic algorithm, the selection of the individuals forming the MAS is random and the evaluation of the group performance is affected by both the agent's ability and the environmental complexity. Consequently, the emergent dynamics of the system can be highly unpredictable, and the success or failure of the MAS may be inaccurately evaluated. To mitigate the effect of chance, we proposed a novel technique - called *n-mates evaluation* - which allows for a better estimation of each individual's effectiveness and its contribution to the final performance of the MAS. Results collected from three different cooperative benchmark tasks indicate that the proposed method is effective and outperforms a traditional genetic algorithm.

Keywords

Genetic algorithms, multi-agent simulation, emergent behaviors, optimization strategies

1. Introduction

Multi-agent systems (MASs) have emerged as a significant research area that has found applications in various fields, including industrial applications, robotics, economics, social sciences, and more. MASs offer flexible and decentralized approaches to problem-solving, allowing adaptation to changes in dynamic and complex environments [1, 2]. Involving interactions of multiple autonomous entities to achieve collective goals, agents in MASs are capable of perceiving their environment and making decisions or actions to achieve their individual objectives while also considering the impact on the overall system [3]. The coordination and communication among them can lead to emergent behaviors that often cannot be achieved by individual agents or with centralized control. The study of multi-agent systems involves understanding agent behavior, interaction protocols, negotiation strategies, and mechanisms for achieving cooperation or competition among agents [1].

WOA 2023: 24th Workshop From Objects to Agents, November 6–8, Rome, Italy

[*]Corresponding author.

[†]The authors contributed equally.

paolo.pagliuca@istc.cnr.it (P. Pagliuca); alessandra.vitanza@istc.cnr.it (A. Vitanza)

0000-0002-3780-3347 (P. Pagliuca); 0000-0002-7760-8167 (A. Vitanza)

Genetic algorithms (GAs) [4] are methods modeling the evolution of natural organisms that are widely used to solve optimization problems. Examples of successful application of GAs to MAS tasks can be found in [5–12]. While genetic algorithms (GAs) have emerged as valuable tools for solving complex optimization and search problems in various fields [13–15], in the context of multi-agent systems, they offer a powerful approach to optimize agent behaviors, interactions, and strategies [16, 17]. The application of genetic algorithms within multi-agent simulation scenarios offers several benefits and challenges. Firstly, genetic algorithms stand out in exploring a diverse space of solutions. Especially in MASs, this ability allows the exploration of alternative strategies and possible emergent behaviors, enabling efficient problem resolution.

In particular, multi-agent simulations aim to study emergent behavior arising from interactions between agents. Thus, GAs provide a means to uncover patterns of emergent behavior and study their underlying mechanisms. Furthermore, since MASs often require agents to adapt to dynamic environments and diverse situations, GAs furnish the opportunity to evolve agents that learn optimal strategies over time, improving their performance in response to changing conditions, taking into account multiple objectives and constraints. This is particularly beneficial in complex scenarios where solutions are based on various factors, leading to improved cooperation, coordination, and conflict resolution within multi-agent systems. Examples of genetic algorithms applied to MASs can be found in [18–23].

As pointed out in [24, 25], the use of heterogeneous agents in MASs remains an open question. Moreover, the evolution of collective behaviors in groups of heterogeneous agents is affected by two main factors: (i) the ability of each individual in the MAS and (ii) the variability of the environmental conditions. This implies that groups formed by skilled individuals acting in a simple environment are more likely to be selected compared to groups containing individuals with less capable partners in complex scenarios. In other words, randomness and chance could alter the evaluation of the individual's capability, ultimately hindering the discovery of truly effective individuals [19, 26, 27]. To deal with such issues, we introduce a new method - called *n-mates evaluation* - allowing us to better estimate the individual's capability within a MAS. Specifically, we designed three evolutionary scenarios involving a simple two-agent MAS, in which individuals have to collaborate in order to solve the problems. The considered problems represent benchmark tasks in the areas of swarm and evolutionary robotics. Differently from traditional methods evaluating random pairs, the *n-mates evaluation* technique evaluates each individual with n different partners sampled from the population and computes its performance as the average of the n evaluations. In this work, we empirically set $n = 5$ and combined our method with a standard genetic algorithm. We then compared its performance with the genetic algorithm, which serves as our baseline. Based on our experiments, the results indicate that the former approach outperforms the latter algorithm. Furthermore, evolved individuals are more adaptive to both the partner's capability and the different environmental conditions.

2. Method Description

Genetic algorithms (GAs) [4, 14, 28] are popular techniques developed in the 70s with the original aim of understanding adaptation in living organisms [4]. These methods displayed the potential to solve many optimization problems and were employed in many different domains (for some examples see [29–37]). Genetic algorithms are usually employed to evolve neural network controllers for autonomous agents [10, 34, 38–40].

For the experiments reported in this work, we employed the Generational Genetic Algorithm (GGA) [6, 41], a slightly modified version of the genetic algorithm developed by Holland in which no crossover is possible between the population members. Specifically, the GGA evolves a population of genotypes, each one being a sequence of integer values that encode the connection weights of a neural network controller (refer to Section 3.4). The transformation to convert genes into connection weights is shown in Eq. 1:

$$w = w_{range} - \frac{g}{MAX_G} * w_{range} * 2.0 \tag{1}$$

where g is the generic gene value (integer), $MAX_G = 255$ is the maximum gene value and $w_{range} = 5.0$ represents the weight range. We used the same settings as in [6].

Fig. 1 shows the pseudo-code for the GGA.

```
1: initialize NGenerations, PopSize, NReproducing, NOffspring, Pop(0)
2: for gen <- 0 to NGenerations do
3:     for p <- 0 to PopSize do
4:         Fitness[p] <- evaluate (p)
5:     done
6:     selectBestReproducing(Fitness, NReproducing)
7:     Pop(gen + 1) = generateOffspring(NReproducing, NOffspring)
8: done
```

Figure 1: Pseudo-code of the Generational Genetic Algorithm (GGA). The process consists of several sequential steps. Firstly, an initial population $Pop(0)$ is formed, and the *individuals* (also known as *chromosomes* or *genotypes*) are initialized with random values. Each individual represents a potential solution for the specific problem. Subsequently, all individuals are evaluated, and scores are assigned to measure their performance in addressing the problem (lines 3-5). The selection process then identifies the most capable individuals, allowing them to engage in reproduction (line 6). Using the mutation and/or recombination operators, a new population $Pop(gen+1)$ is generated (line 7). This cyclic process continues iteratively until a predetermined criterion is achieved, such as reaching a fixed number of generations or achieving an optimal problem solution (lines 2-8).

The application of GGA to MASs formed by heterogeneous agents is tightly dependent on the ability of each member of the group. This implies that the selection of the best individuals for reproduction might be affected by chance. For example, an effective agent evaluated with an unable mate can be discarded since the performance of the pair will be poor. Moreover, selected agents are not retained in the population and the members of the MAS are randomly chosen at each generation. To cope with this issue, we introduce a novel technique that we called *n-mates evaluation*: it consists of evaluating each individual with n mates randomly extracted from the population. The performance of the individual is the average over the n evaluations. In this way, the fitness measure represents a better estimation of the ability of each agent to deal with the evolutionary task. Furthermore, the fitness value provides an indication of how adaptive an agent is within the MAS. Because the extraction of mates is purely random, the *n-mates evaluation* technique forces each agent to adjust its behavior according to the partner's capabilities. In other words, the *n-mates evaluation* method allows for the generalization of manifold behaviors, addressing the issue related to agent's heterogeneity. This is why we empirically estimate the value of parameter *n* by assigning it as 5. The pseudo-code of the *n-mates evaluation* method is shown in Fig. 2.

```
1 input: p, Pop(gen)
2 extractMates(NMates, Pop(gen))
3 Fitness[p] <- 0
4 for pp <-0 to NMates do
5         Fitness[p] <- Fitness[p] + evaluate(p, pp)
6 done
7 Fitness[p] <- Fitness[p] / NMates
```

Figure 2: Pseudo-code of the *n-mates evaluation* procedure. The method takes as inputs an individual p and the population $Pop(gen)$ at generation gen (line 1). A set of n mates randomly drawn from the population (line 2). The individual receives a score indicating the performance obtained with a specific mate (lines 4-6). The actual individual's performance is obtained by averaging the fitness over the number of mates (line 7).

We used the *n-mates evaluation* technique to improve the performance of the GGA. The resulting algorithm is described in Fig. 3. As it can be observed, the *n-mates evaluation* method only affects the agent's evaluation (i.e., the computation of the fitness value).

```
1 initialize NGenerations, PopSize, NReproducing, Noffspring, Pop(0), NMates
2 for gen <- 0 to NGenerations do
3         for p <- 0 to PopSize do
4                 extractMates(NMates, Pop(gen))
5                 Fitness[p] <- 0
6                 for pp <- 0 to NMates do
7                         Fitness[p] <- Fitness[p] + evaluate(p, pp)
8                 done
9                 Fitness[p] <- Fitness[p] / NMates
10        done
11        selectBestReproducing(Fitness, NReproducing)
12        Pop(gen + 1) <- generateOffspring(NReproducing, NOffspring)
13 done
```

Figure 3: Pseudo-code of the *n-mates evaluation* procedure inserted in the GGA (blue lines 4-9).

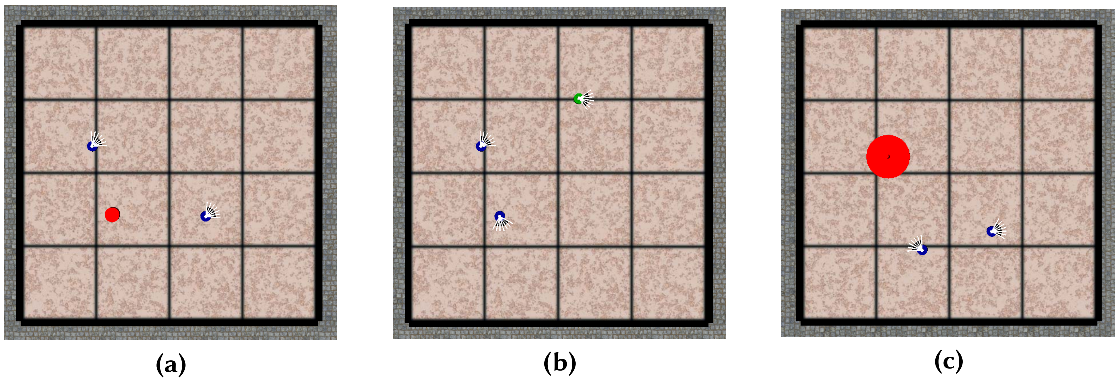
Figure 4: Evolutionary tasks. **(a)** Foraging: food items are shown in red. **(b)** Escaping from predator: the predator, colored in green, moves following the closest detected prey. **(c)** Aggregation: the arena contains an aggregation area colored in red.

3. Problem Formulation

In this section, we describe the evolutionary tasks we used. We consider three benchmark problems largely employed in the fields of swarm and evolutionary robotics [5, 9, 19, 21, 22, 42–46]. Moreover, the tasks are of increasing complexity, as we will explain in the next subsections. A screenshot of the environmental setups is provided in Fig. 4. For all the experiments we used the parameters reported in Table 1. All the experiments have been run by using the FARSA simulator [47, 48], an open software tool widely used in evolutionary robotics [49–51] and MAS [6, 52].

Table 1
Experimental parameter settings.

Experiment Parameter	Value	G.A. Parameter	Value
# of replications	20	population size	100
# of evaluation steps	5×10^8	# of reproducing individuals	10
# of episodes	5	# of offspring per individual	10
# of steps per episode	1000	gene range	$[0, 255]$
arena size	$2m \times 2m$	mutation rate	1%
camera field-of-view (FOV)	90°	crossover	off
camera sectors	6	# of mates	$\{1, 5\}$

3.1. Foraging

The foraging task is considered a benchmark problem in robotics and MASs [6, 19, 20, 43, 45, 46]. In the classic formulation, a group of agents has to coordinate in order to forage as many food items as possible [19, 46]. Concerning our evolutionary problem, two E-puck robots [53] are randomly placed in the environment (Fig. 4(a)). A food item is set in a random position and cannot be eaten by a single agent. When the agents succeed in eating the item, it suddenly reappears in a new random position. The goal for the robots is to gather the highest possible number of food elements. The fitness function is provided in Eq. 2:

$$F_{foraging} = \frac{1}{N_{episodes}} \sum_{i=1}^{N_{episodes}} N_{eaten\_foods} \tag{2}$$

where $N_{episodes}$ is the number of episodes and N_{eaten_foods} represents the number of food items eaten by the two agents. The lower bound for the fitness function is 0.0 (i.e., an agent that does not eat any food items), while there is no upper bound since the fitness value is tightly dependent on the agent's capabilities.

Although the task is quite trivial, it does not necessarily require the collaboration of the two agents. In fact, the problem can be solved simply if the two agents reach a food element in distinct time steps. There is no pressure on explicit coordination of the robots. Nonetheless, an individual capable of reaching quickly the food item might be penalized if evaluated with an unable partner.

3.2. Escaping from predator

In this scenario, we designed a variant of the "Pursuit and Evasion" problems [42, 44] in which three E-puck robots are randomly placed in the environment, with one acting as a predator (Fig. 4(b)). All the agents have the same capabilities. The goal for the evolving agents (acting as preys, blue in Fig. 4(b)) is to avoid being captured by the predator (green in Fig. 4(b)). This can be achieved by staying as close as possible. More specifically, the minimum distance that guarantees the agents cannot be captured by the predator corresponds to the diameter of a robot. This design choice was made considering that this value prevents the predator from standing between the agents. The fitness function for the problem is reported in Eq. 3:

$$F_{predator} = \frac{1}{N_{episodes}} \sum_{i=1}^{N_{episodes}} (1.0 - \frac{robot\_dist}{max\_dist}) \tag{3}$$

where $N_{episodes}$ is the number of episodes, *robot_dist* is the distance of the agents at the end of the episode and *max_dist* is the maximum distance of the robots in the environment. Specifically, *max_dist* corresponds to the diagonal of the arena. The fitness value is bounded in the range $[0.0, 1.0]$.

In order to solve this task, the agents may develop two potential capabilities:

1. they can coordinate with each other so as to move close;
2. they may escape from the predator in order to avoid being captured.

It is worth noting that the predator moves according to a predefined routine computing its direction of motion based on the closest detected prey. A possible solution for the agent under evaluation is to move rapidly and reach the partner before the arrival of the predator. This implies that a very fast individual can solve the problem even if the mate is an unable agent that stays still or turns on the spot.

3.3. Aggregation

Aggregation represents a very simple example of self-organized behavior in various species of animals [54] and typically comes from a shared decision among the members of the group [55]. Several works demonstrated how an aggregation behavior can emerge through the interaction of multiple agents in a MAS [5, 9, 21, 22, 56].

Here, we consider two E-puck robots placed in the arena. A red circular area of $30cm$ diameter is randomly located in the environment (Fig. 4(c)). There is a cylinder object (with a diameter of $2.5cm$) placed at the center of the area indicating its presence. The goal for the agents is to reach the aggregation area and spend as much time as possible within it. The fitness function is given by Eq. 4:

$$F_{aggregation} = \frac{1}{N_{episodes}} \sum_{i=1}^{N_{episodes}} \frac{N_{steps\_on\_aggregation\_area}}{N_{steps}} \tag{4}$$

where $N_{episodes}$ is the number of episodes, N_{steps} indicates the number of steps of each episode and $N_{steps\_on\_aggregation\_area}$ represents the number of steps the two agents spent together over the aggregation area. Similarly to the "escaping from predator" task, the fitness value is bounded in the range $[0.0, 1.0]$.

Differently from the other evolutionary problems, in this case, the robots are equipped with an additional ground sensor allowing the agents to recognize the presence of the area when they pass over it.

In order to solve the evolutionary problem, the agents must develop the following capabilities:

1. explore the environment so as to detect the aggregation area;
2. stay on the area as much time as possible.

Similar to the foraging problem, the agents are not required to explicitly coordinate their arrival at the aggregation area, nor do they need to detect the other mate. However, only when both agents are over the area the fitness increases. This implies that a robot capable of detecting and remaining within the area will not receive a fitness score if its partner is an incapable agent that remains stationary or rotates in place. Consequently, this task is considerably more challenging than the other two problems since the agent's evaluation is strongly dependent on the mate's capabilities.

3.4. Neural Network

The network architecture employed for the experiments described above is depicted in Fig. 5. We adopted the controller used in [6] with some adjustments. In more detail, we used a feed-forward neural network with one internal layer of 8 neurons and 2 output neurons encoding the speeds of the robot wheels. Concerning the input layer, the network has 18 inputs provided by the linear camera (6 inputs for each of the 3 basic colors) in the case of the "foraging" and the "escaping from predator" problems. With respect to the "aggregation" task, there are 3 additional inputs provided by the ground sensor (Fig. 5, dashed circles). This comprehensive network architecture forms the basis of agents' controllers in our evolutionary experimentation.

The connection weights are derived from the corresponding genotypes and initialized to values in the range [-5.0, 5.0], as described in Section 2.

4. Results

In this section, we provide the outcomes of our analyses.

Figures 6-8 show the performance of the two algorithms we used. As it can be seen, the *n-mates evaluation* technique allows us to obtain better performances on average than the GGA algorithm at the end of the evolutionary process in all the considered scenarios. Specifically, results are statistically significant for both the "foraging" and the "escaping from predator" problems (Mann-Whitney U test, $p < 0.05$), while there is no difference concerning the "aggregation" task (Mann-Whitney U test, $p > 0.05$). This indicates that our method is able to discover solutions generalizing with respect to the capabilities of the partner. To further support this claim, we run a post-evaluation phase in which evolved individuals are tested with 25 different partners randomly drawn from the final population. Differently from evolution, the test phase is deterministic, i.e. individuals experience the same initial conditions at the beginning of each episode. Overall, we evaluated 2000 individuals. The results of the post-evaluation (see Fig. 9) demonstrate how the *n-mates evaluation* technique evolved better generalizing individuals than

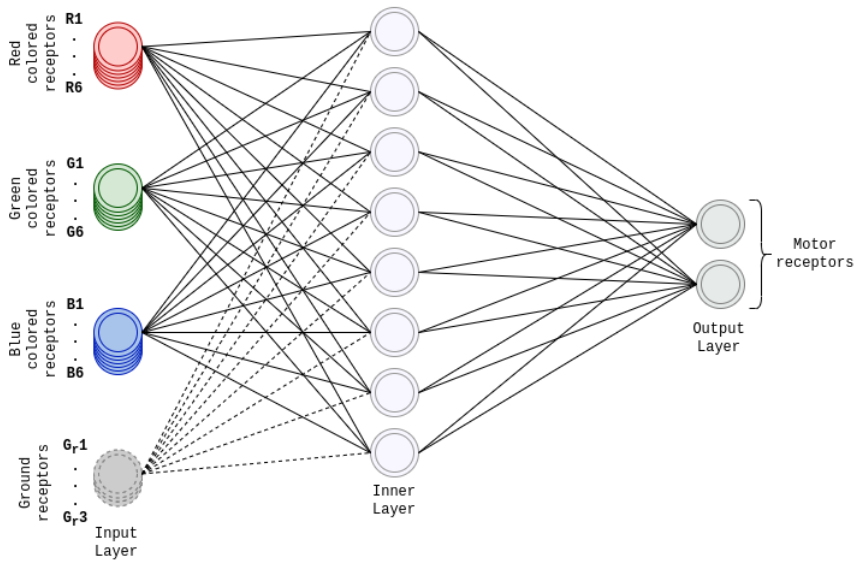

Figure 5: The neural network controller used in the experiments. Within the Input layer, there are specialized receptors for color perception, represented by neurons $R1, ..., R6$ (Red colored receptors), $G1, ..., G6$ (green colored receptors), and $B1, ..., B6$ (blue colored receptors). This results in a cumulative count of 18 neurons. This configuration is applied to both the "foraging" and "escaping from predator" tasks. Additionally, in the "aggregation" task, the input layer contains three additional inputs ($G_r1, ..., G_r3$), related to the ground sensors, bringing the total to 21 neurons. The internal layer, composed of 8 neurons, exerts influence over motor neurons, determining the speeds of the motors.

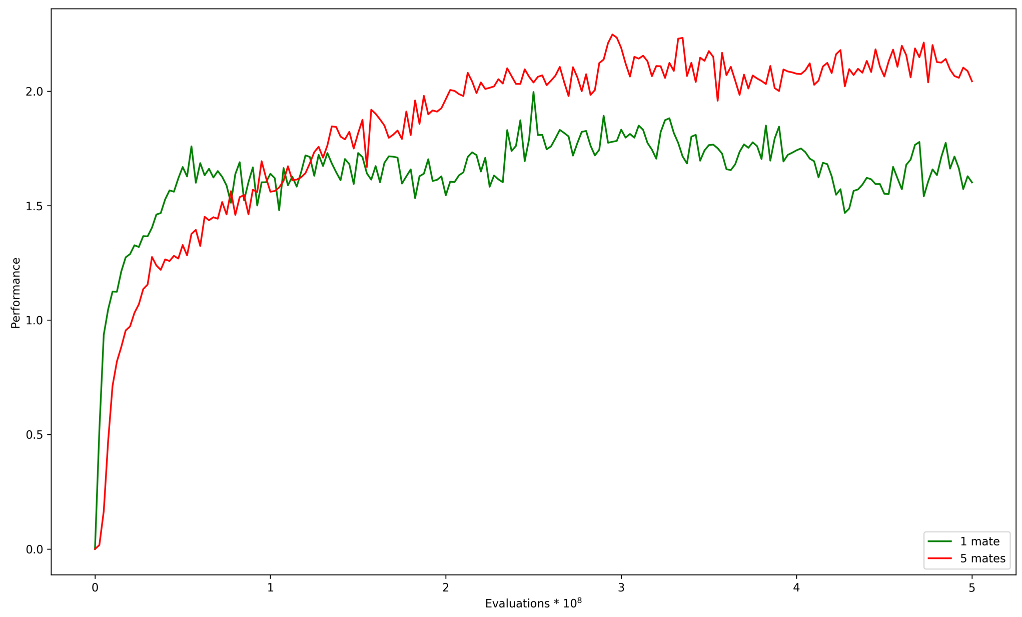

Figure 6: Performance of the evolved agents during evolution in the foraging scenario. Green curve indicates the average performance obtained by the GGA, while the red curve represents the fitness achieved by applying the *n-mates evaluation* procedure. Data have been obtained by averaging 20 replications of the experiment.

the GGA algorithm in all three problems (Mann-Whitney U test, $p < 0.05$). This implies that the former method leads to the discovery of truly effective and adaptive solutions. It is worth noting that this outcome has been obtained at no computational cost.

Figures 6-8 indicate that the *n-mates evaluation* method requires a bootstrap phase during the first part of the evolution before reaching performances comparable to those of the GGA. Specifically, it requires 10^8 evaluation steps in the case of the "foraging" and the "escaping from predator" problems (Figures 6 and 7) and 2×10^8 evaluation steps in the case of the "aggregation" task (Fig. 8). This is not surprising: at the beginning of the evolution, individuals are generally bad at dealing with the task. In collective problems involving the cooperation of heterogeneous agents, evaluating each individual with n random mates is more likely to produce lower performance than the GGA, which evaluates an agent with a single peer. Notwithstanding this initial slow convergence, the *n-mates evaluation* procedure is able to evolve effective and generalizing agents, which adapt their behavior based on the partners' capabilities.

Our analyses reveal differences among the evolutionary problems. Specifically, both the *n-mates evaluation* method and the GGA algorithm obtain good performances on the "foraging" and the "escaping from predator" problems, while their fitness is quite low in the case of the "aggregation" task. This difference might be partially ascribed to the different stimuli agents perceive. In fact, in the former problems, the robots may only receive two types of visual inputs

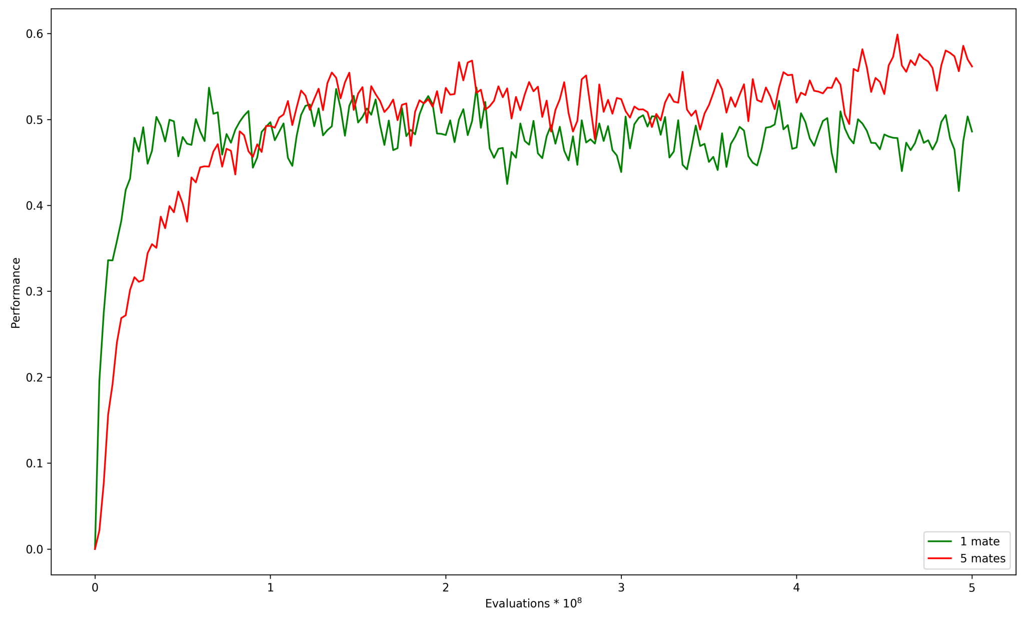

Figure 7: Performance of the evolved agents during evolution with respect to the predator case. Green curve indicates the average performance obtained by the GGA, while the red curve represents the fitness achieved by applying the *n-mates evaluation* procedure. Data have been obtained by averaging 20 replications of the experiment.

provided by both the peer (blue input of the linear camera sensor) and another object that can be either the food item or the predator. Conversely, in the latter task, the agent has to deal simultaneously with both visual stimuli provided by the camera (one from the mate and one from the cylinder inside the area) and the information obtained through the ground sensor. Put in other words, the agent has to properly manage two different types of inputs and map them into an effective behavior. Therefore, the "aggregation" problem is more complex than the other tasks. In addition, this task is strongly more dependent on the partner's capabilities than the other problems. Indeed, given the limited dimension of the cylinder object (see section 3.3), the agents must develop an exploratory capability in order to look for the aggregation area and reach it. However, if the peer does not display the same ability, the performance of the evolving agent will be low even if it succeeds in quickly arriving at the target area.

5. Conclusions

MAS is a well-established paradigm across various domains, due to features like flexibility, decentralization and adaptation to environmental changes. Extensive research has demonstrated the effectiveness of applying MAS to different fields. However, exploiting the potentiality of MASs in dealing with heterogeneous agents presents some challenges, mainly dependent on

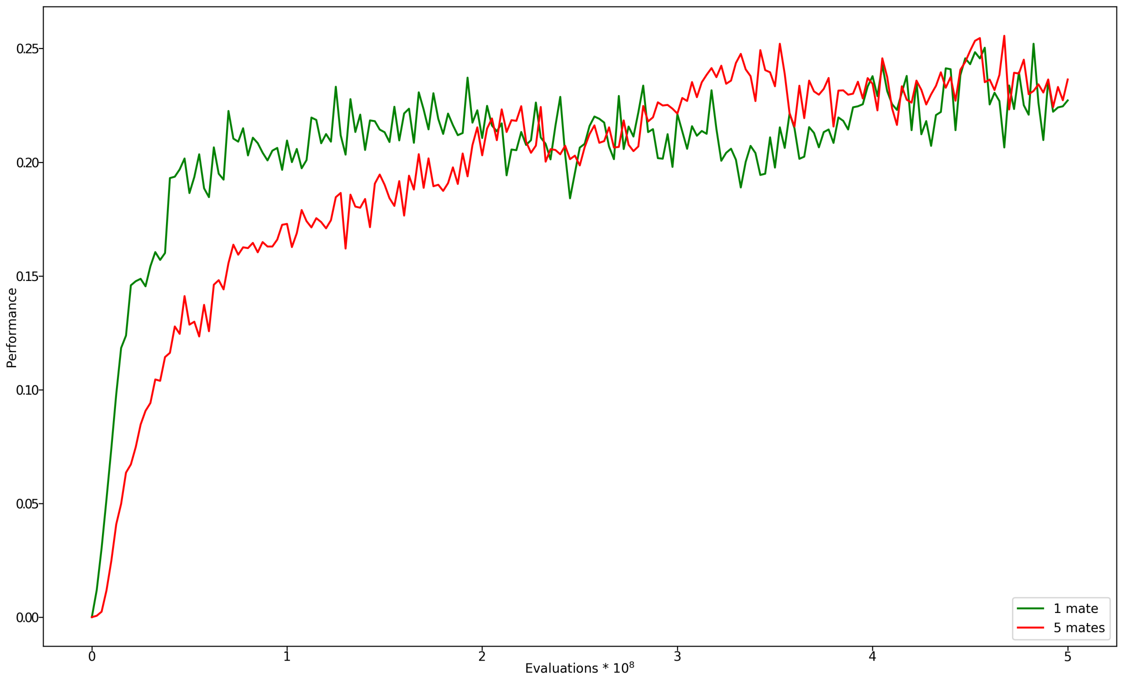

Figure 8: Performance of the evolved agents during evolution with respect to the aggregation problem. The green curve indicates the average performance obtained by the GGA, while the red curve represents the fitness achieved by applying the *n-mates evaluation* procedure. Data have been obtained by averaging 20 replications of the experiment.

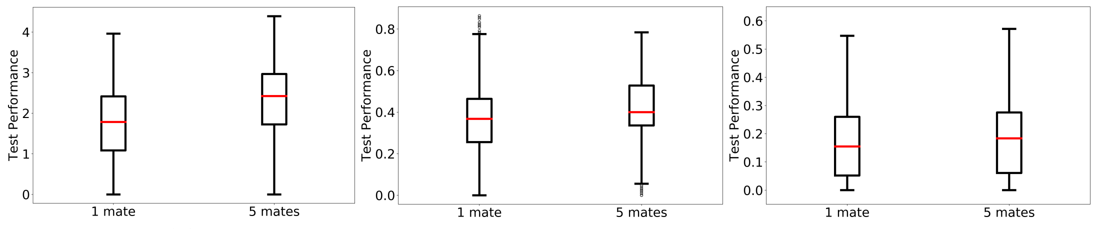

Figure 9: Performance of the evolved individuals in the post-evaluation phase for each task: **(a)** Foraging. **(b)** Escaping from predator. **(c)** Aggregation. Boxes illustrate the inter-quartile range of the data, with the median value indicated by the horizontal line inside each box. The whiskers extend to the most extreme data points that fall within 1.5 times the inter-quartile range from the box. These data points were collected from 20 repetitions.

the individual's characteristics and the specific environmental conditions. In this work, we presented the *n-mates evaluation* technique, a novel method able to improve the performance of a genetic algorithm and, at once, estimate better the actual capability of an individual in heterogeneous MASs. Specifically, we designed three scenarios in which two agents have to collaborate so as to solve the evolutionary problems. We demonstrated how our approach is effective and outperforms a standard genetic algorithm. Moreover, the evolved individuals represent truly effective solutions, i.e., they exhibit an adaptive behavior regardless of the capability of the partners they are evaluated with. Notably, the proposed method allows the discovery of adaptive agents without adding computational complexity to the genetic algorithm it is combined with since it only affects the agent's evaluation. However, the technique leads to the evolution of a population of flexible agents experiencing different environmental conditions, where the partner's capabilities are unknown. This, in turn, allows for the optimization of the exploration-exploitation trade-off because the evolved individuals can immediately adapt to new random mates without incurring additional costs. As a result, the search space exploration is reduced, and the evolutionary process can benefit from the solutions found thus far.

The results of our analysis are promising, but further research is needed to validate the approach more comprehensively. In the future, we plan to extend our study to larger groups of agents, similarly to the experiments reported in [21, 22]. We are also interested in evaluating the technique in more challenging domains, in which the capability of each individual is paramount. Moreover, the parameter n has been set empirically. Future work should be devoted to the analysis of the relationship between the value of n and the final performance. In addition, we are investigating the possibility to adapt the parameter n based on the individual's fitness, without any need to fix it a priori. Finally, we are considering the use of the *n-mates evaluation* procedure in combination with modern evolutionary algorithms, such as OpenAI-ES [20, 57, 58], which proved their efficacy in many problems including a swarm robotics scenario [21].

References

[1] P. G. Balaji, D. Srinivasan, An Introduction to Multi-Agent Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 1–27.

[2] W. Van der Hoek, M. Wooldridge, Multi-agent systems, Foundations of Artificial Intelligence 3 (2008) 887–928.

[3] M. Wooldridge, An Introduction to Multiagent Systems, 2 ed., Wiley, Chichester, UK, 2009.

[4] J. H. Holland, Adaptation in natural and artificial systems, Univ. Mich. Press (1975).

[5] E. Bahgeçi, E. Sahin, Evolving aggregation behaviors for swarm robotic systems: A systematic case study, in: Proc. IEEE Swarm Intelligence Symposium, 2005, pp. 333–340.

[6] P. Pagliuca, D. Inglese, A. Vitanza, Measuring emergent behaviors in a mixed competitive-cooperative environment, International Journal of Computer Information Systems and Industrial Management Applications 15 (2023) 69–86.

[7] R. Patel, E. Rudnick-Cohen, S. Azarm, M. Otte, H. Xu, J. W. Herrmann, Decentralized task allocation in multi-agent systems using a decentralized genetic algorithm, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2020, pp. 3770–3776.

[8] T. Shibata, T. Fukuda, Coordinative behavior in evolutionary multi-agent system by genetic algorithm, in: IEEE International Conference on Neural Networks, IEEE, 1993, pp. 209–214.

[9] V. Trianni, R. Groß, T. H. Labella, E. Şahin, M. Dorigo, Evolving aggregation behaviors in a swarm of robots, in: European Conference on Artificial Life, Springer, 2003, pp. 865–874.

[10] V. Trianni, S. Nolfi, M. Dorigo, Evolution, self-organization and swarm robotics, in: Swarm Intelligence, Springer, 2008, pp. 163–191.

[11] Y. Yang, S. Li, X. Ge, Q.-L. Han, Event-triggered cluster consensus of multi-agent systems

via a modified genetic algorithm, IEEE Transactions on Neural Networks and Learning Systems (2022).

[12] J. Zhou, X. Zhao, X. Zhang, D. Zhao, H. Li, Task allocation for multi-agent systems based on distributed many-objective evolutionary algorithm and greedy algorithm, Ieee Access 8 (2020) 19306–19318.

[13] D. E. Golberg, Genetic algorithms in search, optimization, and machine learning, Addion wesley 1989 (1989) 36.

[14] M. Mitchell, Genetic algorithms: An overview., Complex. 1 (1995) 31–39.

[15] J. McCall, Genetic algorithms for modelling and optimisation, Journal of Computational and Applied Mathematics 184 (2005) 205–222.

[16] I.-K. Jeong, J.-J. Lee, Evolving multi-agents using a self-organizing genetic algorithm, Applied Mathematics and Computation 88 (1997) 293–303.

[17] J.-P. Vacher, T. Galinho, F. Lesage, A. Cardon, Genetic algorithms in a multi-agent system, in: Procs. IEEE International Joint Symposia on Intelligence and Systems, 1998, pp. 17–26.

[18] W. Gruszczyk, H. Kwasnicka, Coalition formation in multi-agent systems—an evolutionary approach, in: International Multiconference on Computer Science and Information Technology, IEEE, 2008, pp. 125–130.

[19] P. Pagliuca, S. Nolfi, Robust optimization through neuroevolution, PLOS ONE 14 (2019) 1–27.

[20] P. Pagliuca, N. Milano, S. Nolfi, Efficacy of modern neuro-evolutionary strategies for continuous control optimization, Frontiers in Robotics and AI 7 (2020) 98.

[21] P. Pagliuca, A. Vitanza, Self-organized aggregation in group of robots with OpenAI-ES, in: Int. Conf. on Soft Computing and Pattern Recognition, Springer, 2022, pp. 770–780.

[22] P. Pagliuca, A. Vitanza, Evolving aggregation behaviors in swarms from an evolutionary algorithms point of view, in: Applications of Artificial Intelligence and Neural Systems to Data Science, Springer, 2023, pp. 317–328.

[23] J. Yang, Z. Luo, Coalition formation mechanism in multi-agent systems based on genetic algorithms, Applied Soft Computing 7 (2007) 561–568.

[24] M. Dorigo, G. Theraulaz, V. Trianni, Reflections on the future of swarm robotics, Science Robotics 5 (2020) eabe4385.

[25] M. Dorigo, G. Theraulaz, V. Trianni, Swarm robotics: Past, present, and future [point of view], Proceedings of the IEEE 109 (2021) 1152–1165.

[26] J. T. Carvalho, S. Nolfi, The role of morphological variation in evolutionary robotics: Maximizing performance and robustness, Evolutionary Computation (2023) 1–18.

[27] N. Milano, J. T. Carvalho, S. Nolfi, Environmental variations promotes adaptation in artificial evolution, in: IEEE Symposium Series on Comput. Intell. (SSCI), 2017, pp. 1–7.

[28] D. E. Goldberg, J. Richardson, et al., Genetic algorithms with sharing for multimodal function optimization, in: Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms, volume 4149, Hillsdale, NJ: Lawrence Erlbaum, 1987.

[29] T. Alam, S. Qamar, A. Dixit, M. Benaida, Genetic algorithm: Reviews, implementations, and applications, arXiv preprint arXiv:2007.12673 (2020).

[30] H. Dawid, Adaptive learning by genetic algorithms: Analytical results and applications to economic models, Springer Science & Business Media, 2011.

[31] C. García-Martínez, F. J. Rodriguez, M. Lozano, Genetic algorithms, in: Handbook of heuristics, Springer, 2018, pp. 431–464.
[32] A. Ghaheri, S. Shoar, M. Naderan, S. S. Hoseini, The applications of genetic algorithms in medicine, Oman medical journal 30 (2015) 406.
[33] I. Gómez-Vargas, J. B. Andrade, J. A. Vázquez, Neural networks optimized by genetic algorithms in cosmology, Physical Review D 107 (2023) 043509.
[34] S. Nolfi, O. Miglino, D. Parisi, Phenotypic plasticity in evolving neural networks, in: Proceedings of PerAc'94. From Perception to Action, IEEE, 1994, pp. 146–157.
[35] V. Trianni, E. Tuci, M. Dorigo, Evolving functional self-assembling in a swarm of autonomous robots, From Animals to Animats 8 (2004) 405–414.
[36] S. Tung Ngo, J. Jafreezal, G. Hoang Nguyen, A. Ngoc Bui, A genetic algorithm for multi-objective optimization in complex course timetabling, in: 2021 10th International Conference on Software and Computer Applications, 2021, pp. 229–237.
[37] A. P. Wieland, Evolving neural network controllers for unstable systems, in: International Joint Conference on Neural Networks, volume 2, IEEE, 1991, pp. 667–673.
[38] A. Liu, Y. Zhang, H. Zhao, S. Wang, D. Sun, Neural network control system of cooperative robot based on genetic algorithms, Neural Computing and Applications 33 (2021) 8217–8226.
[39] T. Salloom, X. Yu, W. He, O. Kaynak, Adaptive neural network control of underwater robotic manipulators tuned by a genetic algorithm, Journal of Intelligent & Robotic Systems 97 (2020) 657–672.
[40] X. Yao, A review of evolutionary artificial neural networks, International journal of intelligent systems 8 (1993) 539–567.
[41] J. J. Grefenstette, et al., Genetic algorithms for changing environments, in: Ppsn, volume 2, Citeseer, 1992, pp. 137–144.
[42] F. Gomez, R. Miikkulainen, Incremental evolution of complex general behavior, Adaptive Behavior 5 (1997) 317–342.
[43] Q. Lu, G. M. Fricke, J. C. Ericksen, M. E. Moses, Swarm foraging review: Closing the gap between proof and practice, Current Robotics Reports 1 (2020) 215–225.
[44] G. F. Miller, D. Cliff, Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics, From animals to animats 3 (1994) 411–420.
[45] E. H. Ostergaard, G. S. Sukhatme, M. J. Matari, Emergent bucket brigading: a simple mechanisms for improving performance in multi-robot constrained-space foraging tasks, in: Proc. of the fifth international conference on Autonomous agents, 2001, pp. 29–30.
[46] A. F. Winfield, Towards an engineering science of robot foraging, Distributed autonomous robotic systems 8 (2009) 185–192.
[47] G. Massera, T. Ferrauto, O. Gigliotta, S. Nolfi, FARSA: An Open Software Tool for Embodied Cognitive Science, in: Proceedings of the 12th European Conference on Artificial Life (ECAL 2013), 2013, pp. 538–545.
[48] G. Massera, T. Ferrauto, O. Gigliotta, S. Nolfi, Designing adaptive humanoid robots through the farsa open-source framework, Adaptive Behavior 22 (2014) 255–265.
[49] J. T. Carvalho, S. Nolfi, Behavioural plasticity in evolving robots, Theory in Biosciences 135 (2016) 201–216.
[50] J. de Greeff, S. Nolfi, Evolution of implicit and explicit communication in mobile robots,

in: Evolution of communication and language in embodied agents, Springer, 2009, pp. 179–214.

[51] P. Pagliuca, S. Nolfi, Integrating learning by experience and demonstration in autonomous robots, Adaptive Behavior 23 (2015) 300–314.

[52] V. Sperati, V. Trianni, S. Nolfi, Evolving coordinated group behaviours through maximisation of mean mutual information, Swarm Intelligence 2 (2008) 73–95.

[53] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, A. Martinoli, The e-puck, a Robot Designed for Education in Engineering, in: Proceedings of the 9th conference on autonomous robot systems and competitions, IPCB: Instituto Politécnico de Castelo Branco, 2009, pp. 59–65.

[54] R. Jeanson, C. Rivault, J.-L. Deneubourg, S. Blanco, R. Fournier, C. Jost, G. Theraulaz, Self-organized aggregation in cockroaches, Animal behaviour 69 (2005) 169–180.

[55] S. Garnier, C. Jost, R. Jeanson, J. Gautrais, M. Asadpour, G. Caprari, G. Theraulaz, Aggregation behaviour as a source of collective decision in a group of cockroach-like-robots, in: European conference on artificial life, Springer, 2005, pp. 169–178.

[56] F. Silva, L. Correia, A. L. Christensen, Evolutionary online behaviour learning and adaptation in real robots, Royal Society open science 4 (2017) 160938.

[57] P. Pagliuca, S. Nolfi, The dynamic of body and brain co-evolution, Adaptive Behavior 30 (2022) 245–255.

[58] T. Salimans, J. Ho, X. Chen, S. Sidor, I. Sutskever, Evolution strategies as a scalable alternative to reinforcement learning (2017).