Using Models of Combinatorial Optimization Problem to Estimate the Parameters of an Intelligent System

Liudmyla Koliechkina[1], and Olena Dvirna[2]

[1] *University of Lodz, 68 Narutowicza, Lodz, 90136, Poland*
[2] *National University "Yuri Kondratyuk Poltava Polytechnic", 24 Pershotravneva Ave., Poltava, 36011, Ukraine*

Abstract

This article is devoted to studying methods of designing intelligent systems using combinatorial optimization. The main concepts and approaches to the creation of intelligent systems are considered, in particular the stages of their life cycle, which include requirements definition, architecture design, development, training, testing and implementation. Special attention is paid to the use of combinatorial optimization methods, which allow solving complex problems related to optimization and decision-making in various contexts.

A mathematical model of module selection in the development of an intelligent system using the properties of combinatorial configurations is constructed.

The article also discusses the prospects for the use of intelligent systems in various fields, including industry, health care, and resource management. The main goal of this work is to emphasize the importance of integrating combinatorial optimization methods into the design process of intelligent systems to increase their efficiency and adaptability.

Keywords

Intelligent systems, combinatorial optimization, vector optimization problem, permutation configuration

1. Introduction

As is known, an intelligent system can be a software system capable of solving tasks traditionally considered creative, belonging to a specific subject area, with knowledge about this area stored in the system's memory. According to [1], the structure of an intelligent system includes three main components: a knowledge base, a decision-making mechanism, and an intelligent interface. In decision-making technologies, an intelligent system is an information-computing system with intelligent support, solving tasks without human participation, that is, without the decision-maker (DM), unlike an intelligentized system, where the operator is present. Different types of intelligent systems are distinguished, such as an intelligent information system, an expert system, a computational-logical system, which can also be a hybrid system, and a reflexive intelligent system.

Computational-logical systems include systems capable of solving management and design tasks based on declarative descriptions of conditions. In this case, the user has the ability to control all stages of the computational process in interactive mode. These systems can automatically construct a mathematical model of the problem and automatically synthesize computational algorithms based on the problem's formulation [2-6]. These capabilities are implemented due to the presence of a knowledge base in the form of a functional semantic network and components for deductive reasoning and planning. The use of intelligent systems in text recognition on images.

An Intelligent Information System (IIS) is a set of software, linguistic, and logical-mathematical tools designed to perform the primary task of supporting human activities and searching for information in an advanced natural language dialogue mode [1].

Intelligent information systems and combinatorics are closely related, as many tasks solved within the framework of intelligent systems are combinatorial in nature.

Let's consider the main aspects of this connection.

ProfIT AI 2024: 4th International Workshop of IT-professionals on Artificial Intelligence (ProfIT AI 2024), September 25–27, 2024, Cambridge, MA, USA

lkoliechkina@gmail.com (L. Koliechkina); lenadvirna@gmail.com (O. Dvirna)

0000-0002-4079-1201 (L. Koliechkina); 0000-0002-0750-6958 (O. Dvirna)

1.1. Second level sectioning

- Combinatorics deals with studying various ways of combining objects into sets according to given rules. This includes problems of selection, arrangement, permutations, and other operations on finite sets.
- Intelligent systems often encounter combinatorial problems when solving issues such as optimization, planning, routing, pathfinding, the knapsack problem, the traveling salesman problem, and others. These problems require efficient algorithms to find the optimal or feasible solution among a multitude of possible combinations.

1.2. The Role of Combinatorics in Algorithms of Intelligent Systems

- **Search and Optimization**: Intelligent systems often use combinatorial methods to find the best solution among many possible options. For example, graph search methods (depth-first search, breadth-first search), heuristic methods (such as A* search algorithms, genetic algorithms) use combinatorial approaches to select paths or combinations.
- **Planning and Decision Making:** Planners and decision support systems are often built on combinatorial methods, which allow for generating and evaluating possible scenarios and actions, selecting the best option from many.
- **Cryptography and Data Protection**: In cryptographic systems, combinatorial methods are used to create and analyze complex encryption keys and to test their security.

1.3. Combinatorial Algorithms in Intelligent Systems

- **Genetic Algorithms:** These algorithms use principles of biological evolution and work with populations of solutions that are combined, mutated, and selected to find the optimal solution. Combinatorics plays a key role in the process of selecting and combining solutions.
- **Branch and Bound Methods:** These are combinatorial optimization methods used to solve problems where many possible solutions need to be explored, while pruning those that are clearly not optimal.
- **Machine Learning Methods:** Combinatorial methods are applied in hyperparameter selection, building model ensembles, feature selection, and other learning tasks.

1.4. Application Examples

- **Traveling Salesman Problem (TSP)**: The Traveling Salesman Problem, where the goal is to find the shortest route between a set of cities, is a classic example of a combinatorial problem often solved in intelligent systems.
- **Pattern Recognition**: In pattern recognition tasks, combinatorial methods are used to compare and select the most suitable patterns or features that match the given images.
- **Resource Allocation**: In resource allocation tasks (e.g., scheduling problems), combinatorial approaches are used to efficiently assign limited resources to multiple tasks.

Thus, intelligent systems utilize combinatorics and combinatorial properties of sets of arrangements, permutations, and polypermutations to solve problems that require considering a large number of options and finding the optimal solution. Examples of such problems are discussed in works [5-12]. Combinatorial methods provide effective ways of generating, evaluating, and optimizing a multitude of solutions, which is key to the successful operation of intelligent systems [5-8, 17-19]. Combinatorics is a fundamental part of intelligent systems, offering methods and algorithms for solving a wide range of problems related to search, optimization, and planning in the presence of a large number of possible combinations [18-24].

The article will consider the combinatorial model of the problem of choosing modules when developing a program for an intelligent system, analyze it, and propose an approach to its solution.

It should be noted that the life cycle of an intelligent system (IS) encompasses many stages of development, implementation, operation, and maintenance of the system. It includes a sequence of steps aimed at creating an effective and reliable intelligent system capable of solving complex problems and making decisions based on data analysis. The main stages of the life cycle of an intelligent system can be divided into the following phases:

1. Requirements Definition: This involves gathering and analyzing requirements. At this stage, the system's objectives, user requirements, and both functional and non-functional requirements are determined. It is crucial to understand what problems the system should solve, what data will be used, and what results are expected. This step includes formulating the problem.
2. System Design: This involves developing the system architecture, selecting appropriate methods and algorithms for data processing and decision-making. It includes detailing individual system components, such as data collection modules, analysis, machine learning, decision-making, user interfaces, and so on.
3. System Development: At this stage, code is written and software modules are developed to perform specific functions of the system. Integration of components involves combining all modules into a unified system, ensuring their interaction and coordinated operation.
4. Training and Adaptation.
5. Testing and Validation: This involves checking each module against the requirements, as well as testing the system as a whole. Validation includes assessing the accuracy, performance, and reliability of the system, and verifying its operation with real data.
6. Deployment.
7. Operation.
8. Maintenance and Updates.
9. Decommissioning.

This life cycle can be repeated several times during the process of system enhancement to adapt to new challenges and requirements.

Next, we will consider one of the stages of the intelligent system life cycle—coding and software module development.

2. Problem statement. Module Selection Problem in the Development of the Software Component of an Intelligent System

One of the fundamental principles of modern intelligent system design is the principle of modularity, which allows for more effective management of various stages of the intelligent system life cycle, such as creation, deployment, maintenance, and enhancement of software and computational resources. The principle of modularity involves developing and implementing a program as a collection of components – modules.

Let's describe the model of the module selection problem in software development. We will consider the problem of optimal program composition, which consists of several modules, with the condition that some of these modules may be implemented on a computer in multiple ways.

At the algorithm development stage, a program can be represented as consisting of n individual interconnected blocks (modules, procedures, programs, segments). For each block j ($j \in N_n$) there are X_j possible implementation options $x_j \in N_{X_j}$. According to the description, the set can be represented as a configuration of polypermutations. Each option is characterized by its execution time $t_j(x_j)$, the amount of memory it occupies, constants and arrays $v_j(x_j)$ and the required total memory $w_j(x_j)$. The goal is to select an option for each block of the program such that the program executes in minimal time T and does not exceed the allocated resources.

3. Mathematical Modeling

Mathematical formulation of the Problem: find the value of x_j such that

$$T = \sum_{j=1}^{n} t_j(x_j) \to \min,$$

$$M = \sum_{j=1}^{n} v_j(x_j) \to \min$$

subject to the constraint $\sum_{j=1}^{n} v_j(x_j) + \max_{1 \le j \le n} w_j(x_j) \le V$,

where V –is the amount of memory allocated for the optimizing program, and the solution belongs to the configuration of polypermutations.

The described mathematical model corresponds to a vector optimization problem on a combinatorial configuration of polypermutations, which can be solved using the algorithms described in [14, 20].

Information arrays in computer memory can be placed at different levels of the hierarchy, each of which may contain one or more memory devices (MDs) of the same or different types with approximately the same speed. The most important parameters of MDs are capacity and speed. Generally, higher-level MDs have higher speed but lower capacity compared to lower-level MDs. We will describe the model of the optimal distribution of arrays across the levels of computer memory.

Arrays are characterized by size and activity. Activity Y refers to the frequency of use or the expected number of accesses to the array over a certain period of time during the operation of the software system. The total time spent accessing the array when solving a set of problems in the computing system depends on how arrays are distributed across memory levels. Therefore, the problem of optimally distributing arrays across hierarchical memory levels arises.

Let T be the total access time to the array;

M the amount of memory;

m be the total number of memory devices (MDs) at all levels of the hierarchy;

V_i, t_i be the capacity and speed of the i -th MD; ($i \in N_m$);

n be the number of arrays;

Y is array usage activity

v_j, r_j be the size and activity of the j-th array, respectively ($j \in N_n$);

$X = (x_1, x_2, .., x_n)$, where $x_j \in N_m$ be the vector representing the distribution of arrays across MDs. The vector can be represented as an element of a permutation configuration [14, 17].

Mathematical model of the problem: find the value such that

$$T = \sum_{j=1}^{n} \tau_j(x_j) \to \min,$$

$$Y = \sum_{j=1}^{n} r_j x_j \to \max$$

subject to the constraint

$$\sum_{j=1}^{n} w_{ij}(x_j) \le V_i,\ i \in N_m,$$

where $\tau_j(x_j) = t_i r_j,\ i = x_j,\ w_{ij}(x_j) = \begin{cases} v_i, & if\ x_j = i, \\ 0 & if\ x_j \ne i. \end{cases}$

This problem is a vector optimization problem on permutations [3, 14].

To solve such problems, vector optimization methods can be used at the first stage, and combinatorial optimization methods are used at the second stage. Some methods of combinatorial optimization are presented in the next part.

4. Solving of the Problem

In the design of intelligent systems, especially those addressing complex optimization problems, various combinatorial optimization methods can be used. The main methods that may be useful are:

Branch and Bound: This method is used to find the optimal solution in problems where a subset of elements needs to be selected or a series of discrete decisions must be made. It involves systematically exploring all possible solutions (combinations) by dividing the problem into subproblems (branches) and pruning (bounds) those branches that cannot lead to an optimal solution.

Dynamic Programming: This method is suitable for problems that can be broken down into interdependent, repetitive subproblems. The idea is to store the results of subproblem solutions and reuse them, significantly reducing computational complexity. Examples include the knapsack problem or the traveling salesman problem for a limited number of cities.

Genetic Algorithms: This heuristic method, inspired by principles of natural selection and genetics, is used to find a "good enough" solution in large search spaces where exact enumeration is too computationally intensive. It is especially useful for problems that require optimizing multiple criteria simultaneously or where the search space is very large.

Simulated Annealing: This heuristic method seeks the global minimum of a function by gradually lowering the "temperature" of the search, which reduces the probability of accepting worse solutions in later stages. It is used for problems where the search space is large but helps avoid local minima.

Ant Colony Optimization (ACO): This method builds paths from the starting state to the goal using artificial "pheromones." It is used in problems where the shortest path or optimal route needs to be found, such as the traveling salesman problem.

Tabu Search: This method uses the idea of iterative improvement, where each new state is checked to ensure it is not in the forbidden (tabu) list. This helps avoid cycles and local minima. It is often used for solving placement, cutting, and scheduling problems.

Particle Swarm Optimization (PSO): This method is based on modeling the behavior of a group of particles moving in the search space, coordinating their actions. Particles move toward better solutions, considering their own experience and that of other particles. It is effective for multi-criteria optimization problems and finding global extrema.

Greedy Algorithm: This method makes locally optimal choices at each step, hoping these choices will lead to a globally optimal solution. It is effective for problems where greedy solutions approximate the optimal global solution, such as in set covering or maximum flow problems.

These combinatorial optimization methods can be integrated into intelligent systems to solve a wide range of problems, such as resource optimization, scheduling, logistics, and many others, where finding the best solution among many possible options is required.

5. Implementation of the Results

The described mathematical model provides a robust framework for optimizing the distribution of arrays across hierarchical memory levels. Its implementation can lead to significant performance improvements, efficient resource utilization, and adaptability to varying workload conditions. The model is particularly useful in high-performance computing, cloud computing, and systems with complex memory hierarchies. By leveraging vector optimization on permutations, it can handle complex configurations and constraints effectively, paving the way for advanced memory management solutions.

6. Conclusions

The concept of an intelligent system is considered, the connection of the system with the problems of combinatorial optimization is analyzed on the basis of the selected literature. Examples of the application of combinatorial algorithms in intelligent systems are given. An applied problem of choosing modules in the development of the software part of an intelligent system is presented and a mathematical model is built. The application of such a model is described and an approach to its solution is proposed. Further research is aimed at the study of numerical experiments for the problem of choosing modules in the development of the software part of the intellectual.

References

[1] Naghshvarianjahromi, M.; Kumar, S.; Deen, M.J. Natural Intelligence as the Brain of Intelligent Systems. *Sensors 23*, 2859 (2023). https://doi.org/10.3390/s23052859

[2] Kaltdorf, M., Breitenbach, T., Karl, S. et al. Software JimenaE allows efficient dynamic simulations of Boolean networks, centrality and system state analysis. Sci Rep 13, 1855 (2023). https://doi.org/10.1038/s41598-022-27098-7

[3] Liudmyla Koliechkina, Olena Dvirna, Serhii Khovben, Multi-Criteria Decision Discrete Model for Selecting Software Components in Component-Based Development, 3rd International Workshop of IT-professionals on Artificial Intelligence 2023, (ProfIT AI 2023), Waterloo, Canada, November 20-22, 2023, pp.182-193

[4] L. Koliechkina, T. Hudz and V. Kylnyk, "Modeling of Bank Performance Indicators Based on Business Intelligence and Data Analysis," 2023 IEEE 13th International Conference on Electronics and Information Technologies (ELIT), Lviv, Ukraine, 2023, pp. 87-92

[5] Pichugina, O., Yakovlev, S. Optimization on polyhedral-spherical sets: Theory and applications. 2017 IEEE 1st Ukraine Conference on Electrical and Computer Engineering, UKRCON 2017 - Proceedings, art. no. 8100436, 2017, pp. 1167-1174. doi: 10.1109/UKRCON.2017.8100436

[6] Yakovlev, S.V. The theory of convex continuations of functions at the vertices of convex polygons. Computational Mathematics and Mathematical Physics, 34 (7), 1994, pp. 959-965

[7] Emmerich, M., Deutz, A., 2018. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. Nat. Comput. 17.

[8] Stoyan, Yu.G., Sokolovskii, V.Z., Yakovlev, S.V. Method of Balancing Rotating Discretely Distributed Masses. Energomashinostroenie, №2, 1982, pp. 4-5.

[9] L. Lavagno, I. L. Markov, G. Martin, and L. K. Scheffer, Eds., Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology, 2edition. Boca Raton: CRC Press, 2016, https://doi.org/10.1201/9781315215112

[10] Di Puglia Pugliese, L., Granat, J., Guerriero, F., 2020. Two-phase algorithm for solving the preference-based multicriteria optimal path problem with reference points. Comput. Oper. Res. 121, 104977.

[11] Bökler, F. K., 2018. Output-Sensitive Complexity of Multiobjective Combinatorial Optimization with an Application to the Multiobjective Shortest Path Problem (Ph.D. thesis). Technische Universität Dortmund.
[12] Odu G. O. and Charles-Owaba O. E. (2013). Review of Multi-criteria Optimization Methods – Theory and Applications. IOSR Journal of Engineering (IOSRJEN), vol. 3, Issue 10, pp. 01 – 14.
[13] Panos M. Pardalos, Antanas Žilinskas, and Julius Žilinskas. Non-Convex Multi-Objective Optimization. Springer Optimization and Its Applications. Springer International Publishing, 2017.
[14] Koliechkina, L.N., Dvirna, O.A. & Khovben, S.V. A Two-Step Method for Solving Vector Optimization Problems on Permutation Configuration. Cybern Syst Anal 57, 442–454 (2021). https://doi.org/10.1007/s10559-021-00369-3
[15] Bernhard Korte and Jens Vygen. Combinatorial Optimization: Theory and Algorithms. Springer, 5th edition, 2012.
[16] Pardalos P.M. Handbook of combinatorial optimization / P.M. Pardalos, D-Z. Du, R.L.Graham. – New York: Springer, 2013. – 3409 p.
[17] Donets, G.P., Koliechkina, L.N., Nahirna, A.N. A Method to Solve Conditional Optimization Problems with Quadratic Objective Functions on the Set of Permutations. Cybernetics and Systems Analysis. 2020. Vol. 56, N 2. P. 278-288.
[18] B. Korte, and J. Vygen, Combinatorial Problems in Chip Design, in Building Bridges, Springer, Berlin, Heidelberg, 2008, pp. 333–368.
[19] Yemelichev V.A., Kovalev M.M., Kravtsov M.K. Polytopes, graphs and optimisation. Cambridge University Press, Cambridge. 1984. 243p.
[20] Koliechkina L.N., Nagornaya A.N., Semenov V.V. Method of solving problem of conditional optimization on combinatorial set of arrangements. Journal of Automation and Information Sciences. 2019. 51, N.8. P. 31-42.
[21] K. Kawarabayashi and Y. Kobayashi, The Induced Disjoint Paths Problem, in Integer Programming and Combinatorial Optimization, A. Lodi, A. Panconesi, and G. Rinaldi, Eds. Springer Berlin Heidelberg, 2008, Pp. 47–61.
[22] Yakovlev, S.V. Bounds on the minimum of convex functions on Euclidean combinatorial sets. Cybernetics, 25 (3), 1989pp. 385-391. doi: 10.1007/BF01069996
[23] Stoyan, Y.G., Yakovlev, S.V., Emets, O.A., Valuiskaya, O.A. Construction of convex continuations for functions defined on a hypersphere. Cybernetics and Systems Analysis, 34 (2), 1998, pp. 176-184. doi: 10.1007/BF02742066
[24] Pichugina, O., Yakovlev, S. Functional and analytic representations of the general permutations. Eastern-European Journal of Enterprise Technologies, 1 (4), 2016, pp. 27-38. doi:10.15587/1729-4061.2016.58550