Module Selection Optimization via Combinatorial Techniques in Intelligent Systems*

Liudmyla Koliechkina[1,*,†], Oksana Pichugina[2,†], Yurii Skob[2,†] and Olena Dvirna[3,†]

[1]*University of Lodz, 68 Gabriela Narutowicza Str., Lodz, 90-136, Poland*
[2]*National Aerospace University "Kharkiv Aviation Institute", 17 Vadym Manko St, Kharkiv, 61070, Ukraine*
[3]*National University "Yuri Kondratyuk Poltava Polytechnic", 24 Vitaliya Hrytsayenka, Poltava, 36011, Ukraine*

Abstract

This paper attacks the problem of optimal module selection and program layout in intelligent information systems through Euclidean combinatorial optimization. Two mathematical models are introduced: module selection under execution time and memory constraints (Model 1), and distribution of data arrays across memory (Model 2). Both are formulated as linear multi-objective Euclidean combinatorial optimization problems (LMOCOP).

The study proposes a two-stage solution method for LMOCOP, which is based on utilizing Euclidean combinatorial configurations (e-configurations). In the first stage, linear convolution reduces the multiobjective optimization problem to a single-objective one. In the first stage, polyhedral a spherical relaxation and cutting planes are applied to obtain an optimal multipermutation configuration.

A numerical example illustrates the ability to generate Pareto-optimal solutions by the proposed approach to solving LMOCOP.

Keywords

combinatorial optimization, linear optimization, multi-objective optimization, mathematical modeling, intelligent system, linear convolution, cutting-plane, multipermurtation, multipermutohedron, program layout, software modularity, Euclidean combinatorial configuration

1. Introduction

Mathematical modeling methods are widely used for mathematical representation of real systems [1, 2, 3]. These methods facilitate the analysis of complex systems, enable the prediction of their behavior, and assist in finding optimal solutions [4, 5, 6]. There are various methods of mathematical modeling, each of which is adapted to a specific area of application [7, 8, 9].

The main approaches to mathematical modeling are analytical methods, numerical methods, statistical approaches, and optimization methods. In particular, optimization methods are used to determine the best solutions from a certain point of view among those that satisfy predefined constraints. This category includes integer, discrete and combinatorial optimization methods used to solve problems whose models contain one or more objective functions and additional constraints. The most studied class of problems is linear programming, in which the objective function and constraints are linear. If combinatorial conditions are present in the constraints, optimization problems fall into the combinatorial class. Accordingly, combinatorial optimization methods become applicable to them [10, 11, 12, 13, 14]. Combinatorial optimization models and methods are widely used in various practical areas, including finance, economics, and logistics [15, 16, 17].

There is a deep and close connection between mathematical modeling methods and combinatorial optimization methods, as both approaches are often used to solve complex problems arising in various

ProfIT AI'25: 5th International Workshop of IT-professionals on Artificial Intelligence, October 15–17, 2025, Liverpool, UK

*You can use this document as the template for preparing your publication. We recommend using the latest version of the ceurart style.

*Corresponding author.

†These authors contributed equally.

lkoliechkina@gmail.com (L. Koliechkina); o.pichugina@khai.edu (O. Pichugina); y.skob@khai.edu (Y. Skob); lenadvirna@gmail.com (O. Dvirna)

0000-0002-4079-1201 (L. Koliechkina); 0000-0002-7099-8967 (O. Pichugina); 0000-0003-3224-1709 (Y. Skob); 0000-0002-0750-6958 (O. Dvirna)

scientific and practical fields. Mathematical modeling provides tools for formalizing real problems, while combinatorial optimization helps to find the best solutions in such models.

After constructing a mathematical model, it is often necessary to find an optimal solution. If the model has a discrete nature (for example, the task of choosing the best combinations from a set of possible options), then combinatorial optimization methods are used. Combinatorial problems that arise in mathematical modeling can be so complex that exact methods cannot solve them in an acceptable time. That is, they belong to NP-complex problems. In such cases, mathematical models use heuristic methods of combinatorial optimization, such as genetic algorithms, the ant colony method, the tabu search method, and the branch and bound method. These methods make it possible to find close to optimal solutions for complex combinatorial problems that arise in the modeling of real systems described in works [18, 19, 20].

Mathematical modeling of complex systems, such as transport networks, information systems, energy systems, and logistics, often involves considering a large number of variables and options, making them natural candidates for combinatorial optimization.

Hence, mathematical modeling and combinatorial optimization complement each other: mathematical modeling provides tools for accurately describing real-world problems. In contrast, combinatorial optimization provides methods for finding optimal solutions in discrete option spaces. This combination is key to solving many practical problems in economics, engineering, logistics, information technologies, and complex systems [6, 21].

There is a deep and close connection between mathematical modeling methods and combinatorial optimization methods, since both approaches are often used sequentially when solving complex problems arising in various scientific and practical fields. Mathematical modeling provides tools for formalizing real-world problems, while combinatorial optimization helps find optimal solutions in such models. Many practical problems allow the construction of multiple mathematical models, each of which can be solved using specific methods. Accordingly, expanding the set of approaches to modeling a certain problem increases the chance of finding an effective solution by choosing from a wider set of methods.

After constructing a mathematical model, the optimal solution to the corresponding optimization problem is sought. If the model is discrete in nature (for example, the task of selecting the best combinations from a set of possible options), then combinatorial optimization methods are used. Combinatorial optimization models that arise in the mathematical modeling of complex practical problems usually have exponential computational complexity. Accordingly, exact methods cannot solve such problems of sufficiently large dimension within an acceptable time. In such cases, heuristic combinatorial optimization methods are widely applied to the models, such as genetic algorithms, the ant colony method, the tabu search method, and the branch and bound method etc. These methods allow finding solutions to complex combinatorial problems that are close enough to the optimal ones.

Mathematical modeling of complex systems such as transportation networks, information systems, energy systems, and logistics often involves choosing from a large but finite number of options, making such models natural candidates for attacking by combinatorial optimization approaches.

Thus, mathematical modeling and combinatorial optimization complement each other: mathematical modeling provides tools for adequately describing real-world problems. In turn, combinatorial optimization provides methods for finding optimal solutions in discrete spaces of options. This combination is key to solving many practical problems in economics, engineering, logistics, information technology, and complex systems [6, 21].

This paper examines the real-world problem of optimal selection of information system software parameters. To solve this problem, a multi-objective combinatorial optimization model on permutation configurations is proposed, along with a two-stage solution method that combines two well-known methods: linear convolution and combinatorial clipping. The study is structured as follows: the introduction is dedicated to an overview of the paper topic, while the second chapter presents the basic concepts and definitions used to construct two mathematical models of applied problems and the proposed method of their solving. The third and fourth sections are devoted to the formulation of these applied problems and their formalization as combinatorial optimization problems. The last sections provide an outline of the proposed method for solving the problem and an illustrative numerical example.

2. Prerequisites

The concept of configuration was introduced by C.Berge [22] with the aim of creating a formal and rigorous structure for defining combinatorial objects, as well as for solving problems associated with the accumulation of verbal descriptions as the complexity of these objects increases. Such formalization helps to simplify the study of combinatorial structures by reducing ambiguity in their definitions. The properties of Euclidean combinatorial configurations (e-configurations) as a separate subclass of configurations and the sets they form are discussed in detail in many works, such as [13, 20, 23]. e-configurations play an important role in combinatorial geometry and optimization.

Let the set $B = \{b_1,\ b_2, ..., b_m\}$ be given, $A = \{a_1, a_2, .., a_n\}$ be a finite set, and $\chi : B \to A$ be the mapping associating a single element $a \in A$ with each element $b \in B$, i.e. $a = \chi(b)$. According to [22], a configuration is a mapping $\chi : B \to A$ that satisfies certain constraints Λ.

Given the finiteness of sets A and B, a configuration is called combinatorial configuration (c-configuration). As a result of mapping $\chi : B \to A$ we get an ordered sequence π of A-elements:

$$\pi = \left(\begin{array}{cccc} b_1 & b_2 & ... & b_m \\ a_{j_1} & a_{j_2} & ... & a_{j_m} \end{array} \right) = [a_{j_1}, a_{j_2}, ..., a_{j_m}]\,,$$

where $j_i \in J_n, i \in J_m$ ($J_m = \{1, 2, ..., m\}$). Further, we will use the later notation $\pi = [a_{j_1}, a_{j_2}, ..., a_{j_m}]$.

In most cases, the set B can be unified, meaning that the elements of the set can be replaced by their ordinal numbers. By setting the bijective mapping between B and J_m, we obtain the transformation of the mapping into

$$\phi : J_m \to A, \tag{1}$$

where J_m is called a numbering set. Note that the configuration elements π do not change, i.e.

$$\pi = \left(\begin{array}{cccc} b_1 & b_2 & ... & b_m \\ a_{j_1} & a_{j_2} & ... & a_{j_m} \end{array} \right) = \left(\begin{array}{cccc} 1 & 2 & ... & m \\ a_{j_1} & a_{j_2} & ... & a_{j_m} \end{array} \right) = [a_{j_1}, a_{j_2}, ..., a_{j_m}] \tag{2}$$

Suppose, elements in A are strictly ordered, namely, $a_i \prec a_{i+1}, i \in J_{n-1}$. A combinatorial configuration can be represented by a tuple [13]:

$$\langle \phi, A,\ \Lambda \rangle\,, \tag{3}$$

where ϕ is the mapping (1), which satisfies a set of constraints Λ, A is a resulting set with strictly ordered elements.

This transformation redefines elements in terms of their ordinal positions, allowing for a more structured and systematic analysis of combinatorial configurations. Specifically, this approach simplifies the representation of configurations by using a standard numerical indexation instead of arbitrary elements.

Let $A^* = \{\alpha_1, \alpha_2, .., \alpha_n\}$ be a resulting set in the formation of the configuration (3) set consisting of real vectors of the same dimension k, i.e.

$$\alpha_j = (a_{1j}, a_{2j}, ..., a_{kj})^\top \in \mathbb{R}^k, \quad j \in J_n,$$

while Λ be a set of constraints allowing singling out the required configuration. Then, according to (2), $\pi = [\alpha_{j_1}, \alpha_{j_2}, ..., \alpha_{j_m}]$.

We will assign a vector to each configuration in a one-to-one correspondence $\pi = [\alpha_{j_1}, \alpha_{j_2}, ..., \alpha_{j_m}]$

$$x = (x_1, x_2, ..., x_N) \in \mathbb{R}^N,\ N = k \cdot m \tag{4}$$

whose components are an ordered set of elements of a multiset

$$\widetilde{A}(x) = \left\{ a_{1j_1}, a_{1j_2}, ..., a_{1j_m}, a_{2j_1}, a_{2j_2}, ..., a_{2j_m}, a_{kj_1}, a_{kj_2}, ..., a_{kj_m} \right\}$$

thus establishing a bijective mapping such that

$$x = \psi\left(\pi\right), \ \pi = \psi^{-1}(x). \tag{5}$$

An Euclidean combinatorial configuration (e-configuration) is called mapping $\psi : (\phi,\ A^*,\ \Theta) \to \mathbb{R}^N$, where $\phi : J_m \to A^*$, A^* is a resulting set of the form (3), Θ is a collection of constraints on the mappings ϕ and ψ.

Thus, an e-configuration is defined as a mapping of a combinatorial configuration (2) into the Euclidean space $\mathbb{R}^N$. This mapping $\phi,\ \psi$ assigns specific positions to elements of the configuration in the Euclidean space, producing a vector $x \in \mathbb{R}^N$ that represents the configuration geometrically. The vector x given by formula (4). The set $\tilde{A}$ of vectors induced by this mapping is referred to as the **inducing multiset** of the Euclidean combinatorial configuration x. This concept allows combinatorial structures to be studied using the tools of Euclidean geometry, which facilitates a more accurate analysis of their properties and interrelationships.

Let

$$E = \left\{x \in \mathbb{R}^N \ :\ x \text{ satisfies } (5)\right\}. \tag{6}$$

A set $X = E_{mk}(\tilde{A}) \subseteq \mathbb{R}^n$ of the form (6) is called a set of e-configurations of multipermutations (permutations) if an inducing multiset of each its elements coinsides with $\tilde{A}$, the multiset inducing the set X, i.e. $\forall x \in X \ A\left(x\right) = \tilde{A}$. Here and further, k is the number of different elements in $\tilde{A}$.

A set $X = E_{lk}^{m}(\tilde{A}) \subseteq \mathbb{R}^n$ of the form (6) is called a set of e-configurations of partial multipermutations (partial permutations) if an inducing multiset of each its elements is a proper subset of the multiset $\tilde{A}$ inducing the set X, i.e. $\forall x \in X \quad \tilde{A}\left(x\right) \subset \tilde{A}$. Here, $l > n$.

Without loss of generality, we can assume that $\tilde{A}$ is ordered, i.e.

$$\tilde{A} = \{\tilde{a}_1, ..., \tilde{a}_n\} : \tilde{a}_1 \leq ... \leq \tilde{a}_n.$$

In [13], it is shown that the convex hull of a set E_{nk} of e-configurations of multipermutations (multipermutation e-configurations) is a multipermutohedron $\Pi_{nk}(\tilde{A}) = convE_{nk}(\tilde{A})$, whose vertex set coincides with the set of permutations, i.e. $vert\Pi_{nk}(\tilde{A}) = E_{nk}(\tilde{A})$.

Theorem 1. *[24] The multipermutohedron $\Pi_{nk}(\tilde{A})$ is given by linear constraints:*

$$\textstyle\sum_{i \in I} x_i \leq S_{|I|}, I \subset J_n, \tag{7}$$

$$\textstyle\sum_{i=1}^{n} x_i = S_n, \tag{8}$$

where

$$S_j = \sum_{i=1}^{j} \tilde{a}_{n-i+1}, j \in J_n.$$

Theorem 2. *[25] If the point $x \in \mathbb{R}^n$ satisfies the constraints (8),*

$$x_1 \leq \ldots \leq x_n; \tag{9}$$

$$\textstyle\sum_{i=1}^{j} x_{n-i+1} \leq S_j, j \in J_{n-1}, \tag{10}$$

then $x \in \Pi_{nk}\left(\tilde{A}\right)$.

Theorem states that to check the condition $x \in \Pi_{nk}\left(\tilde{A}\right)$ for an arbitrary point in Euclidean space, it is sufficient to check just n constraints of the polytope out of $2^n - 1$.

Other properties of the polytope $\Pi_{nk}(\tilde{A})$ and its generalization, called the generalized permutohedron, can be found in the works [26, 27, 28] and a compact analytic description of a multipermutohedron is presented in [29].

Suppose that functions $f_i : X \to \mathbb{R}^1,\ i \in J_l$ are components of the optimality criterion $F(x) = (f_1(x), ..., f_l(x))$ in the following multi-criteria optimization problem: find a vector x such that

$$F(x) = (f_1(x), ..., f_l(x)) \to extr, \tag{11}$$

$$x \in D \subseteq X,$$

where X is the domain of objective functions, while $D \subseteq X$ is the feasible domain singled out from X by the constraints Λ.

Problem (11) is a problem of multiobjective Euclidean combinatorial optimization (multicriteria Euclidean combinatorial optimization problem, MOCOP). If $D = X$, this is an unconstrained optimization problem over X, otherwise it is constraint ones.

Let all components of the vector criterion be linear functions, i.e.

$$f_i(x) = c_i^\top x,\ i \in J_l, \tag{12}$$

and D is singled out from X by linear constraints $Ax \leq b$, i.e.

$$D = \{x \in X : Ax \leq b\}. \tag{13}$$

Now, the MOCOP becomes: find a vector x such that

$$f_i(x) = c_i^\top x \to extr,\ i \in J_l, \tag{14}$$

subject to constraints:

$$Ax \leq b, \tag{15}$$

$$x \in X. \tag{16}$$

Problem (11), (14)-(16) is a linear MOCOP (LMOCOP). Note that without limiting accuracy, it can be assumed that all LMOCOP criteria are maximization criteria.

If $l = 1$, LMOCOP is degenerates into a linear single-objective Euclidean combinatorial optimization problem LCOP:

$$f_1(x) = c_1^\top x \to extr,$$

subject to constraints (15), (16).

Among LMOCOP, we will focus on problems, where

$$\text{extr=max},\ X = E_{nk}(\tilde{A}) \tag{17}$$

i.e. on linear multiobjective permutation-based optimization problems further referred to as $Z(F, D)$, where X satisfies (17) and D is given by (13). A single-objective analogue of LMOCOP is denoted as $Z'(f, D)$.

An unconstrained problem $Z'(f, D)$ can be solved easily in polynomial time [30], whereas the presence of additional constraints, i.e., the transition to a constrained optimization problem, significantly complicates its solution.

In the next section, we present a formulation of an applied problem enabling a formalization as $Z(F, D)$, where $D \subset X$.

3. Problem statement

One of the fundamental principles of modern intelligent information system design is modularity. Modularity significantly improves the efficiency of managing various stages of the software life cycle, including the development, implementation, maintenance, and advancement of software and mathematical components for computer systems.

The modular approach involves creating software in the form of a set of separate interacting components called modules. Each module is designed to perform a specific function and can be developed, tested, and updated independently of the others. This architectural strategy facilitates parallel development and simplifies future modifications and updates to the system, ensuring its scalability and ease of maintenance.

A critical issue arises in modular software development, which is the selection and optimal configuration of modules at the design stage. In particular, this concerns the problem of optimal program composition when the overall software system consists of several modules, while several software implementation options are available for some modules. These options may differ in functionality, computational efficiency, memory consumption, or compatibility with hardware resources.

Formally, the problem of module selection can be viewed as a combinatorial optimization problem, the goal of which is to select the optimal subset and configuration of modules to optimize certain criteria, such as minimizing execution time, minimizing memory usage, maximizing reliability, or achieving a given level of economic efficiency, taking into account technical constraints.

3.1. Problem 1 statement and model

Let us consider the problem of ordering modules and selecting a way of their implementation when developing software. Specifically, this is the task of optimally assembling a program consisting of several modules, some of which can be implemented on a computer in different ways.

At the software development stage, the program can be represented as n separate interconnected blocks (modules, procedures, programs, segments). For each block j, there are a_j possible implementation options. Then a vector of variables $x = (x_1, ..., x_n)$ representing a plan for implemented options will be an element of a set $E_{nk}(\mathcal{A})$ of multipermutation e-configurations induced by a multiset $\mathcal{A} = \{a_1, ..., a_n\}$ having k distinct elements. Each block is characterized by the execution time $t_j(x_j)$, the amount of memory occupied $m_j(x_j)$, and the required total memory $w_j(x_j)$. The required total memory includes the total memory for the executable code, constants, arrays, and additional overhead. The goal is to select a variant for each program block such that the program terminates in a minimum time T, while not exceeding the allocated resources.

Let us formulate the problem as a mathematical model of combinatorial optimization with two minimization criteria.

Mathematical formulation of the problem: find a vector $x = (x_1, ..., x_n)$ such that

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

subject to the constraint $\sum_{j=1}^{n} v_j(x_j) + \max\limits_{1\leq j\leq n} w_j(x_j) \leq V$, where V is the amount of memory allocated for the optimizing program.

As can be seen in this problem, the search domain is a set, and the problem contains two objective functions and one constraint, i.e., overall it belongs to class MOCOP.

A linear version of it is formed if

$$\exists t_j, v_j, w_j > 0 : t_j(x_j) = t_j x_j, v_j(x_j) = v_j x_j, w_j(x_j) = w_j x_j, j \in J_n.$$

Then the objective and constraints becomes

$$\begin{gathered} T = \textstyle\sum_{j=1}^{n} t_j x_j \to \min, \\ M = \textstyle\sum_{j=1}^{n} v_j x_j \to \min, \\ \textstyle\sum_{j=1}^{n} v_j x_j + \max\limits_{1\leq j\leq n} w_j x_j \leq V. \\ x \in E_{nk}(\mathcal{A}). \end{gathered} \tag{18}$$

The constraint is still nonlinear but it is transformed into a linear one after introducing additional variables y for representing $\max\limits_{1\le j\le n} w_j x_j$. Respectively, denoting $y = \max\limits_{1\le j\le n} w_j x_j$, we can rewrite (18) as a collection of linear constraints:

$$\begin{gathered} \sum_{j=1}^{n} v_j x_j + y \le V, \\ w_j x_j \le y, j \in J_n. \end{gathered}$$

Now, we came to a MOCOP with combinatorial variable vector x and discrete variable y.

3.1.1. Example

Let us consider a program consisting of three modules that can be implemented in different ways. Let us denote these modules as A, B, and C. Module A allows two possible implementations, which are represented as permutations over the set $A' = \{0, 1\}$. Module B allows three possible implementations, described by permutations from the set $B' = \{0, 0, 1\}$. Similarly, module C allows two possible implementations, represented as permutations over the set A'. Thus, the overall configuration of the program can be described by a set of polypermutations, i.e., the Cartesian product of several sets of permutations.

For convenience, we represent the solution variables as a vector $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$, which uniquely encodes the selected procedures for all modules. For each x_j, the execution time t_j, memory usage, constants and arrays v_j, and the required total memory w_j are determined. Thus, this formulation corresponds to the mathematical model described above.

3.2. Problem 2 statement and model

Arrays of information in computer memory can be allocated across different levels of hierarchy, each containing one or more memory devices (MDs) of similar or distinct types with comparable speeds. The key parameters of each MD are its capacity and transfer speed. Typically, higher-level devices offer greater speed but smaller capacity than lower-level ones. The following model describes the optimal allocation of data arrays across memory devices to balance these characteristics effectively.

We have n memory devices ($MD_1, MD_2, \ldots, MD_n$) each characterized by its capacity C_i (MB), transfer speed v_i (MB/s), and user activity weight per MB r_i. There are n data arrays with sizes $a_1, a_2, \ldots, a_n$ (MB). Each array must be placed on exactly one memory device without splitting. The goal is to design an assignment plan minimizing total processing time and maximizing total user activity, subject to per-device capacity constraints.

Let us introduce necessary notations:

- $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ is a multiset of array sizes;
- k is number of distinct elements in $\mathcal{A}$;
- $E_{n,k}(\mathcal{A})$ is a set of all multi)permutation e-configurations induced by $\mathcal{A}$;
- $x = (x_1, \ldots, x_n) \in E_{n,k}(\mathcal{A})$ is a variable vector of sizes of assigned arrays, where x_i is size of array placed on MD_i;
- T is total processing time (seconds);
- R is total activity (activity units).

Since $1/v_i$ (s/MB) is a transfer time per unit of data on MD_i, Model 2 has the form of:

$$\begin{gathered} T = \sum_{i=1}^{n} \frac{x_i}{v_i} \to \min \\ R = \sum_{i=1}^{n} r_i x_i \to \max \end{gathered}$$

subject to:

$$x_i \le C_i, \quad i \in J_n,$$

$$x \in E_{nk}(\mathcal{A}).$$

This model is a linear bi-objective constrained Euclidean combinatorial optimization problem over the multipermutation set $E_{nk}(\mathcal{A})$, i.e. it is a LMOCOP.

Model 2 reflects practical IT planning trade-offs in hierarchical or distributed storage: reducing input/output latency while increasing activity-based performance within strict capacity limits. It applies to caching, in-memory databases, edge computing, and hybrid storage systems, providing a rigorous mathematical basis for forming Pareto-efficient data placement under constraints having wide applications in intelligent information systems.

4. Solution Techniques

For solving an LMOCOP, we propose first to apply methods of multi-criteria optimization, reducing it to a standard single-objective optimization and then solve the resulting problem using the combinatorial optimization method. Thus, we divide the solving process into two stages and propose a two-stage approach to solving LMOCOP.

Since Problem 1 and Problem 2 are problems of multi-objective optimization on e-configuration sets, when they are solved, it makes sense combining multi-criteria methods with combinatorial optimization approaches. Such a combination makes it possible obtaining a feasible e-configuration as a solution while ensuring effectiveness of this solution for each of multiple criteria.

Multi-criteria (multi-objective) optimization problems aims to simultaneously optimize several objectives that may be in conflict. There are several quite different approaches to solving such problemss [31, 32, 33].

The main multi-criteria optimization approaches can be divided into the following groups:

1. Methods of reduction to a single-criterion problem (convolution methods): all criteria are combined into one function by their linear weighting. Determining weighting coefficients for each criterion and reflecting their relative importance are necessary. The result is a one-criteria optimization problem.
2. The approximation method of all partial criteria to the ideal point. This method searches for solutions, each of which cannot be improved by one criterion without worsening another. Such a set of solutions is called Pareto-efficient. It allows you to explore all the compromise options between the criteria.
3. Heuristic and metaheuristic methods: these algorithms are used for the evolutionary search of optimal solutions. They search for several solutions simultaneously, making it possible to find a set of Pareto-optimal solutions efficiently.
4. Hierarchical optimization methods.
   a) Analytic Hierarchy Process (AHP): involves decomposition of the problem into hierarchical levels (goal, criteria, alternatives). The criteria are compared in pairs; based on this, a priority matrix is built to determine the best solution.
   b) Analytic Network Method (ANP): This is a generalization of the AHP method that considers the relationships between criteria at different levels of the hierarchy.

Since the introduced models include combinatorial constraints, in combination with a vector optimization method, it is necessary to apply a combinatorial optimization approach to find a feasible solution.

Therefore, below most common methods of combinatorial optimization are outlined:

1. Branch and Bound method involves dividing the problem into sub-problems (branches) and cutting off those that cannot lead to an optimal solution.
2. Dynamic programming is suitable for solving problems that can be broken down into interdependent, repeating subproblems.

3. Genetic algorithms are used to find a "good enough" solution in large search spaces where an exact search is too computationally complex. At the same time, such algorithms can also be useful for problems where it is necessary to optimize several criteria simultaneously or where the search space is very large.
4. Simulated Annealing is commonly applied to problems where the search space is large, while the method allows avoiding getting stuck in local minima.
5. Greedy Algorithms is based on achieving locally optimal solution at each step, while hoping hoping that, expectedly, these solution sequence converges to a globally optimal solution.

Based on the analysis of methods and properties of combinatorial configurations, and continuing research and developing the results of works [13, 16, 19], we will formulate a two-stage approach to solving LMOCOP, which is the formulated above practical problem of software design for information systems.

The proposed approach to solving the problem is based on the linear convolution (aggregation) of the partial criteria and further reduction of it's solution search to solving a series of single-objective combinatorial optimization problems. We present an approach to method for solving the single-objective problems, which is based on two continuous relaxation and cutting planes.

4.1. Combinatorial Optimization Algorithm

Input: a constraint linear permutation-based problem $Z\left(F,G\right)$ for a certain $G=E_{nk}(\tilde{A})$ and additional linear constraints are given by (15) (see (17)).

4.1.1. Multiobjective problem transformation

Let us reduce the multi-objective combinatorial optimization problem $Z\left(F,D\right)$ to a single-objective LCOP using linear convolution. For that, we set weighting coefficients $\lambda_j\in\mathbb{R}^1_{>0}, j\in J_l, \sum_{i=1}^{l}\lambda_i=1$, that express the degree of importance of each criterion and move to maximizing the linear combination of the objective functions. That is, we come to the problem:

$$Z\left(F,D\right)\rightarrow Z'\left(f,D\right)=\{f\left(x\right)=\sum_{i=1}^{l}\lambda_i c_i^{\top}x\rightarrow\max, Ax\leq b, x\in X\}.$$

4.1.2. Linear permutation-based optimization

Next, we solve the single-objective optimization problem $Z'\left(f,D\right)$.

First, we solve a continuous relaxation of the problem on the polytope $P=convD$. An issue with its solution by conventional linear programming methods is that the system of constraints (7), (8), generally, contains an exponential on n number of constraints, so all of them cannot be involved in the solving problem if n is quite large. Therefore, it is necessary to develop specific methods that use the specifics of the problem and properties of the objective function domain X and feasible domain D.

This paper proposes an iterative approach to solving $Z'\left(f,D\right)$ by solving a series of relaxation linear optimization problems on nested polyhedra. The approach is essentially a cutting-plain method utilizing the fact that the set $X=E_{nk}(\tilde{A})$ is vertex-located (VLS), i.e. $X=vert(convX)$ and is inscribed into a hypersphere S centered at $\overline{\mathbf{a}}=(\overline{a},...,\overline{a})$, where $\overline{a}=\frac{1}{n}\sum_{i=1}^{n}\tilde{a}_i$. Polyhedral relaxation of $Z'\left(f,D\right)$ is the problem $Z'\left(f,P\right)$, where $P=convD=\{x\in\Pi_{nk}(\tilde{A}):Ax\leq b\}$.

Theorem 2 underlies the Sequential Constraint Connection Method (SCCM) [25] of constrained linear optimization on the multipermutohedron, i.e. for solving $Z'\left(f,P\right)$. Briefly, at the initial iteration, the relaxed problem is solved on the superset of the feasible domain (7), (8), (15), $x\in\left[\tilde{a}_1,\tilde{a}_n\right]^n$. Then the constraints (9) of Theorem 2 are checked for the obtained point x. If they hold, the process terminates with the conclusion $x\in\Pi_{nk}\left(\tilde{A}\right)$, respectively, x is an optimal solution. If (9) holds, while (10) violates, the conclusion $x\notin\Pi_{nk}\left(\tilde{A}\right)$ is made. If (9) violates, the transition from x to a point y is made by rearranging x-coordinates such that $y_1 \leq \ldots \leq y_n$. If y satisfies (10), i.e., $\sum_{i=1}^{j} y_{n-i+1} \leq S_j, j \in J_{n-1}$, then the conclusion $x, y \in \Pi_{nk}\left(\tilde{A}\right)$ is made. Otherwise, the process continues iteratively by adding to the previous constraints one or all of the detected constraints that are violated at x (y) and solving the obtained linear optimization problem until conclusion $x, y \notin \Pi_{nk}\left(\tilde{A}\right)$ or $x, y \in \Pi_{nk}\left(\tilde{A}\right)$ according to Theorem 2.

Let x^* and y^* be the optimal solutions to the problems $Z'(f, D)$ and $Z'(f, D)$,respectively. The result of applying SCCM is y^*. If $y^* \in X$, then the problem $Z'(f, D)$ has been solved, namely, y^*. Otherwise, the cutting-plain method (CPM) is applied, aiming to cut y^* while leaving all points of D feasible. SCCM and CPM iteratively until we get a point of D as an output of SCCM.

The paper [34] proposes the Polyhedral-Surface Cutting Plane Method (PSCPM) of linear optimization over a vertex-located set. PSCPM is based on representing a VLS as an intersection of its convex hull and a strictly convex surface. Adapting PSCPM to $Z'(f, D)$, a VLS is D, its convex hull is a polytope P, while the hypersphere S is the strictly convex surface. Let us outline PSCPM:

- First, a polyhedral relaxation is solved on P that is $Z'(f, P)$, and its solution x is verified on belongingness to S.
- If it holds, the original problem $Z'(f, D)$ has been solved, and x is its optimal solution.
- Otherwise, a spherical relaxation $Z'(f, D)$ is considered, and a cut of x is formed utilizing a polyhedral cone with apex at x given by active P-constraints at the point and an intersection of its extreme rays with the hypersphere S.

This process continues iteratively until the termination condition is met.

Thus, this section presents a two-stage approach to solving the problem $Z(F, D)$ that combines the linear convolution method (stage 1) with PSCPM and SCCM.

5. Numerical example

An illustrative example of solving the following problem using the algorithm proposed above is presented.

Problem statement. Determine $x \in X$, where X is a set of permutations induced by a set $\tilde{A} = J_8 = \{1, 2, 3, 4, 5, 6, 7, 8\}$, such that:

$$\begin{aligned}
f_1(x) &= 6x_1 + 13x_2 + 15x_3 + 3x_4 + 10x_5 + 26x_6 + 29x_7 + 30x_8 \to \max, \\
f_2(x) &= 25x_1 + 13x_2 + 11x_3 + 6x_4 + 15x_5 + 31x_6 + 13x_7 + 32x_8 \to \max, \\
f_3(x) &= 10x_1 + 4x_2 + 19x_3 + 17x_4 + 11x_5 + 17x_6 + 12x_7 + 8x_8 \to \max,
\end{aligned}$$

subject to constraints:

$$\begin{aligned}
&5x_1 + 6x_2 + 8x_3 + x_4 + 16x_5 + 9x_6 + 7x_7 + 11x_8 \leq 225, \\
&x_1 + 12x_2 + 11x_3 + 17x_4 + 3x_5 + 19x_6 + 45x_7 + 45x_8 \leq 487.
\end{aligned} \tag{19}$$

Clearly, this problem is LMOCOP since several objectives are present, and both objective and constraints are linear, while the objective function domain is a permutation configuration set. Let us first represent this problem as $Z(F, D)$:

- the number of objectives is $l = 3$;
- $n = \tilde{A}$,
- no repetitions in $\tilde{A}$, hence $k = n = 8$,
- $X = E_{88}(\tilde{A})$,
- the constraint matrix and free terms vector:

$$A = \begin{pmatrix} 5 & 6 & 8 & 1 & 16 & 9 & 7 & 11 \\ 1 & 12 & 11 & 17 & 3 & 19 & 45 & 45 \end{pmatrix}, b = \begin{pmatrix} 225 \\ 487 \end{pmatrix}.$$

Table 1
The optimal values of the individual objective functions

Function	Maximum Value
f_1	507
f_2	659
f_3	480

Table 2
Computational results

Weighting coefficients	Values x_{sol}	Values (f_1, f_2, f_3)	Values $(\Delta f_1, \Delta f_2, \Delta f_3)$
$\lambda_1 = 0,35;\ \lambda_2 = 0,3;\ \lambda_3 = 0,35$	$(7,4,6,8,1,5,2,3)$	(491,633,480)	(16,26,0)
$\lambda_1 = 0,2;\ \lambda_2 = 0,2;\ \lambda_3 = 0,6$	$(7,6,3,8,1,4,2,5)$	(507,695,430)	(0,0,50)
$\lambda_1 = 0,4;\ \lambda_2 = 0,3;\ \lambda_3 = 0,3$	$(7,6,4,8,1,3,2,5)$	(497,639,432)	(10,20,48)

- the feasible domain $D = \{x \in X : x \text{ satisfies } (19)\}$.

Solving process outline At initial step, the vector of the ideal solution is constructed. For this purpose, the values of the functions f_1, f_2 and f_3 are maximized over the feasible set D. For that, we applied the algorithm given in Sec. 4.1 and solve the problems $Z'(f_1, D) - Z'(f_3, D)$. In order to solve these scalar optimization problems, the simplex method is employed. Consequently, the following optimal solutions are obtained Table 1.

Thus, we have the vector of the ideal solution $f^* = (507,\ 659,\ 480)$. To solve the posed problem, it is necessary to perform a search for the Pareto-optimal solution. For this purpose, the aggregation method is employed, using different weighting coefficients for the objective functions and comparing the resulting outcomes.

Thus, we reduce the problem to a scalar combinatorial optimization problem. Weighting coefficients $\lambda_1,\ \lambda_2,\ \lambda_3$ were applied. A scalar function $F = \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3$ was obtained. For the computations, the simplex method is employed. The results of the calculations are presented in a table Table 2.

All three proposed solutions belong to the Pareto-optimal set. However, when comparing the second and third solutions, the improvement in function f_3 in the third solution is negligible, while the deterioration in the values of functions f_1, f_2 is relatively significant. Therefore, from the perspective of balancing the objectives, the second solution is the most evident choice. For the final decision, it is advisable to involve the decision maker, who can take into account the priorities and preferences among the criteria.

6. Bridging our models with ML and AI

The module selection problem (Model 1) and the program layout problem (Model 2) can be extended to highly important tasks such as selecting and configuring layers, activation functions, connections, and other neural network components under constraints such as inference time, accuracy, energy consumption, etc.

7. Conclusion

The study investigates the problem of selecting and arranging software modules in intelligent information systems, highlighting its formulation as a multiobjective combinatorial optimization problem. Two mathematical models were proposed, each reflecting a crucial stage of the software lifecycle, namely,

- optimal selection of program modules with execution time and memory constraints,
- optimal allocation of arrays in hierarchical memory structures.

These problems were formalized as linear optimization problems in terms of Euclidean combinatorial configurations, providing a unified mathematical framework for their analysis and solution.

The proposed two-stage approach, combining linear convolution of criteria with combinatorial optimization, enables a reduction of complex multi-objective optimization problems to a sequence of tractable single-objective continuous optimization problems, where surface and polyhedral relaxations are combined. The case study confirmed the ability of the method to identify Pareto-optimal solutions.

Overall, the results demonstrates applicability of combinatorial optimization methods to the domain of software engineering. Future work may focus on extending the approach to larger-scale systems and other applied domains including machine learning and artificial intelligence.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT and Grammarly for grammar and spelling checks, as well as for improving the wording of certain paragraphs.

References

[1] E. A. Bender, An Introduction to Mathematical Modeling, Dover Publications, Mineola, NY, 2000.

[2] D. Edwards, Guide to Mathematical Modelling, Industrial Press, Inc., New York, NY, 2007.

[3] T. Witelski, M. Bowen, Methods of Mathematical Modelling: Continuous Systems and Differential Equations, Springer Undergraduate Mathematics Series, Springer International Publishing, Cham, 2015. doi:10.1007/978-3-319-23042-9.

[4] M. Conforti, G. Cornuéjols, G. Zambelli, Integer Programming, Springer, Cham, 2014.

[5] X.-S. Yang, Mathematical modeling, in: Engineering Mathematics with Examples and Applications, Elsevier, 2017, pp. 325–340. doi:10.1016/B978-0-12-809730-4.00037-9.

[6] F. Papadimitriou, Geo-mathematical modelling of spatial-ecological complex systems: an evaluation, Geography, Environment, Sustainability 3 (2010) 67–80. doi:10.24057/2071-9388-2010-3-1-67-80.

[7] A. C. Fowler, Mathematical Models in the Applied Sciences, Cambridge University Press, Cambridge, 1998.

[8] M. Naghshvarianjahromi, S. Kumar, M. J. Deen, Natural intelligence as the brain of intelligent systems, Sensors 23 (2023) 2859. doi:10.3390/s23052859.

[9] M. Kaltdorf, T. Breitenbach, S. Karl, M. Fuchs, D. K. Kessie, E. Psota, M. Prelog, E. Sarukhanyan, R. Ebert, F. Jakob, G. Dandekar, M. Naseem, C. Liang, T. Dandekar, Software JimenaE allows efficient dynamic simulations of boolean networks, centrality and system state analysis, Scientific Reports 13 (2023) 1855. doi:10.1038/s41598-022-27098-7.

[10] M. Bona, Combinatorics Of Permutations, Chapman And Hall/Crc, 2020.

[11] P. M. Pardalos, D.-Z. Du, R. L. Graham (Eds.), Handbook of Combinatorial Optimization, 2nd ed., Springer, New York, 2013.

[12] S. V. Yakovlev, The method of artificial space dilation in problems of optimal packing of geometric objects, Cybernetics and Systems Analysis 53 (2017) 725–731. doi:10.1038/s41598-020-67842-9.

[13] S. Yakovlev, O. Pichugina, L. Koliechkina, Combinatorial point configurations and polytopes, Wydawnictwo Uniwersytetu Łódzkiego, Łódz, 2023. URL: https://www.press.uni.lodz.pl/wul/catalog/book/589. doi:10.18778/8331-391-7.

[14] S. V. Yakovlev, I. V. Grebennik, Localization of solutions of some problems of nonlinear integer optimization, Cybernetics and Systems Analysis 29 (1993) 727–734. doi:10.1007/BF01125802.

[15] L. Koliechkina, T. Hudz, V. Kylnyk, Modeling of bank performance indicators based on business intelligence and data analysis, in: 2023 IEEE 13th International Conference on Electronics and Information Technologies (ELIT), 2023, pp. 87–92. doi:10.1109/ELIT61488.2023.10310849.

[16] L. Koliechkina, O. Dvirna, S. Khovben, Multi-criteria decision discrete model for selecting software components in component-based development, in: Proceedings of the 3rd International Workshop of IT-professionals on Artificial Intelligence (ProfIT AI 2023), volume 3641 of *CEUR Workshop Proceedings*, CEUR, Waterloo, Canada, 2023, pp. 182–193. URL: https://ceur-ws.org/Vol-3641/paper16.pdf.

[17] Y. G. Stoyan, V. Z. Sokolovskii, S. V. Yakovlev, Method of balancing rotating discretely distributed masses, Energomashinostroenie; (USSR) 2 (1982) 4–5. URL: https://www.osti.gov/etdeweb/biblio/6490782.

[18] L. Koliechkina, O. Pichugina, A horizontal method of localizing values of a linear function in permutation-based optimization, in: H. A. Le Thi, H. M. Le, T. Pham Dinh (Eds.), Optimization of Complex Systems: Theory, Models, Algorithms and Applications, volume 991, Springer International Publishing, Cham, 2020, pp. 355–364. doi:10.1007/978-3-030-21803-4_36, series Title: Advances in Intelligent Systems and Computing.

[19] N. Semenova, L. Koliechkina, V. Koliechkin, Solving vector optimization problems on combinatorial configurations with fuzzily specified data, in: Selected Papers of the III International Scientific Symposium “Intelligent Solutions” (IntSol-2023), volume 3538, CEUR, Kyiv - Uzhhorod, Ukraine, 2023, pp. 257–266.

[20] Y. G. Stoyan, S. V. Yakovlev, O. A. Emets, O. A. Valuiskaya, Construction of convex continuations for functions defined on a hypersphere, Cybernetics and Systems Analysis 34 (1998) 176–184. doi:10.1007/BF02742066.

[21] B. Korte, J. Vygen, Combinatorial Optimization: Theory and Algorithms, 6th ed., Springer, New York, NY, 2018.

[22] C. Berge (Ed.), Principles of combinatorics, Academic Press, New York, 1971.

[23] S. V. Yakovlev, The theory of convex continuations of functions at the vertices of convex polygons, Computational mathematics and mathematical physics 34 (1994) 959–965.

[24] R. Rado, An inequality, Journal of the London Mathematical Society s1-27 (1952) 1–6. doi:10.1112/jlms/s1-27.1.1.

[25] Y. G. Stoyan, O. O. Yemets, Theory and methods of Euclidean combinatorial optimization, ISDO, Kyiv, 1993.

[26] A. Postnikov, Permutohedra, associahedra, and beyond, International Mathematics Research Notices 2009 (2009) 1026–1106. doi:10.1093/imrn/rnn153.

[27] A. Postnikov, V. Reiner, L. Williams, Faces of generalized permutohedra, Documenta Mathematica 13 (2008) 207–273. doi:10.4171/dm/248.

[28] J. C. Martínez Mori, S. Samaranayake, Permutatorial optimization via the permutahedron, Operations Research Letters 50 (2022) 441–445. doi:10.1016/j.orl.2022.06.008.

[29] M. X. Goemans, Smallest compact formulation for the permutahedron, Mathematical Programming 153 (2015) 5–11. doi:10.1007/s10107-014-0757-1.

[30] S. V. Yakovlev, Bounds on the minimum of convex functions on euclidean combinatorial sets, Cybernetics 25 (1989) 385–391. doi:10.1007/BF01069996.

[31] M. Toloo, S. Talatahari, I. Rahimi (Eds.), Multi-Objective Combinatorial Optimization Problems and Solution Methods, Academic Press, London, 2022. doi:10.1016/C2020-0-00431-7.

[32] M. Ehrgott, M. M. Wiecek, Mutiobjective Programming, volume 78, Springer New York, New York, NY, 2005, pp. 667–708. doi:10.1007/0-387-23081-5_17, series Title: International Series in Operations Research & Management Science.

[33] M. Ehrgott, Multicriteria Optimization, volume 491 of *Lecture Notes in Economics and Mathematical Systems*, Springer, Berlin, Heidelberg, 2000. doi:10.1007/978-3-662-22199-0.

[34] O. Pichugina, L. Koliechkina, N. Muravyova, The polyhedral-surface cutting plane method of optimization over a vertex-located set, in: N. Olenev, Y. Evtushenko, M. Khachay, V. Malkova (Eds.), Advances in Optimization and Applications, volume 1340 of *Communications in Computer and Information Science*, Springer International Publishing, Cham, 2020, pp. 84–98. doi:10.1007/978-3-030-65739-0_7.