Graph-based approach to the synthesis of an algorithm for parallel solving of interrelated organizational tasks using vertex separation*

Yuri Samokhvalov[1, †], Eduard Bovda[2,*, †], Viktor Klimenko[2, †] and Dmytro Ustynov[2, †]

[1] *Taras Shevchenko National University of Kyiv, Volodymyrska 60, Kyiv, 01033 Kyiv, Ukraine*
[2] *Kruty Heroes Military Institute of Telecommunications and Information Technology, Knyaziv Ostrozkyh Street 45/1, Kyiv, 01011, Ukraine*

Abstract
A mathematical model for synthesizing an algorithm for parallel solving of interconnected organizational problems is proposed. According to this model, the synthesis process consists of two stages. In the first stage, the main solution options for the entire set of problems are formed as graph structures. Also, for the main variants that use intermediate results of solving other problems, additional variants are formed based on the separation of the corresponding vertices. Then the expected execution time of the algorithms is calculated using the PERT method. In this case, the optimistic and pessimistic times are specified by fuzzy linguistic estimates, and the Golenko β-distribution is used to calculate the most probable time. In the second stage, the option with the minimum time expenditure is selected. For this purpose, a graph of primary and additional options is formed, and the selection problem is reduced to finding the shortest path in this graph.

Keywords
organizational task, graph, separation, algorithm, synthesis, variant, beta distribution, fuzzy intervals

1. Introduction

Organizational tasks represent a set of management and coordination activities that arise in the functioning of organizations and their structural units [1]. They cover activity planning, resource allocation, coordination of performers' work, monitoring of task execution, and integration of the obtained results into a unified managerial decision. A distinguishing feature of such tasks is their complexity and interdependence, which necessitates the application of methods for their efficient solution [2].

In general, the process of solving an organizational task includes the formulation of the initial task by a manager, its decomposition into subtasks, the distribution of subtasks among performers, their parallel solution, and the integration of results into a single decision [3]. This approach is characteristic both for the entire organization and its individual structural units.

A key aspect of this process is the parallel solution of subtasks. Independent subtasks can be solved simultaneously by different performers, whereas interrelated subtasks require coordination of results or adherence to a specific execution order. At the same time, the nature of the parallel solution process depends on the type of interdependencies involved. Subtasks may be linked either by their final results or by intermediate results.

Applied Information Systems and Technologies in the Digital Society (AISTDS-2025), October 01, 2025, Kyiv, Ukraine
* *Corresponding author.*
† These authors contributed equally.
yu1953@ukr.net (Y. Samokhvalov); edepig8305@ukr.net (E. Bovda); vklim55@gmail.com (V. Klimenko); ustynov.dmitriy@gmail.com (D. Ustynov)
0000-0001-5123-1288 (Y. Samokhvalov); 0000-0002-8267-2120 (E. Bovda); 0000-0001-7834-7213 (V. Klimenko); 0009-0004-3993-1096 (D. Ustynov)

To formalize the process of parallel subtask solving, graph models are widely used [4, 5]. At present, there exist many approaches (methods) to distributed/parallel graph computation, which can be grouped as follows:

- Parallelism and load balancing. This group of methods focuses on the efficient allocation of computational resources among system nodes and reducing workload imbalance. The foundation lies in graph partitioning algorithms that ensure an even distribution of vertices and edges across processes while minimizing interprocessor communications [6, 7].
- Parallel libraries and data structures. Methods of this group aim to simplify the development of parallel graph algorithms by providing ready-to-use high-level abstractions and efficient data structures. The core idea is to supply developers with libraries that conceal the low-level details of parallelism (task distribution, load balancing, synchronization) while ensuring high performance [8–10].
- Distributed algorithms. These algorithms are designed for performing graph computations in distributed environments, where graphs are too large for a single node or require processing at the scale of clusters. They provide graph partitioning across nodes, result synchronization, minimization of communication overhead, and workload balancing [11–13].

In addition, one can note the vertex separation method [4, 14], which consists in partitioning the set of graph vertices into subsets in such a way that the internal connections within each group are preserved, while the intergroup connections are minimized or eliminated. This allows parallel task solving within each group using resources of different structural units or computing systems. The dependencies between groups then determine the order of combining partial solutions into an overall result.

Summarizing, the mentioned groups of methods are characterized by different strategies of scaling graph computations, which are mainly focused on solving problems linked by final results, while tasks connected by intermediate results have received comparatively little scholarly attention.

This article proposes a mathematical model for one possible approach to parallel solutions for interrelated organizational problems, which may be linked by both final and intermediate results. This approach also further develops the ideas behind the vertex separation method in graph models, which, overall, will reduce the time it takes of receiving decisions and create the preconditions for improving the efficiency of management processes.

2. Problem statement of algorithm synthesis for solving organizational tasks

One of the characteristic features of organizational tasks is their informational interdependence at the level of both final and intermediate results of their solution. Based on this, the essence of the problem of determining an algorithm for solving the tasks of an organizational structure lies in selecting such an algorithm for solving the entire set of tasks, taking into account the interrelations between them, so that the overall execution time of their solution is minimized.

At present, the most common methods for solving problems of this class are graph theory methods. In the terminology of this theory, the given problem can be formulated as follows.

Let $S=\{s_i | i=\overline{1,N}\}$ be the set of tasks of the organizational structure, $E_i=\{e_{ij} | j=\overline{1,n_i}\}$ – the set of basic solution variants for the i-th task, and $A=\|\alpha_{ii'jj'}\|$ – the information dependency matrix between the solution variants of different tasks ($i \neq i'$). Here, a basic algorithm is defined as a standard procedure independent of the solutions of other tasks.

Let each set E_i be represented by a directed graph $G_i(X_i, Y_i)$), where $X_i=\{x_{il} | l=\overline{1,n_i}\}$ is the set of vertices representing information-processing operations of the solution variants of task s_i, and $Y_i=\{y_{ilk} | k=\overline{1,m_i}\}$ is the set of edges that determine the execution order of these operations. Each variant e_{ij} corresponds to a path in graph G_i, which is associated with an execution time denoted as t_{ij}. The objective is to identify, in each graph G_i, such a path e_i that the total execution time T of their combination E=$(e_1, e_2, \dots, e_N)$ is minimized:

$$T{=}min\sum_{i=1}^{N} t_i,$$

where t_i is the execution time of the chosen variant of task e_i.

At the same time, a constraint is imposed on the choice of variants. If any variant of task s_i is connected with several variants of task $s_{i'}$, then when solving task i by this variant, exactly one of the variants of task i' associated with it must also be selected. That is,

$$\text{if } \alpha_{ii'jj'} \neq 0 \;\&\; e_{ij} \in E, \text{ then } e_{i'j'} \in E \qquad (1)$$

Thus, the solution of this problem is carried out in two stages: first, a graph structure of solution variants of tasks is constructed, taking into account their interdependencies, and then, in this graph structure, the shortest path is found, which determines the algorithm for solving the organizational tasks.

3. Construction of the graph structure

At the beginning, for each problem s_i, algorithms (options) for its solution are determined $e_{ij}(j{=}\overline{1,n})$. Such variants can be constructed efficiently using the method [15], which is a development of Glushkov's predictive graph method.

Next, using the (PERT) method [16], a network schedule for its implementation is constructed for each option. First, the execution time t_{ij} of variant e_{ij} is determined as:

$$t_{ij} = \sum_{k=1}^{n} t_e^{ijk}, \qquad (2)$$

where t_e^{ijk} is the expected execution time of the k -th operation, and n is the number of operations in variant e_{ij} .

The expected time is defined as:

$$t_e^{ijk} = \frac{t'_{ijk}+4t_{ijk}^{*}+t''_{ijk}}{6}, \qquad (3)$$

where t'_{ijk} is the optimistic estimate of the execution time, $\overset{*}{t}_{ijk}$ is the most probable estimate, and t''_{ijk} is the pessimistic estimate.

The optimistic estimate is the minimum execution time under the most favorable conditions, whereas the pessimistic estimate is the maximum execution time under the least favorable conditions. The most probable time corresponds to the median of the probability distribution (cumulative probability 0.5).

Practical experience with network planning demonstrates that it is generally difficult to reliably estimate the minimum (maximum) execution time of a task in conditions of uncertainty, and even more so to provide a reliable estimate of the most probable time. In order to improve the objectivity of such estimates, the following approaches are proposed.

3.1. Determination of minimum and maximum time

We will define the minimum and maximum time for completing the job using fuzzy statements of the type "the time for completing the job is approximately in the range from a to b," which are natural for a person under conditions of uncertainty. The above statement represents a fuzzy trapezoidal number (a, b, α, β),, where α and β are fuzziness coefficients that define the boundaries of the possible execution time interval, i.e., the minimum and maximum time.

The fuzziness coefficients are calculated approximately using the formulas [17]:

$$\alpha{=}a{-}k \cdot \frac{b(a)}{2}, \;\; \beta{=}b{+}k \cdot \frac{b(b)}{2}, \qquad (4)$$

where $k \approx 2.55$, and $b(a)$ and $b(b)$ are the distances between the transition points for numbers approximately equal to a and b. The error of the approximate calculation Δ<0.5, which is fully acceptable in practice.

To calculate the distances between transition points, the mechanism proposed in [18] is used. This mechanism is based on expert data, which allows, for numbers approximately equal to $N \in [1,99]$, to compute the distances between their transition points (see Table 1).

Table 1
Calculation of Distances Between Transition Points

Numeric x	Distance calculation formula $b(x)$ between the transition points for the number x
1, 2, 3, 4, 6, 7, 8, 9	0,46 x
10, 20, 30, 40, 60, 70, 80, 90	(0,357-0,00163 x) x
35, 45, 55, 65, 75, 85, 95	(0,213-0,00067 x) x
5	2,8
15	6,48
25	6,75
50	24
For other numbers	$\frac{1}{2}\left(b\left(\left[\frac{x}{10}\right] \cdot 10+5\right)+b\left(x-\left[\frac{x}{10}\right] \cdot 10\right)\right)$

Using this table, for numbers N>99, the distance between their transition points is calculated according to the following algorithm.

Let the least significant digit of the number N have order q. Let r_q be the least significant digit of this number, and r_{q+1} be its digit whose order is one higher than the order of r_q. Define classes of numbers M_d, $d \in \{0, 1, 2\}$, where $d=q \bmod 3$. Then:

1. If $N \in M_0$, then $b(N)=b(x) \cdot 10^{q-2}$, where $x=r_q \cdot 10$·and $b(x)$ is calculated from Table 1.
2. If $N \in M_1$, there are two possible cases:
   a) if $r_{q+1}=0$, then $(N)=b(x) \cdot 10^{q-1}$, where $x=r_q$;
   b) if $r_{q+1} \neq 0$, then $b(N)=b(x) \cdot 10^{q-1}$, where $x=r_{q+1} \cdot 10+r_q$.
3. If $N \in M_2$, there are also two possible cases:
   a) if $r_{q+1}=0$, then $x=r_q \cdot 10$; $b(N)=b(x) \cdot 10^{q-2}$;
   b) if $r_{q+1} \neq 0$, then $x=r_{q+1} \cdot 10+r_q$; $b(N)=b(x) \cdot 10^{q-1}$.

As a result, the value $b(N)$ is obtained.

Let's consider an example illustrating this approach.

Example. Let's represent the statement "the completion time for the k-th work is approximately in the range from 90 to 120 minutes" as a fuzzy trapezoidal number. As noted, a fuzzy trapezoidal number has the form (a, b, α, β), where in this case a=90, b=120, and α and β are the minimum and maximum work completion times, which are calculated using formulas (1). According to these formulas, we need to find b(90) and b(120). Since 90<99, b(90) is found from Table 1: b(90) = (0.357 - 0.00163 90) 90 = 19. For the number 120, the value of b(120) is calculated by the algorithm.

For the number 120 we have q=2, r_q=2, r_{q+1}=1, d=2mod3=2, i.e. its class is M_2. Then, according to point 3(b), we have x=12, b(120) = b(12)·10. According to the table:

$$b(12) = \frac{1}{2}\left(b\left(\left[\frac{12}{10}\right] \cdot 10+5\right)+b\left(12-\left[\frac{12}{10}\right] \cdot 10\right)\right) = \frac{1}{2}(6.48+0.92)=3.7, \text{ and } b(120) = 37.$$

Then, according to (1), α =65.8, β =167.2. As a result, the fuzzy trapezoidal number has the form (90,120, 65.8, 167.2). Then the optimistic and pessimistic estimates of the time to complete the k-th work are equal to 65.8, 167.2.

3.2. Determination of the most probable time

Let t_{ijk}=$[t'_{ijk}, t''_{ijk}]$ be the estimate of the execution time of the k -th operation of variant e_{ij}, where t'_{ijk}, t''_{ijk} is the optimistic and pessimistic time. Since any value within the given interval can be considered a possible execution time, a probability $P(t)$ is introduced that the operation will be completed by time t ($t'_{ijk} \leq t \leq t''_{ijk}$).

The most probable time $\overset{*}{t}_{ijk}$is then taken as the median of the probability $P(t)$ distribution over the given interval, i.e., $P(\overset{*}{t}_{ijk})$=0,5.

To describe random variables that are bounded within a finite interval, the beta distribution is typically used. Various stochastic network models have been developed [19–21]. Among them, the simplest is the model by D. Golenko [21]. This model requires specifying two estimates: an optimistic estimate (a) and a pessimistic estimate (b) of the task duration.

The Golenko model is based on the beta distribution over the interval $[a, b]$ with parameters α=2 and β=3, having the probability density function:

$$f(t|a,b)=\frac{12}{(b\text{-}a)^4}\cdot(t\text{-}a)(b\text{-}t)^2$$

and the cumulative distribution function:

$$F(t)=\frac{1}{B(2,3)}\int_0^x y\cdot(1\text{-}y)^2\,dy,$$

where $B(2,3)=\int_0^1 y\cdot(1\text{-}y)^2dy$ is the beta function, and $x=\frac{t\text{-}a}{b\text{-}a}$ is the scaled variable ($0 \leq x \leq 1$) [22]. Since $B(2,3)=\frac{1}{12}$, it follows that

$$F(t)=12\int_0^x y\cdot(1\text{-}y)^2\,dy.$$

Thus, the most probable execution time of an operation is

$$t=F^{\text{-}1}(0.5).$$

As a result, each variant e_{ij} is represented by a network graph G_{ij}, in which the vertices correspond to operations r_{ijk} with their execution times t_{ijk}, and the edges define the order of their execution. Figure 1 shows an example of a network schedule e_{ij} variant of solving task s_i.

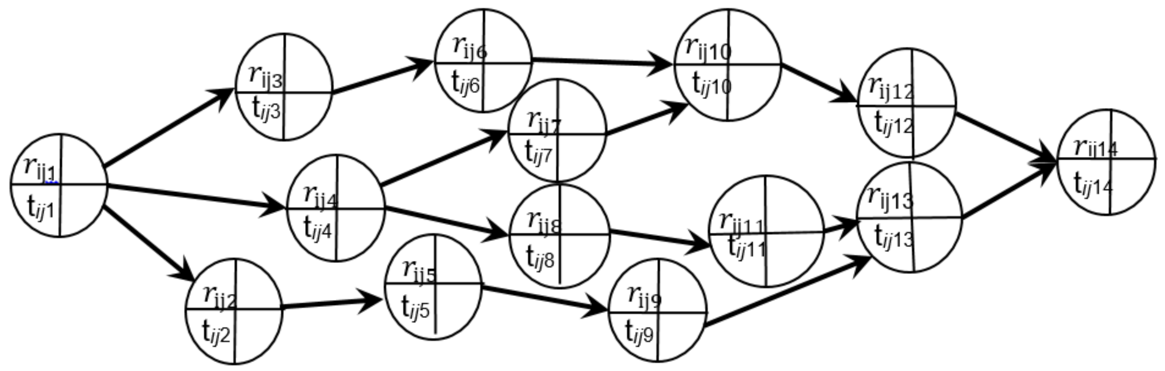

Figure 1. Network graph G_{ij} of variant e_{ij} for solving task s_i

In the constructed network graph G_{ij}, the execution time t_{ij} of variant e_{ij} is determined using the critical path method. The critical path is calculated from the initial vertex of the network to the terminal vertex using the formula t_{ijk}=$max t_{ijk}$+t_{ijl}}, where k is the current vertex r_{ijk}, and l is the set r_{ijl} of vertices that have a direct connection to r_{ijk}. The longest path e in the corresponding graph G_{ij} determines the execution time t_{ij} of the entire set of operations: t_{ij}=$max \sum_{k=1}^{n_e} t_{ijk}$, where n_e is the number of vertices in the path e.

As noted, the interdependence of tasks may occur at the level of final results as well as intermediate results of other tasks. Such results can be considered as alternative sources of information, and their use represents additional solution variants for the tasks. To account for the possibility of using intermediate results of task solutions s_i, an information linkage between the solute on variants e_{ij} is performed. For this purpose, each operation r_{ijk} is described by an aggregate:

$$W_{ijk}=r_{ijk}(V_{ijk}, t_{ijk}),$$

where $V_{ijk}=(v_1^{ijk}, v_2^{ijk}, \ldots, v_{n_{ij}}^{ijk})$ and $W_{ijk}=(w_1^{ijk}, w_2^{ijk}, \ldots, w_{m_{ijk}}^{ijk})$ are vectors of input and output data of the operation, the elements of which are represented by the corresponding information categories, and t_{ijk} is the execution time of the k -th operation.

Next, the input and output categories v_d^{ijk} and w_b^{ghl} $(i \neq g)$ are considered. If the names of these categories coincide and the execution of operation r_{ghl} completes earlier than operation r_{ijk}, i.e., t_{ghl}<t_{ijk}, then a connection is established from the source vertex r_{ghl} in the graph G_{gh} to the consumer vertex r_{ijk} in the graph G_{ij}.

The presence of such a connection can be considered as the possibility of reusing intermediate results from another task, i.e., as an alternative information source for operation r_{ijk}. This, in turn, leads to an additional solution variant $e_{ij'}$ for task s_i. The formation of this variant is based on the separation of the consumer vertex [22].

Let r_{ijk} be the consumer vertex in graph G_{ij}. The separation of this vertex is performed by introducing a new vertex r_{ijq} into the graph G_{ij}, which represents the operation of obtaining input data from an alternative information source. The new vertex r_{ijq} is then connected to the source vertex r_{ghl}, as well as to vertices r_{ijk} and to the vertex that immediately precedes the preparation of input data for r_{ijk} (see Fig. 2). Fig. 2 shows the separation of the consumer vertex r_{ij}.

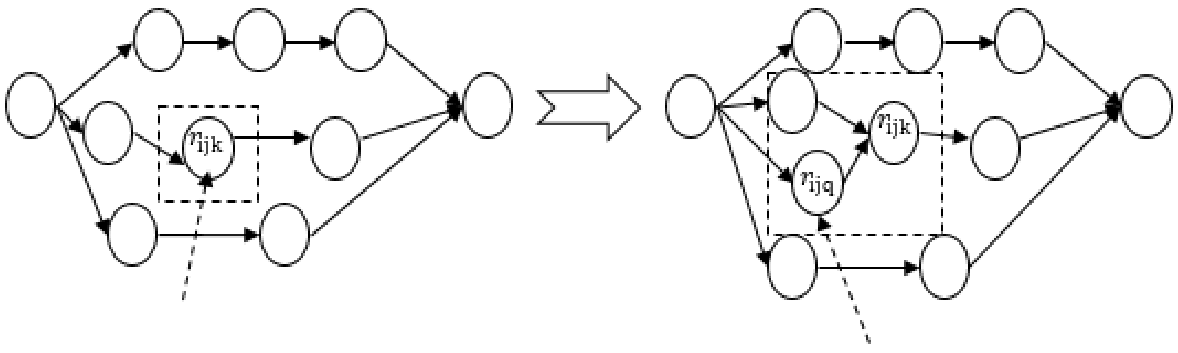

Figure 2. Separation of the consumer vertex r_{ij}

As a result of linking task variants and separating consumer vertices, additional solution variants $e_{ij'}$ for the task s_i are formed, for which the execution times are also determined. As a result, the main and additional options for solving organizational problems are represented by a set of interconnected graphs of their solution (Fig. 3).

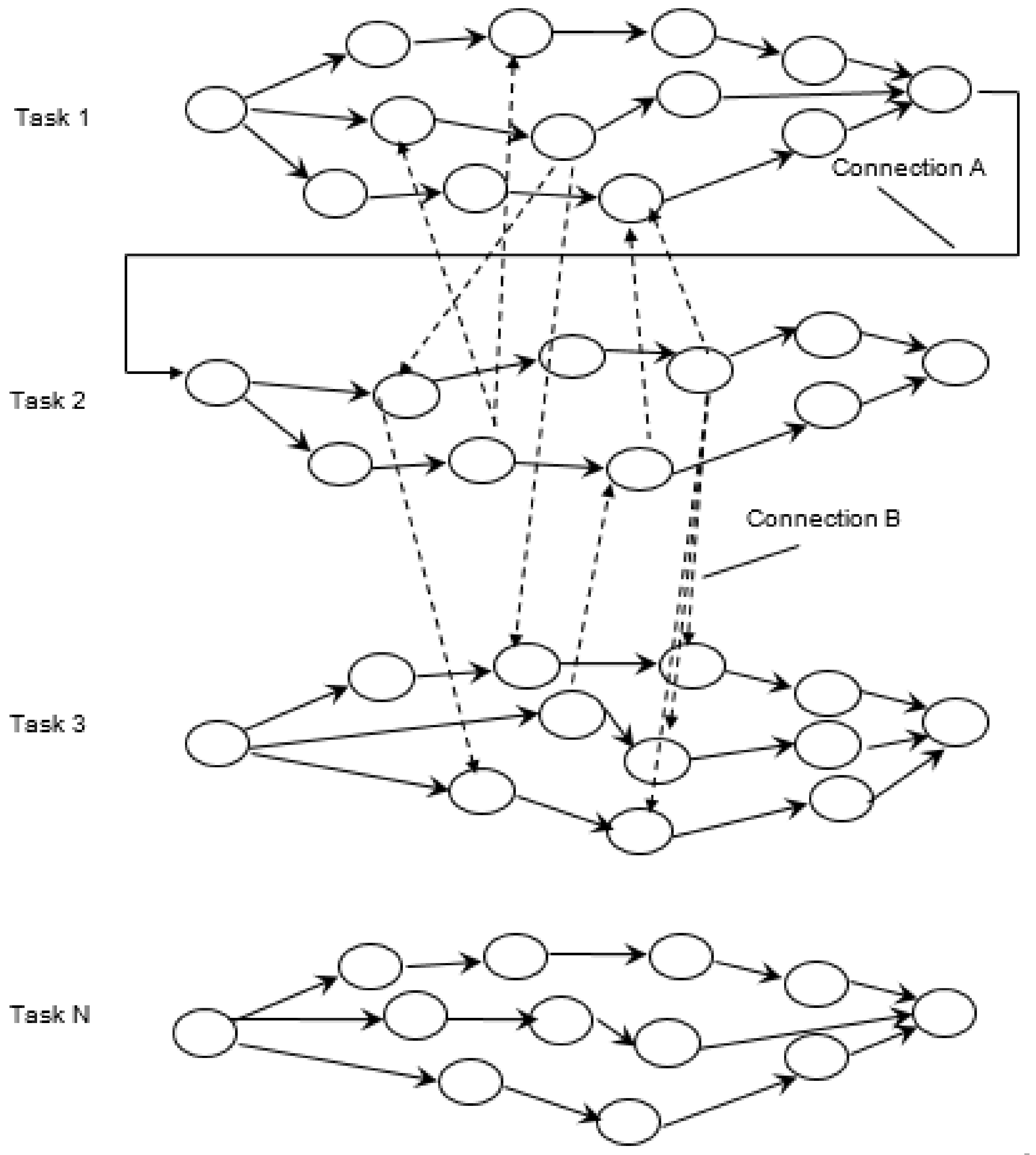

Figure 3. Graph structure of task solution algorithms

In the figure, the connections between variants based on the use of final results are labeled as "A," while those based on the use of intermediate results are labeled as "B."

3.3. Finding the shortest path in the graph structure

In the given graph structure, finding the shortest path using conventional algorithms is not directly possible. Therefore, the structure is first transformed into a standard form graph G. The construction of such a graph is carried out as follows [22].

Let E_i=$(e_{i1}, e_{i2}, \dots, e_{in_i})$ be the vector of solution variants for task i.

Step 1. The vectors E_i^T are ordered according to the task numbers. The elements of the vectors E_i^T correspond to the vertices of the graph G. Fig. 4 shows an example of vector ordering.

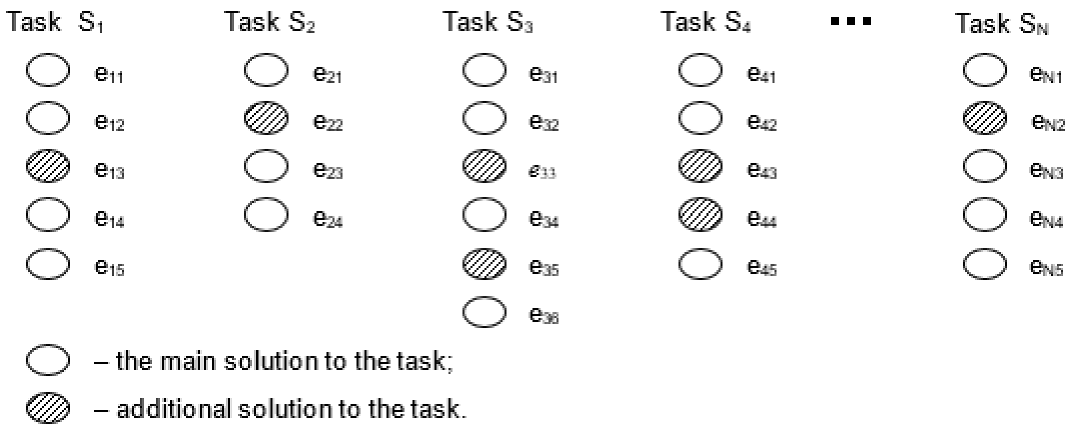

Figure 4. Arrangement of task solution variants

Step 2. According to the graph structure (see Fig. 3), the connectivity matrices $M_{ij}=\|m_{gh}\|$ of the solution variants e_{ig} and e_{jh} for tasks s_i and s_j (including unconnected tasks) are constructed. Next, adjacent tasks s_i and s_{i+1} $(i=\overline{1,N\text{-}1})$ are considered, and based on the corresponding matrices $M_{i(i+1)}$, connections between their variants are established. Additionally, connections are created between all variants e_{ij} of task s_i and the main variants $e_{(i+1)k}$ of task s_{i+1}, as well as between unconnected additional variants $e_{(i+1)l'}$ of task s_{i+1} and all main variants e_{ij} of task s_i. An example of linking solution options for related problems is shown in Fig. 5.

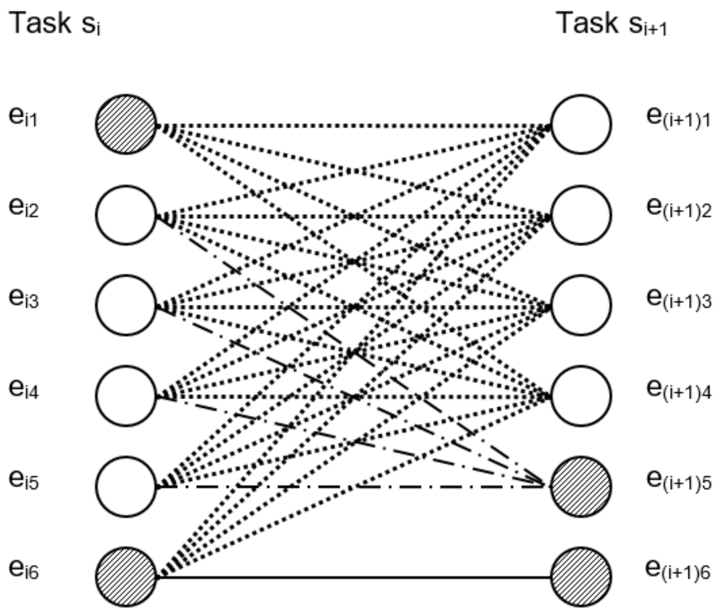

Figure 5. Linking solution variants of adjacent tasks

Step 3. The graph constructed in this way is supplemented with initial and terminal vertices, which are connected to all variants of tasks s_1 and s_N, respectively. Figure 6 shows an example of such a graph for three problems.

Step 4. Each edge u_{ijkl}, which connects variants e_{ij} and e_{kl}, is assigned a weight (corresponding to its execution time t_{ij} for the variant e_{ij}). Edges connecting the initial vertex H are assigned zero weight, whereas edges leading from variants e_{nj} to the terminal vertex K are assigned weights t_{nj}.

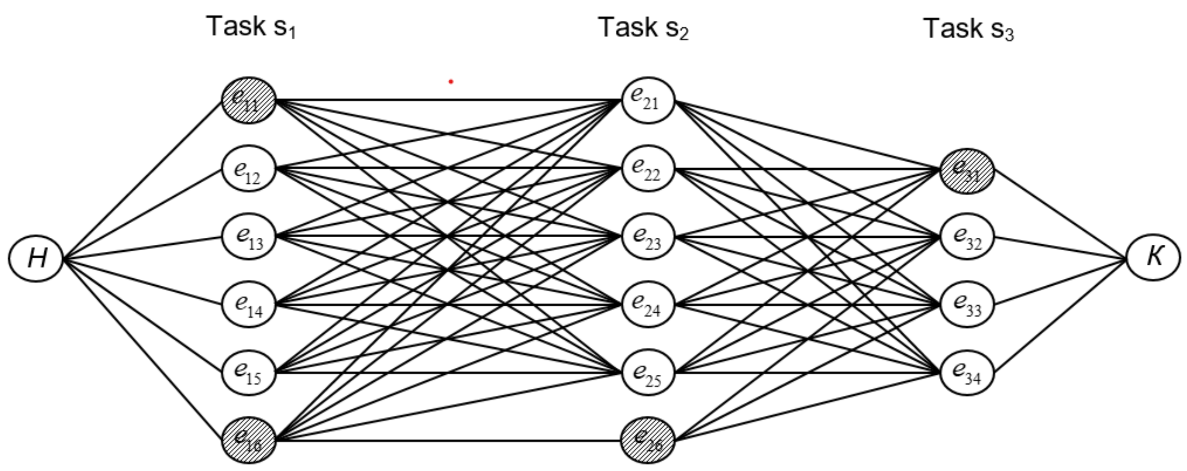

Figure 6. Graph of solution variants for three tasks

Since the graph G has a standard structure, the minimum path E={e_{ij}}, where i is the task number and j is the algorithm for solving it, can be found using known algorithms. This path will be an algorithm for solving the entire set of tasks of the organizational system, taking into account their interrelations. For example, in the graph in Figure 6, such a path could be (e_{11}, e_{25}, e_{33}). That is, task s_1 is solved by algorithm e_{11}, task s_2 by algorithm e_{25}, and task s_3 by algorithm e_{33}.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

Conclusions

A mathematical model for synthesizing an algorithm for parallel solution of interconnected organizational problems is considered. According to this model, the main solution options for the entire set of problems are first generated as graph structures. For the main options, which use intermediate results other problems, additional options are generated based on the separation of the corresponding vertex of the graph. The expected execution time for all algorithms is then calculated using the PERT method. Moreover, the optimistic and pessimistic completion times for each task are specified by fuzzy interval linguistic estimates, which are natural for humans under uncertainty. The Golenko β-distribution, which uses such estimates, is used to calculate the most probable time. The option with the minimum time expenditure is then selected. The problem of choosing such an option is equivalent to finding the shortest path in a graph. If a variant of one task is related to a variant of another task, both variants are selected. In general, the approach considered does not claim to be complete and can be used as a pilot for creating algorithms for optimal solutions to organizational problems, and the given examples of its main stages demonstrate the practical feasibility of the proposed model.

References

[1] R. L. Daft, Management (14th ed.). Cengage Learning, 2020.
[2] B. Liskov and P. L. G. Liskov, Principles of Computer System Design: An Engineering Approach (2nd ed.). The MIT Press, 2023.
[3] T. H. Davenport, S. W. M. Tomczyk, and M. L. Hammer, The AI-Powered Organization. Harvard Business Review Press, 2022.

[4] T. H. Davenport and J. Kirby, Designing Intelligent Organizations: AI, Analytics, and the Future of Work. Harper Business, 2023.

[5] J. A. Bondy and U. S. R. Murty, Graph Theory. Springer, 2008.

[6] S. Even, Graph Algorithms. Cambridge University Press, 2012.

[7] L. Meng, Z. Huang, Q. Zhang, L. Chen, and J. Yu, "A Survey of Distributed Graph Algorithms on Massive Graphs," *ACM Computing Surveys*, vol. 56, no. 4, 2024, doi: 10.1145/3659969.

[8] D. Q. Nguyen and K. Pingali, "Galois: A C++ Library for Parallel Graph Analytics," *SIGPLAN Notices*, 2018, doi: 10.1145/3276483.3276491.

[9] V. Kovalchuk, S. Ivanov, and M. Pavlenko, "Parallel Graph Processing in Multicore Systems: Data Structures and Libraries," *Information Technologies and Computer Engineering*, no. 2, pp. 45–53, 2023.

[10] O. Shevchenko and M. Tymoshenko, "GPU Libraries for Scalable Graph Algorithms in Scientific Research," *Scientific Works of KhNURE*, no. 58, pp. 112–120, 2022.

[11] G. Malewicz et al., "Pregel: A System for Large-Scale Graph Processing," in *Proc. SIGMOD*, 2010.

[12] J. E. Gonzalez et al., "PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs," in *Proc. OSDI*, 2012.

[13] R. Xiang, Y. He, Q. Sun, and J. Yu, "Adaptive Distributed BFS and PageRank for Skewed Graphs," in *Proc. IEEE BigData*, 2022.

[14] E. Turban, C. Pollard, and G. Wood, Information Technology for Management: On-Demand Strategies for Performance, Growth and Sustainability. Wiley, 2018.

[15] Y. Y. Samokhvalov, "Development of the Prediction Graph Method Under Incomplete and Inaccurate Expert Estimates," *Cybern Syst Anal*, vol. 54, pp. 75–82, 2018, doi: 10.1007/s10559-018-0008-1.

[16] "Program Evaluation and Review Technique (PERT)," Available: http://www.referenceforbusiness.com/encyclopedia/Per-Pro/Program-Evaluation-and-Review-Technique-PERT.html (Accessed: Nov. 15, 2025).

[17] Y. Samokhvalov, "Risk Assessment of Innovative Projects Based on Fuzzy Modeling," in *Lecture Notes in Computational Intelligence and Decision Making. ISDMCI 2020*, S. Babichev, V. Lytvynenko, W. Wójcik, and S. Vyshemyrskaya, Eds. Cham: Springer, vol. 1246, 2021, doi: 10.1007/978-3-030-54215-3_17. [

[18] A. Borisov, O. Krumberg, and I. Fedorov, Decision Making Based on Fuzzy Models: Examples of Use. Riga: Knowledge, 1994.

[19] L. Huang, "The Stochastic Network Model," in *Learning for Decision and Control in Stochastic Networks. Synthesis Lectures on Learning, Networks, and Algorithms*. Springer, Cham, 2023, doi: 10.1007/978-3-031-31597-8_2. [20] Ya. D. Gelrud, "Generalized Stochastic Network Models for Managing Complex Projects," *Bulletin of NSU*, vol. 10, no. 4, pp. 36–51, 2010.

[20] D. I. Golenko-Ginzburg, Stochastic Network Models for Planning and Management of Development. Nauchnaya Kniga, 2010.

[21] Y. Samokhvalov and O. Burba, Pre-Design of Automated Systems. Kyiv, 2013.