

DOI: 10.17323/2587-814X.2026.1.67.85

A review and comparison of newer methods for task allocation among performers

Timofey Yakovlevich Shevgunov^{a,b} 

E-mail: shevgunov@gmail.com

Anna Alexandrovna Kroshilina^b

E-mail: ankrosh@vk.com

^a Moscow Aviation Institute (National Research University), Moscow, Russia

^b Graduate School of Business, HSE University, Moscow, Russia

Abstract

This paper presents a description of the current state and the results of an analysis of recent advances in the problem domain of automated task distribution among employees. The purpose of the study is to identify the main trends and patterns in the development of existing task allocation methods, to determine their strengths and limitations, and to justify the need for new approaches and algorithms that can improve the efficiency of task delegation to employees. Using a unified system of notations for the key concepts of the subject area, the article provides a concise descriptive review of ten universal task distribution algorithms published over the past twenty years. The comparative analysis was carried out according

to a set of criteria reflecting both the technical and the organizational-behavioral aspects of how these algorithms function. The key evaluation criteria included: the degree to which performer competencies are taken into account; adaptability to changing external conditions and team composition; requirements for completeness and structure of the input data; robustness to incomplete or noisy data; transparency and explainability of decision-making; computational complexity; scalability with an increasing number of tasks and employees; implementation and maintenance costs; and orientation toward personnel development and competence enhancement. The comparative analysis we carried out made it possible to identify the advantages and shortcomings of each method and to formulate recommendations for their most effective practical application. The results showed that none of the examined algorithms can be considered a universal tool for delegation. Furthermore, it was found that comprehensive information about a performer's suitability for solving tasks requiring diverse competencies is either ignored or insufficiently utilized by many algorithms. This observation leaves open the problem of developing new approaches to task allocation and designing new algorithms based on them.

Keywords: task distribution, task allocation, performer assignment, assignment matrix, task delegation algorithms, round-robin algorithm, front based algorithm, genetic algorithm management, product digital twin, resource digital twin

Citation: Shevgunov, T. Ya., & Kroshilina, A. A. (2026). A review and comparison of newer methods for task allocation among performers. *Business Informatics*, 20(1), 67–85.
<https://doi.org/10.17323/2587-814X.2026.1.67.85>

Introduction

One of the factors determining the achievement of high productivity in the internal workflows of any organization, regardless of its organizational and legal form, is the effectiveness of planning and distributing tasks during the direct implementation of business processes. An increase in overall project complexity or operation under changing environmental conditions inevitably requires management, at the level of the organization or its structural units, to plan more accurately and allocate the available resources more rationally, which is one of the

important conditions ensuring rapid adaptation of a company's business processes. At the same time, the limited labor resources available to a company in the short term make it necessary to ensure their optimal utilization over as long working intervals as possible. In addition, there is a growing risk of erroneous decision-making [1] when tasks are distributed manually under workflow conditions close in intensity to the production capacity limits of the company. This risk further increases when tasks differ significantly in their characteristics, are rare or new, and are encountered by performers for the first time.

In organizing the task allocation process within organizations, three key problems are typically identified. The first problem consists in the uneven workload of employees, leading either to actual idle time or to overload of individual employees or their groups. The second problem lies in the absence of a systematic mechanism for taking employee competencies into account and verifying their compliance with the minimum requirements necessary for the effective execution of assigned tasks. The third problem manifests itself in managerial subjectivity during task allocation, which may reduce employee motivation and serve as a potential source of violations of business ethics or established corporate culture norms. One of the characteristic indicators that task allocation within an organization or its subdivision is ineffective is the systematic violation of task deadlines by performers, manifested in the fact that no less than one quarter of all assigned tasks are not completed on time for various reasons, regardless of which performers they were assigned to them. Another characteristic indicator is the systematic duplication of assignments, which may manifest itself either in assigning the same task to two independent performers or in incomplete delegation, when the execution of a task requires continuous personal involvement of a manager to monitor its timing or quality.

The analysis conducted on papers published from 2005 to 2025 aimed at identifying new and original methods of task allocation among performers has shown that, although this research area does not belong to the most prioritized ones, the development of new methods, algorithms, and delegation techniques continues to remain the subject of scientific and practical investigations carried out by individual research groups. The relevance of ongoing research in task allocation methods, the key direction of which is the development of universal algorithms and techniques, is driven by the need to form stable mechanisms for the operational redistribution of tasks capable of ensuring a balance between control and performer autonomy.

A detailed analysis of the sources made it possible to identify a number of new, noteworthy, but rather narrowly specialized solutions, for example those designed for cloud data centers [2], distributed computing systems [3], software developer support systems [4], or control systems for multi-purpose aeromobile systems [5]. Algorithms structurally similar to task allocation algorithms are also used in solving problems such as the distribution of research among academic schools [6], the selection of scientific and methodological information [7,8], and the allocation of credit application flows for commercial banks [9]. Particular attention should be paid to the analysis of changes introduced into established delegation procedures under conditions of mandatory use of remote management technologies [10]. It should also be noted that in foreign practice, certain solutions aimed at improving the efficiency of task allocation [11] may be regarded as results of intellectual property and receive legal protection in the form of patents.

However, the main attention of the authors in analyzing the publications was focused on identifying new universal methods that can be directly applied or quickly adapted for task allocation among performers in organizations of various industries and forms of ownership. This paper presents a review and comparative analysis of ten identified universal methods designed to automate task allocation among performers in companies or their subdivisions. The compiled list includes methods implementing substantially different ideas underlying delegation, while variants of the same algorithms that differ only slightly from their main versions were not included. In order to systematize the descriptions of various methods, a basic formalization of the notation for general factors was carried out in Section 1. Section 2 provides brief substantive descriptions sufficient for forming a general understanding of each of the ten algorithms under review. The formal notation used in the descriptions of some of them may slightly differ from that used in the original publica-

tions. Section 3 presents a comparative analysis of the selected algorithms according to a number of criteria, allowing their intrinsic characteristics to be compared and potential areas of application to be identified. The conclusion presents the findings of the current review and recommendations for further development of the analyzed domain.

1. Formalization of the task allocation problem

The subject area of task allocation among performers can be formalized using the following mathematical entities. Let

$$T = \{t_n, n = \overline{1, N}\}$$

denote a non-empty finite set of tasks that must be completed;

$$E = \{e_i, i = \overline{1, I}\}$$

denote a non-empty finite set of performers, or employees;

$$C = \{c_m, m = \overline{1, M}\}$$

denote a non-empty finite set of competencies required by performers from E to complete tasks from the set T .

Each task $t \in T$ is characterized by the following attributes: $C(t)$ is the set of competencies required for its execution; $p(t)$ is the priority of task t , expressed using an ordinal scale, for example, “low,” “medium,” “high,” or “critical”; $s(t)$ is the complexity of task t , expressed quantitatively in terms of labor intensity or the time required for its completion, for example, in hours.

Each performer $e \in E$ is characterized by the following attributes: $C(e)$ is the set of competencies possessed by the performer and applicable to tasks from the set T ; $a(e)$ is the available time for task execution, expressed in time units, for example, in hours; $w(e)$ is the current workload of performer e , expressed in time units.

Thus, the problem under consideration can be formulated as follows: it is required to distribute tasks from the set T among performers from the set E in such a way that both task characteristics (required competencies, priority, complexity) and performer characteristics (possessed competencies, available time, current workload) are taken into account. This allows the problem to be classified as an optimization problem in which it is necessary to minimize or maximize a certain objective function subject to a system of constraints. To formalize the assignment, it is convenient to introduce a binary function $x(t, e)$, which takes the value 1 if task t is assigned to performer e , and 0 otherwise.

Depending on managerial objectives, an organization may define different principles for distributing tasks among performers. Examples of key objectives include reducing the risk of personnel overload, ensuring a balanced distribution of workload among team members, or maximizing the utilization of available resources. In addition, in the strategic perspective, the development of employee competencies becomes important, enabling the formation of a more flexible and resilient organizational structure. Therefore, the choice of the optimality criterion is determined not only by current operational needs but also by long-term business priorities, such as operational efficiency, cost reduction, increased resilience to external risks, and the development of human capital. Below, four objective functions are considered, which most fully reflect these organizational goals.

1.1. Minimization of workload for employees with high current utilization

This criterion takes into account the available time for task execution for each employee $a(e)$, as well as task complexity $s(t)$ and task priority $p(t)$. Then the objective function F_1 has the following form:

$$F_1 = \sum_{e \in E} \sum_{t \in T} x(t, e) (a(e) - s(t)) \rightarrow \max \quad (1)$$

The optimization process consists in searching for such values of the binary function $x(t, e)$ that ensure the assignment of tasks to those employees who will retain the largest amount of free time after the assignment of the distributed task. Thus, the task is assigned to the less loaded employee, which leads to minimization of the workload of already highly utilized performers. Such an approach allows work to be redistributed in favor of employees with lower current utilization, thereby reducing the risk of burnout and overload. This approach is widely applied in organizations where it is important to maintain stable team performance, for example in call centers or IT support services, where balanced distribution of incoming requests reduces the probability of service failures.

1.2. Workload balancing among performers

The task consists in minimizing disproportions in the distribution of task labor intensity among employees. The average value across performers is calculated as

$$\bar{a} = \frac{1}{\#(E)} \sum_{e \in E} a(e), \quad (2)$$

which is then sequentially compared with each value $a(e)$. The difference between these values should tend toward a minimum:

$$F_2 = \sum_{e \in E} |a(e) - \bar{a}| \rightarrow \min. \quad (3)$$

Such a criterion ensures a more uniform distribution of tasks, which is particularly important in teams

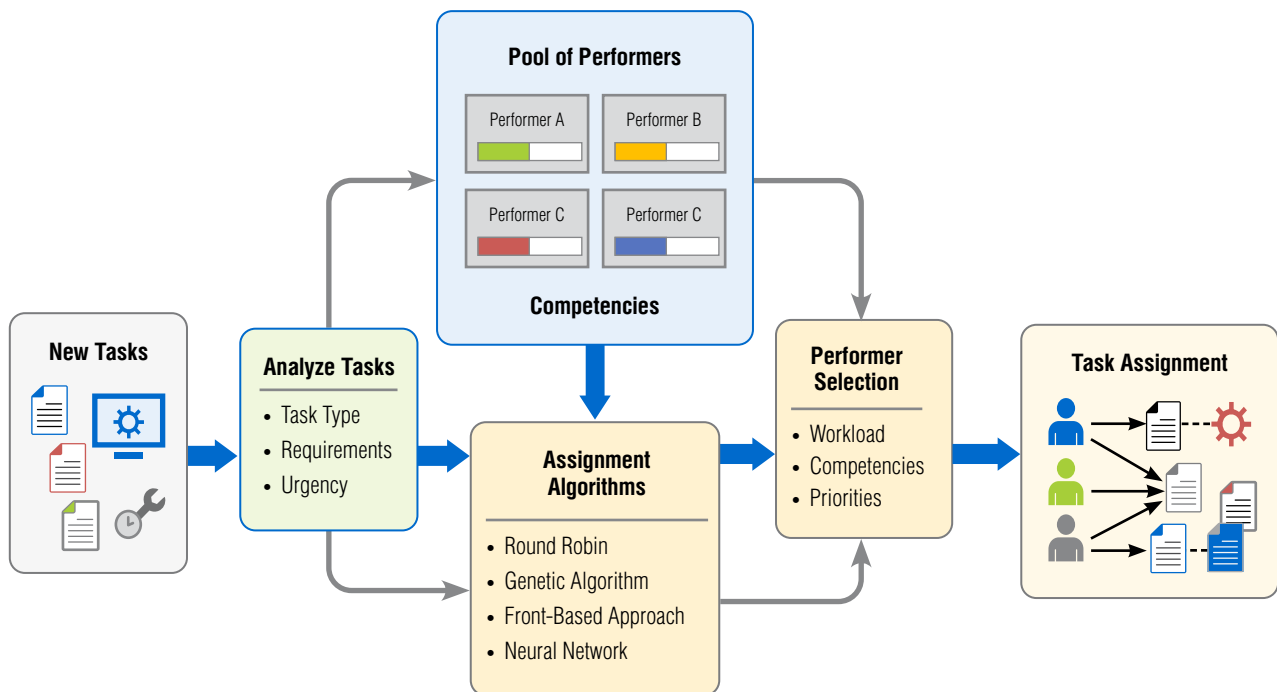


Fig. 2. Block diagram of the task allocation process among performers

where interchangeability of performers is assumed. An example is the distribution of work assignments within a software development team or among consultants of a project group, where excessive concentration of tasks on individual employees may create risks of deadline violations.

Figure 1 presents a generalized block diagram of a typical process of task allocation among performers in an organizational system operating under conditions of a non-stationary task flow and a heterogeneous pool of performers. The process begins with the arrival of new tasks forming a dynamic input flow T . For each task, a preliminary analysis stage is performed, including the determination of the task type, the required competencies $C(t)$, the priority level $p(t)$, and the urgency of execution. At this stage, a set of formalized characteristics is formed, which are subsequently used in the algorithmic task allocation procedure.

In parallel with the task pool, the performer pool E is considered, characterized by individual sets of competencies $C(e)$, current workload $w(e)$, available resource $a(e)$, and possible constraints. An important feature of the model being considered is the presence of uneven workload among performers, which is reflected in the “Performer Pool” block and constitutes one of the key factors in making assignment decisions.

At the next stage, a method or algorithm for task allocation is selected. Depending on organizational objectives and the structure of the input data, various approaches may be applied: cyclic distribution, combinatorial optimization methods, evolutionary algorithms, front based methods, neural network based correctors, and others. Thus, the “Allocation Algorithm” block reflects the variability of possible task flow processing scenarios. Subsequently, a performer is selected for a specific task. The decision is made taking into account a combination of factors: current workload, correspondence between competencies and task requirements, task priority, and strategic

objectives, such as workload balancing or employee competency development.

After the task is assigned, the system state parameters are updated, reflecting changes in the workload of each performer. This ensures the relevance of data under conditions of a non-stationary task flow, when new tasks arrive before previously assigned tasks have been completed. Thus, the process presented in the *Fig. 1* has an iterative nature and implements a closed-loop task allocation management mechanism, ensuring adaptation to changes in the composition of tasks, the state of performers, and the selected allocation strategy.

The typical task allocation scheme presented in *Fig. 1* has an inter-industry character and finds application in a wide range of organizational systems. Below are examples from various areas of activity demonstrating the characteristics of the set of tasks T , the set of performers E , the relevant constraints, and the target performance indicators.

In the banking sector, one of the typical allocation tasks is the processing of credit applications from legal entities and individuals. In this case, the set T includes applications of varying levels of complexity and risk, while the set E consists of employees of underwriting, risk analysis, and compliance departments. The constraints include regulatory requirements, internal bank regulations, permissible application processing times, and workload limits for specialists. The target business indicators include the average processing time of an application, the proportion of approved applications with an acceptable risk level, and the throughput of the unit. Similar problems are discussed in studies devoted to the optimization of banking business processes and operational risk management [12].

In medical institutions, task allocation is associated with assigning patients to physicians of appropriate specialization, planning diagnostic procedures,

and managing medical staff schedules [13]. In this case, the set T includes consultations, surgeries, and diagnostic examinations, while the set E consists of medical personnel with various competencies $C(e)$. The constraints include waiting time standards, urgency of medical cases, equipment availability, and shift based work schedules. The target business indicators include the total flow of treated patients, average waiting time, and balanced workload distribution among physicians.

In the construction industry, task allocation arises in the planning and coordination of work at construction sites [14]. The set of tasks T may include preparatory work, installation of structures, engineering works, and quality control activities, while the set E consists of specialized teams and subcontracting organizations. The constraints include technological sequencing of work, material delivery deadlines, safety requirements, and fixed project completion dates. Target indicators include adherence to the project schedule, minimization of downtime, and reduction of project costs.

In the judicial system, the distribution of cases among judges represents an assignment problem taking into account specialization, case complexity, and the current workload of the judicial staff [15]. In this case, the set T consists of cases of various categories, while the elements of E are judges with different qualifications and specializations. The constraints include procedural deadlines, requirements for balanced workload distribution, and principles of random case assignment established by regulations. The key target indicators include the average case processing time, compliance with procedural deadlines, and workload balancing among judges.

In service organizations, task allocation is associated with queue management, distribution of service requests, and planning of equipment usage [16]. The set T is formed from incoming requests or applications, while the set E consists of specialists of various

profiles. The constraints include service level agreements (SLA), task priority level, geographical distance, and personnel availability. The target indicators include average service time, the percentage of requests completed within the prescribed time, and the personnel workload coefficient.

The examples above demonstrate that the formal model of task allocation based on the sets T and E , the competency functions $C(t)$, $C(e)$, and the binary assignment function $x(t, e)$, is sufficiently universal and can be applied across a wide range of industries. At the same time, specific constraints and objective functions may vary significantly, which justifies further specialization of task allocation algorithms aimed at meeting industry-specific requirements and accounting for the characteristics of incoming task flows.

2. Task allocation algorithms

Currently, there exists a wide variety of approaches to task allocation among performers, including both strict mathematical optimization methods and heuristic algorithms, as well as combined schemes. The choice of a particular approach is determined by the nature of the problem, the dimensionality of the input data, and the requirements for the speed of obtaining a solution. Such methods are widely applied in project management, logistics, service companies, and other domains where rational use of resources and efficient execution of work are required. This section provides a brief review of ten universal task allocation algorithms published over the past twenty years.

2.1. Round-robin algorithm for uniform task distribution among performers

The Round-robin algorithm [17] is one of the typical examples of uniform task distribution, which can

be described as the sequential assignment of tasks to performers in a cyclic order. It is assumed that all performers are equal to one another, that is, there are no preferences in favor of any performer based on priority, qualification, or strict rules assigning specific types of tasks to particular individuals, and that all tasks have identical complexity and do not contain formally distinguished subtasks. The objective of this algorithm is to distribute tasks in such a way that no performer experiences overload.

Formally, the algorithm can be described as follows. Let, at a given moment, a set of tasks T be subject to allocation, and let the priority $p(t)$ of each task $t \in T$ be known. For each performer e from the set of performers eligible and available to execute such tasks, E , the current workload $w(e)$ is known.

The tasks from the set T are ranked according to their priority, forming a list T_p , which determines the order of allocation: tasks with a higher value of $p(t)$ are given priority in assignment. For each task t from T_p , a set of candidate performers $E_c(t)$ is formed. In order for a performer e to be included $E_c(t)$, the performer must have sufficient free time to complete task t , taking into account their current workload $w(e)$ and available time $a(e)$:

$$w(e) + s(t) \leq a(e). \tag{4}$$

The selection of a specific performer e for task t may be carried out based on additional criteria, such as minimal workload $w(e)$ at the moment of assignment or achieving a distribution of workload that is close to uniform upon completion of the allocation of all tasks from T . After assigning task t to performer e , their workload is updated as follows:

$$w(e) \leftarrow w(e) + s(t). \tag{5}$$

The cycle is repeated until all tasks from the list T_p have been allocated.

2.2. Algorithm based on the assignment matrix

The mathematical model for optimizing the process of task and labor resource distribution within an enterprise, proposed in [18] using combinatorial optimization methods, formed the basis of an algorithm for task allocation among employees based on constraint analysis and the branch-and-bound method.

The algorithm is based on constructing an assignment matrix, the processing of which results in the selection of a performer e from the set of candidate performers E for each task t from the set of tasks to be allocated T . For each task, along with the priority $P(t)$, expressed using a three-level ordinal scale (low, medium, high), a time constraint is introduced, described by the start date $D_{start}(t)$ and the end date $D_{end}(t)$. Each performer e is characterized by an integral indicator $Q(e)$, as well as the dates $V_{start}(e)$ and $V_{end}(e)$ of the period during which the performer is unavailable. The integral indicator is calculated as a linear combination:

$$Q(e) = w_c Q_c(e) + w_q Q_q(e) + w_t Q_t(e), \tag{6}$$

where $Q_c(e)$ and $Q_q(e)$ represent, respectively, assessments of the volume and quality of completed work;

$Q_t(e)$ represents an assessment of deadline compliance; w_c , w_q , and w_t are weight coefficients, which in [18] are proposed to be equal to 0.35, 0.4, and 0.25, respectively.

When calculating all performance indicators of performers, the algorithm initially introduces three levels: a baseline level from which performance is measured, a normal level that must be achieved, and a target level toward which the performer should strive.

To construct the assignment matrix and determine the optimal solution, an integer programming algorithm such as the branch-and-bound method [18] is used. The matrix is initialized with cost values corresponding to tasks. In each row of the assignment ma-

trix, the minimum element is identified and subtracted from each element of that row, resulting in at least one zero element appearing in the row. Then, in each column of the matrix, the minimum element is identified and, provided that there is no zero in that column, it is subtracted from the column elements.

A pair (t, e) , representing a branching candidate, is selected among those for which the matrix element value equals zero. A coefficient is calculated by summing the minimum value of the corresponding task row and the minimum value of the corresponding performer column. Among all such coefficients, the maximum one is selected, which determines the optimal decision: the task corresponding to the current row is assigned to the performer corresponding to the current column. Since each performer may be assigned only one task, the column of the assigned performer and the row of the assigned task are removed from the assignment matrix.

The algorithm sequentially analyzes subsets of performers, determines the optimal solution for each subset, and excludes it from further consideration. As a result, the best allocation of tasks among all possible combinations of performers is determined.

2.3. Algorithm for optimal workload distribution considering specialization and available time based on an adapted genetic algorithm

In [19], a composite genetic algorithm (GA) was proposed for solving the problem of academic workload distribution. The input data included a set of teachers, which is the set of performers E , and a set of disciplines and types of classes, which is the set of tasks T . In addition, the following characteristics were introduced: $K(e)$ is the qualification of performer e , i.e., the set of disciplines that the teacher is able to teach; and $R(t, e)$ is the relevance of task t to performer e , expressed as a value in the numerical interval $[0, 1]$, rep-

resenting the degree of correspondence between the performer's qualification and the task content.

The GA includes the following operations: generation of an initial population as a random distribution of workload hours; selection as choosing performers based on specialization and workload criteria; generation of new allocation variants through crossover and mutation; and identification of the current optimal workload distribution subject to constraints.

The algorithm begins by generating an initial population of solutions, which is allocation variants $P_0 = \{p_q^{(0)}, q = \overline{1, Q}\}$, where each variant $p_q^{(0)}$ represents a mapping of the set of tasks T onto the set of performers E , described by binary variables $x(t, e)$, indicating which performer e is assigned to task t . Assignments are generated randomly but in such a way that the constraints are satisfied. The first constraint requires that task t may be assigned to performer e only if $t \in K(e)$. The second constraint requires that, for each performer $e \in E$, the total volume of assigned tasks remains within an admissible workload interval.

For each allocation variant p_q , a fitness function $F(p_q)$ is calculated. It consists of several components. The component $F_1(p_q)$ evaluates the degree of correspondence $R(t, e)$ for all pairs (t, e) . The component $F_2(p_q)$ evaluates the uniformity of workload distribution among performers. The component $F_3(p_k)$ evaluates, for all performers, the integral proximity of workload $W(e)$ to the admissible interval $[A_{\min}(e), A_{\max}(e)]$. Based on the values of $F(p_k)$, a set of the most fit allocation variants P_{sel} is formed.

After evaluation, evolutionary steps are performed. A new generation is formed from the best variants by applying crossover and mutation operators. From the set P_{sel} , pairs of individuals p_a and p_b are randomly selected, to which the GA crossover operator is applied, implementing a random exchange of parts of the task allocation or subsets of pairs (t, e) between the two in-

dividuals. The resulting offspring are included in an intermediate generation P_{next} . For some individuals $p \in P_{next}$, a mutation operator is applied, consisting of randomly reassigning certain tasks $t \in T$ to other admissible performers $e \in E$ such that $t \in K(e)$.

The new generation is formed by combining the best individuals from the current generation with the new individuals of the intermediate generation: $P_{new} = P_{sel} \cup P_{nex}$. The algorithm iterations continue until a stopping criterion is reached, defined as the condition that 90% of the individuals in the current population have the same maximum fitness value. As a result of multiple iterations, an assignment matrix $X = \{x(t, e)\}$ is obtained that satisfies all constraints and optimizes the selected criteria. In cases where the number of optimized parameters exceeds fifty, multi-agent genetic algorithms (MAGAMO) [20] may be used to reduce solution time, enabling efficient large-scale multi-objective optimization.

2.4. Heuristic algorithm for optimal allocation

The heuristic algorithm for optimal distribution of objects among storage units proposed in [21] has polynomial computational complexity and can be applied to solving problems in various domains, such as distribution of parallel big data processing flows, warehouse logistics, and automated scheduling. It may also be adapted to the problem of task allocation among performers. This is a heuristic greedy algorithm in which the allocation decision at each step is made without consideration of long-term consequences. The algorithm aims at uniform filling of storage units and minimization of the difference between the most and least loaded units. In [21], objective functions, formalization of the problem, and results of experimental studies evaluating the algorithm's effectiveness are also presented.

Tasks $t \in T$ are sorted, forming a list ordered in descending order of their labor intensity $s(t)$. Task allocation

is performed iteratively until the list is exhausted. At each iteration, two tasks are assigned to performers: the most labor-intensive and the least labor-intensive tasks from T . The index of the candidate performer is determined as the remainder of the division of the task's position in the list by the number of performers. When adding task t to performer e , the following condition must be satisfied:

$$w(e) + s(t) > a(e), \quad (7)$$

otherwise, the task is not assigned to the given performer and is instead considered for the next candidate performer. After assigning a task to a performer, their workload is updated according to formula (5), and the assigned tasks are removed from the list. Since the algorithm is a polynomial-time heuristic greedy algorithm, it ensures high computational speed and satisfactory allocation quality even for large data volumes under strict time constraints.

2.5. Algorithm based on random sample partitioning

To address the problem of uneven task distribution among employees, which leads to overload of some and underutilization of others, [22] describes software that enables, with the participation of a human manager, the effective allocation of new tasks based on the analysis of current employee workload $w(e)$ and task complexity $s(t)$. The software is based on the Random Sample Partition Algorithm (RSP) [23], which processes large volumes of data by dividing them into smaller data blocks available for direct human analysis. The analysis results are presented in a convenient form for the manager, who subsequently makes the final decision regarding task assignment.

Based on information about task priority $p(t)$ and complexity $s(t)$, as well as information about available time $a(e)$ and current workload $w(e)$ of performers, a set D is formed containing data on tasks and perform-

ers. The elements of D are pairs (t, e) supplemented with characteristics that allow evaluation of task execution efficiency and the level of professional development. The set D is randomly divided into K non-overlapping subsets $\{D_k, k = \overline{1, K}\}$ of approximately equal size.

For each subset D_k , an analysis is performed aimed at determining the current workload $w(e)$ of each performer, evaluating task complexity $s(t)$ and priority $p(t)$, and calculating the aggregate workload of each performer. The result is an informational object $R(D_k)$ in the form of an analytical report containing, among other things, information about the number of tasks assigned to each performer or group of performers, the composition of the subset of tasks, their execution time, complexity, priorities, as well as potential overload of employees and their available resources sufficient to execute new tasks.

The analytical reports $R(D_k)$ obtained for all subsets are combined into a general result, a report R_{total} , which represents recommendations for task allocation taking into account workload balancing and task complexity. The resulting report R_{total} is displayed in the graphical user interface of the software for the manager, enabling the latter to make the final decision regarding task assignment to performers. The algorithm under consideration performs the collection and structuring of data on tasks and performers, as well as primary workload analysis, which allows it to be regarded as a possible tool within decision support systems, providing recommendations to a manager responsible for task allocation and personnel management.

2.6. Neural network based workload adjustment algorithm

When considering the production planning problem as a multi-objective optimization problem, the authors of [24] justified that the implementation of traditional task allocation systems may be limited due to the restricted set of fixed criteria used in them. To

overcome this limitation and enable consideration of additional factors that are difficult to formalize, such as order priorities or equipment specifics, a hybrid algorithm was proposed that combines traditional optimization methods with plan adjustment using neural networks. The neural network based corrector used in the algorithm is based on two types of artificial neural networks (ANNs): a multilayer perceptron (MLP) [25] and a self-organizing map (SOM) by Kohonen [26].

An important feature of the algorithm is the grouping of performers into types of work centers and tasks into task groups. All tasks from the set T are grouped according to their relevance to work center types R using the Kohonen SOM trained via a self-organization algorithm. For each task t , a work center type $R(t)$ is determined that can execute it in the shortest time. Tasks are combined into groups $G(t)$, where each group corresponds to the type of work center most suitable for executing tasks of this category. Based on historical data and predefined expert rules, an MLP-type ANN is trained, which dynamically adjusts task priorities $p(t)$, the distribution of tasks among groups $G(t)$, and the selection of performers, taking into account available time $a(e)$ and workload $w(e)$. After the correction stage, tasks from the set T are assigned to performers from E in such a way as to minimize total completion time, balance workloads $w(e)$, and preserve task priorities $p(t)$ and available time constraints $a(e)$.

The algorithm also incorporates certain constraints. For example, not all tasks may be assigned to any performer, and a decisive assignment mechanism is possible: if a task belongs to a high-priority order, it may be rigidly assigned to a specific performer regardless of optimality according to other criteria.

2.7. Heuristic front based algorithm

Based on the analysis of the problem of optimal allocation among performers within a project consisting of tasks connected by dependencies representable as an

acyclic directed graph, [27] proposes a heuristic algorithm that allows finding an acceptable allocation of tasks within given time limits while minimizing the project execution cost. The algorithm is based on the principle of front based execution, according to which tasks are sequentially assigned to performers based on task labor intensity and performer availability.

For formalization, additional definitions are introduced: $D(t)$ is the set of predecessor tasks for task t ; $R(t) \subseteq E$ is the set of performers capable of executing task t ; and the labor intensity $s(t, e)$ of task t depends on the specific performer e . The front of tasks at step k , denoted F_k , represents a subset of tasks from T whose predecessor tasks have already been completed, that is, $D_k(t) = \emptyset$. Tasks from F_k become candidates for allocation. For each task $t \in F_k$ and each performer $e \in R(t)$, the start time of task t by performer e is calculated as:

$$x(e) = \max\left(y_k(e), \max_{t' \in D(t)} y(t')\right) \quad (8)$$

and the completion time is determined as:

$$y(e) = x(e) + s(t, e), \quad (9)$$

The task is assigned to the performer e^* for whom $y_k(e)$ is minimal. For this performer, the values of $a(e^*)$ and $w(e^*)$ are updated, and for the task the start time $x(t) = x_k(e^*)$ and completion time $y(t) = y_k(e^*)$ are fixed. Allocation of tasks from the set F_k continues until it is exhausted.

After assigning all tasks from the current front F_k , a new front F_{k+1} is formed, including tasks whose predecessor tasks have now been completed, that is, $D_{k+1}(t) = \emptyset$, and which require allocation. At the same time, the completion times of already assigned tasks for performers are updated as $y_{k+1}(e) = y_k(e)$. The formation of new task fronts continues until all tasks from the set T have been allocated.

After allocation of all tasks, feasibility and optimality are verified. Feasibility consists in checking com-

pliance with directive deadlines $d(t)$ for each task, that is, $y(t) \leq d(t)$. Optimality is assessed using an objective function whose minimization reduces the total project execution cost:

$$f = \sum_{e \in E} \sum_{t \in T} p(t) x(t, e) (d(t) - y(t, e)). \quad (10)$$

If the generated plan is feasible and optimal, the project schedule is formed and the total cost is evaluated. Alternative allocation variants are generated by different assignments of performers to tasks. The first front, consisting of tasks without predecessors, is determined by complete enumeration of all possible assignments. Subsequently, for each new task, all available performers are considered, and execution times are calculated taking into account workload and task dependencies. Thus, different variants correspond to different combinations of assignments $x(t, e)$, and the optimal one is selected by comparing the resulting combinations using the objective function that ensures the minimum project cost.

2.8. Iterative task delegation algorithm

When considering an approach to automating the task delegation process in project management, [28] proposed a model based on the use of iterative algorithms operating on data regarding employee experience, availability, and preferences stored in a NoSQL-type database. The main objective of developing such an algorithm was to reduce the involvement of the project manager in routine task allocation and to improve assignment quality through analysis of accumulated data. The authors present an iterative task delegation algorithm based on task keywords $K(t)$ and rating indicators of performer e . If the standard algorithm fails to assign a task due to insufficient data, a backup assignment mechanism is applied, taking into account the set of professional skills and preferences of performers.

The algorithm operates as follows. When a new task $t \in T$ is added, its keywords $K(t)$ are extracted. For each keyword $k \in K(t)$, performers $e \in E$ with the highest rating $R(e, k)$ are identified, and their ratings are summed to form an overall rating $R_s(e)$ for each performer. A list of performers ranked in descending order of $R_s(e)$ is formed. This list is then sequentially processed to check whether the current performer is available and capable of accepting the task, taking into account their workload $w(e)$ and available time $a(e)$. If a suitable performer is found, the task is assigned to them.

If no suitable performer is found, a backup algorithm is triggered. It uses professional skills $Q(e)$ and personal preferences $L(e)$ of the performer to compute a combined rating:

$$R_c(e) = Q(e) \cdot w_q + L(e) \cdot w_l, \quad (11)$$

where w_q and w_l are weights determining the relative importance of skills and preferences, respectively. The most suitable performer is then selected in a manner analogous to the selection based on $R_s(e)$. If the task remains unassigned after this stage, it is marked as requiring manual assignment. Such tasks are assigned by the project manager after algorithmic processing of the entire set of tasks T .

2.9. Task delegation algorithm based on a multi-agent system

In [29], an algorithm for planning, decomposition, and delegation of tasks in an unstructured decentralized environment is presented, where agents possess limited information about their own capabilities and the capabilities of other agents. The described approach to task allocation using a multi-agent system is based on the assumption that each agent may partially execute a task and delegate the remaining parts to other agents. The approach relies on a recursive task decom-

position algorithm based on applied artificial intelligence methods, which allows a task to be decomposed into subtasks, enabling the agent to execute those subtasks it can implement independently and delegate the remaining subtasks to other agents. Such an assignment process continues recursively until the entire task is distributed or a stopping condition is reached. The algorithm operates under conditions of incomplete information about the capabilities of all agents and supports partial planning, allowing planning with abstract actions that are later replaced with specific steps through delegation.

An iteration of the performer assignment algorithm for a task t proceeds as follows. First, it is verified whether the task is primitive, that is, does not require further decomposition. If so, among the performers E , a performer e is identified for whom condition (4) holds and the competency sufficiency condition $C(t) \subset C(e)$ is satisfied. If such a performer is found, the task is assigned to them; otherwise, the task is marked as unresolved.

If the original task t is not primitive, the agent invokes a decomposition function $D(t)$, which returns the set of its subtasks $\{t'_k, k = \overline{1, K}\}$. For each subtask t' , if the condition $C(t') \subset C(e)$ is satisfied, the agent e assigns the subtask to itself, updating its workload according to (5). If the condition is not satisfied, the subtask is delegated to another agent $e' \in E$ possessing a sufficient set of competencies for its execution, that is, $C(t') \subset C(e')$. Delegation of subtasks continues until all subtasks are assigned or an additional stopping condition is reached.

As an additional development of the algorithm, a two-phase mechanism may be used. In the first phase, the agent requests readiness from other agents to execute the subtask, and in the second phase, the subtask is transferred to the performer who first confirms the ability to execute it.

2.10. Adaptive expectation based algorithm

The systematization of approaches aimed at improving the efficiency of crowdsourcing using applied artificial intelligence methods, conducted in [30], made it possible to identify three key directions: task delegation, performer motivation, and quality control. For each of these directions, an updated taxonomy was proposed, and limitations and development prospects were analyzed. Based on this analysis, the authors proposed an algorithm for delegating complex tasks using the WMST (Weighted Multi-Skill Tree) model [31], which is employed to evaluate performer skills. The purpose of developing this algorithm was to provide a procedure for assigning tasks to performers with optimal skill matching and workload balancing.

For each performer e , a WMST model is preliminarily constructed to assess their competencies $C(e)$. During allocation of the task set T , for each task t , a performer e is selected who satisfies three conditions: correspondence of competencies $C(t) \subset C(e)$; sufficiency of execution resources $A(e) \geq S(t)$; and minimal current workload $W(e)$. The last condition is aimed at balancing the overall workload of the entire performer set E . If the selected performer is temporarily unavailable, the algorithm revises the assignment by selecting a new performer satisfying the above conditions.

3. Comparative analysis of algorithms

The comparative analysis of task allocation algorithms was carried out according to a number of criteria allowing for a comprehensive evaluation of their effectiveness and practical applicability. The degree of consideration of competencies reflects the extent to which the correspondence between performer characteristics and task requirements is taken into account, thereby determining the algorithm's ability to ensure optimal allocation based on professional skills and experience.

Adaptability to changes characterizes the flexibility of the algorithm and its ability to adjust decisions when the composition of performers changes, execution conditions are modified, or new tasks appear. Data requirements indicate the volume, structure, and accuracy of input information necessary for correct algorithm operation. Robustness to incomplete or noisy data determines the reliability of the algorithm under conditions of uncertainty and informational distortions. Transparency and explainability of decisions are associated with the interpretability of obtained results and the possibility of analyzing the reasons influencing the selection of a specific performer. Computational complexity characterizes the amount of computational resources and time required to obtain a solution, while scalability reflects the algorithm's ability to maintain efficiency as the number of tasks and performers increases. Implementation cost includes the total organizational, software, and hardware expenses associated with deploying and maintaining the algorithm. Finally, orientation toward personnel development reflects the potential of the algorithm to support employee qualification enhancement, identify competency gaps, and contribute to forming an optimal structure of labor distribution.

The analysis of *Table 1* shows that different algorithms possess specific advantages and limitations, which determine their areas of practical application. The main recommendations for selecting an appropriate approach are presented below.

The **Round-robin** algorithm described in Section 2.1 represents a simple and transparent method applicable under conditions of relatively uniform workload. It is effective in situations requiring rapid and low-cost task allocation with limited input data and high demands for explainability.

The **assignment matrix method** described in Section 2.2 is recommended when detailed task data are available and high transparency of decision-making is important. However, this approach demonstrates lower robustness to noisy data and limited scalability.

Table 1.

Algorithm comparison by key criteria

No.	Criterion	Round Robin (2.1)	Assignment Matrix (2.2)	Genetic Algorithm (2.3)	Heuristic Greedy (2.4)	RSP (2.5)	ANN Corrector (2.6)	Front-Based (2.7)	Iterative (2.8)	MA Approach (2.9)	WMST Model (2.10)
1	Competency consideration	None	Lim	Lim	None	Lim	Lim	Lim	Lim	Lim	Lim
2	Adaptability	Low	Med	High	Low	Med	High	Med	High	High	High
3	Data requirements	Min	High	Med	Min	Med	VH	Med	Med	High	High
4	Robustness to imperfect data	High	Low	Med	Med	High	Med	Med	Med	High	Med
5	Transparency	Full	High	Low	High	Med	Low	Med	Med	Low	Low
6	Computational cost	Low	High	High	Low	Med	High	Med	Med	High	High
7	Scalability	High	Lim	High	High	High	Med	Med	High	High	Med
8	Implementation cost	Low	High	Med	Low	Med	VH	Med	Med	High	High
9	Personnel development focus	None	Part	Yes	None	Part	Yes	Part	Part	Yes	Yes

Legend: Limited – Lim, Minimal – Min, Medium – Med, Very High – VH, Partial – Part.

The **genetic algorithm** described in Section 2.3 is advisable for solving complex optimization problems with multiple constraints. Its advantage lies in high adaptability to changing conditions and the ability to search for globally optimal solutions. At the same time, it requires significant computational resources and time costs.

The **heuristic (greedy) algorithm** described in Section 2.4 is oriented toward rapid solution generation

under limited data availability and high dynamics. Its advantage is high execution speed and low computational complexity; however, the solutions obtained are generally far from optimal.

The **RSP method** described in Section 2.5 demonstrates robustness when working with incomplete and noisy data. It provides a compromise between randomness and optimality, making it applicable under uncertainty conditions.

The **ANN based corrector** described in Section 2.6 shows high effectiveness in self-learning systems designed to process large data volumes. This makes it applicable for forecasting and adaptive management, although its implementation requires significant resources and high computational capacity.

The **front based algorithm** described in Section 2.7 provides balanced results under conditions of moderate task complexity. Its main advantage lies in its universality, making it applicable when there are no strict requirements for optimality or robustness.

The **iterative algorithm** described in Section 2.8 is recommended when the solution needs to be gradually refined. This approach is effective in dynamic environments, allowing step-by-step improvement and adaptation to changes.

The **multi-agent (MA) approach** described in Section 2.9 is most appropriate for systems involving interaction among multiple participants or subsystems. It is effective under high environmental variability and for collective-type tasks.

The **WMST based algorithm** described in Section 2.10 is applicable to long-term strategic planning in large-scale systems. It provides high adaptability and robustness but requires significant resources for implementation and maintenance.

Conclusion

The results of the analysis of ten task allocation algorithms published over the past twenty years allow us to conclude that modern approaches, despite their considerable diversity, still possess a number of significant limitations that constrain their application as universal tools within systems for automation and optimization of business processes. The primary reason for this lies in the fact that most of the examined algorithms are based on formal processing of quantitative characteristics, such as

employee workload, time expenditures for task execution, or task priorities, while excluding from consideration the individual competencies of performers. Taking into account only quantitative characteristics may lead to situations in which task assignments are made without sufficient regard for the correspondence between actual skills and possible employee preferences, which may negatively affect overall project performance.

At the same time, approaches characterized by greater flexibility and adaptability, supporting practical personnel management in complex projects, require that performer competencies be considered not as scalar, binary, or simple numerical parameters, but as multi-dimensional characteristics, individual components of which influence the quality, speed, and reliability of task execution in different ways. However, modern task allocation algorithms either insufficiently incorporate this factor or introduce it into their models in a simplified manner, without adequate mathematical and logical substantiation. This indicates an existing need for the development of new, more advanced task allocation methods that would organically combine standard formally quantifiable indicators with competency analysis as a key element of decision-making.

Thus, despite the existence of a number of new solutions in the field of automated task management, the creation of algorithms capable of comprehensively taking into account not only temporal and workload indicators but also professional skills of employees remain a challenging problem. Such approaches could serve as innovative tools for improving efficiency in task allocation, ensuring minimization of erroneous decisions and increasing employee motivation. ■

Acknowledgments

This research was funded by state assignment of the Ministry of Science and Higher Education of the Russian Federation, projects No. FSFF-2026-0008.

References

1. Baldin, K. V., Perederyaev, I. I., & Golov, R. S. (2017). *Upravlenie riskami v innovatsionno-investitsionnoi deyatel'nosti [Risk management in innovation and investment activities]*. Moscow: Dashkov i K (in Russian).
2. Chauhan, N., Kaur, N., Saini, K. S., Verma, S., Alabdulatif, A., Khurma, R. A., Garcia-Arenas, M., & Castillo, P. A. (2024). A systematic literature review on task allocation and performance management techniques in cloud data center. *Computer Systems Science and Engineering*, 48(3), 571–608. <https://doi.org/10.32604/csse.2024.042690>
3. Zhang, P., Zhang, A., & Xu, G. (2020). Optimized task distribution based on task requirements and time delay in edge computing environments. *Engineering Applications of Artificial Intelligence*, 94, 103774. <https://doi.org/10.1016/j.engappai.2020.103774>
4. Al-Fraihat, D., Sharrab, Y., Al-Ghuwairi, A.-R., Alzabut, H., Beshara, M., & Algarni, A. (2024). Utilizing machine learning algorithms for task allocation in distributed agile software development. *Heliyon*, 10(21), e39926. <https://doi.org/10.1016/j.heliyon.2024.e39926>
5. Skaltsis, G. M., Shin, H.-S., & Tsourdos, A. (2023). A review of task allocation methods for UAVs. *Journal of Intelligent & Robotic Systems*, 109(4), 76. <https://doi.org/10.1007/s10846-023-02011-0>
6. Lerner, I. M., Marinosyan, A. Kh., Grigoriev, S. G., Yusupov, A. R., Anikyeva, M. A., & Garifullina, G. A. (2024). An approach to the formation of intellectual academic genealogy using large language models. *Electromagnetic Waves and Electronic Systems*, 29(4), 108–120 (in Russian). <https://doi.org/10.18127/j5604128-202404-09>
7. Grigoriev, S. G., Lerner, I. M., Marinosyan, A. Kh., Arutyunova, N. K., & Grigorieva, M. A. (2025). On the issue of educational and methodological information selection for implementing an adaptive learning management system: Algorithm of a priori authors classification. *Informatics and Education*, 40(2), 66–78 (in Russian). <https://doi.org/10.32517/0234-0453-2025-40-2-66-78>
8. Grigoriev, S. G., Lerner, I. M., Marinosyan, A. Kh., & Grigorieva, M. A. (2025). On the issue of educational and methodological information selection for implementing an adaptive learning management system: Algorithm of selecting authors of literature taking into account the emotional and psychological characteristics of users based on the ideas of academic genealogy. *Informatics and Education*, 40(3), 69–79 (in Russian). <https://doi.org/10.32517/0234-0453-2025-40-3-69-79>
9. Beklaryan, A. L., & Akopov, A. S. (2018). Simulation model of the optimal allocation credit applications for interregional underwriting center of a commercial bank. *Vestnik Komp'uternyykh i Informatsionnykh Tekhnologii*, 11, 46–56 (in Russian). <https://doi.org/10.14489/vkit.2018.11.pp.046-056>
10. Taranenko, D. P., & Kolesnikov, A. V. (2021). Features of delegation of powers in the context of the development of remote control technologies. *Normirovanie i Oplata Truda v Promyshlennosti (Rationing and Remuneration of Labor in Industry)*, 12, 56–58 (in Russian). <https://doi.org/10.33920/pro-3-2112-06>
11. Schaad A., & Pymont B. (2010). Review mechanism for controlling the delegation of tasks in a workflow system. US Patent No. 7831978.
12. Akhavein, J., Frame, W. S., & White, L. J. (2005). The diffusion of financial innovations: An examination of the adoption of small business credit scoring by large banking organizations. *The Journal of Business*, 78(2), 577–596. <https://doi.org/10.1086/427639>

13. Gupta, D., & Denton, B. (2008). Appointment scheduling in health care: Challenges and opportunities. *IIE Transactions*, 40(9), 800–819. <https://doi.org/10.1080/07408170802165880>
14. Abanda, F. H., Musa, A. M., Clermont, P., Tah, J. H. M., & Oti, A. H. (2020). A BIM-based framework for construction project scheduling risk management. *International Journal of Computer Aided Engineering and Technology*, 12(2), 182. <https://doi.org/10.1504/ijcaet.2020.105575>
15. Voigt, S., & El Bialy, N. (2013). Identifying the determinants of judicial performance: Taxpayers' money well spent? *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.2241224>
16. Aksin, Z., Armony, M., & Mehrotra, V. (2007). The modern call center: A multi-disciplinary perspective on operations management research. *Production and Operations Management*, 16(6), 665–688. <https://doi.org/10.1111/j.1937-5956.2007.tb00288.x>
17. Yadav, P. K., & Saxena, S. (2020). Design and implementation of Round Robin scheduling algorithm using dispatch latency. *International Online Conference on Emerging Trends in Multi-Disciplinary Research (ETMDR-2020)*, 356–365.
18. Eremina, I. I., & Lysanov, D. M. (2020). Matematicheskaya model' optimizatsii protsessa raspredeleniya zadach i trudovykh resursov na predpriyatii [Mathematical Model for Optimizing the Process of Task and Labor Resource Distribution in Enterprises]. *American Scientific Journal*, 42, 65–71 (in Russian).
19. Korchevskaya, E. A., Ermachenko, S. A., Nikonova, T. V., Markova, L. V., & Shpakova, Y. A. (2023). Ispol'zovanie geneticheskogo algoritma dlya resheniya zadachi raspredeleniya uchebnoi nagruzki [Using a genetic algorithm to solve the problem of distributing academic workload]. *Vestnik Vitebskogo gosudarstvennogo universiteta*, 3(120), 15–19 (in Russian).
20. Akopov, A. S., & Hevencev, M. A. (2013). A Multi-agent Genetic Algorithm for Multi-objective Optimization. *2013 IEEE International Conference on Systems, Man, and Cybernetics*, 1391–1395. <https://doi.org/10.1109/smc.2013.240>
21. Munerman, V. I., & Munerman, D. V. (2019). Analysis of an optimal distribution algorithm. *Modern Information Technologies and IT-Education*, 15(3), 619–625 (in Russian). <https://doi.org/10.25559/sitito.15.201903.619-625>
22. Suresh, M., Samuel, R. B., Bhuvaneshwar, T., Jaubin, R. H., & Balaji, R. (2020). Automation of employee workload management using random sample partition algorithm. *International Journal of Recent Technology and Engineering (IJRTE)*, 8(6), 5282–5286. <https://doi.org/10.35940/ijrte.f9761.038620>
23. Salloum, S., Huang, J. Z., & He, Y. (2019). Random sample partition: A distributed data model for big data analysis. *IEEE Transactions on Industrial Informatics*, 15(11), 5846–5854. <https://doi.org/10.1109/tii.2019.2912723>
24. Luk'yanov, L. A., Spivak, S. I., & Khristolyubov, V. L. (2016). Neironnaya set' korrektor dlya raspredeleniya rabot v zadache vnutritsekhovogo planirovaniya [Neural network corrector for job distribution in intra-shop scheduling]. *Vestnik Bashkirskogo universiteta*, 21(4), 859–863 (in Russian).
25. Efimov, E. N., & Shevgunov, T. Ya. (2012). Development and analysis of the technique for the building of artificial neural networks based on adaptive elements. *Trudy MAI*, 51 (in Russian). <https://trudymai.ru/eng/published.php?ID=29159>

26. Kokhonen T. (2017). *Samoorganizuyushchiesya karty [Self-Organizing Maps]*. Moscow: Laboratoriya znanii (in Russian).
27. Novikova, T., & Novikov, A. (2015). Algorithms for solving problems of optimum distribution work in network canonical structures. *Forestry Engineering Journal*, 4(4), 309–317 (in Russian). <https://doi.org/10.12737/8515>
28. Pop, B., & Boian, F. (2014). Algorithms for automating task delegation in project management. *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, 2, 1191–1196. <https://doi.org/10.15439/2014f426>
29. Lopes, A. L., & Botelho, L. M. (2007). Task decomposition and delegation algorithms for coordinating unstructured multi agent systems. *First International Conference on Complex, Intelligent and Software Intensive Systems (CISIS'07)*, 209–214. <https://doi.org/10.1109/cisis.2007.52>
30. Wang, S., Li, Q., Cui, L., Yan, Z., Xu, Y., Shi, Z., Min, X., Shen, Z., & Yu, H. (2022). Towards AI-empowered crowdsourcing. *arXiv:2212.14676*. <https://doi.org/10.48550/arXiv.2212.14676>
31. Al-Anzi, F. S., Al-Zame, K., & Allahverdi, A. (2010). Weighted multi-skill resources project scheduling. *Journal of Software Engineering and Applications*, 03(12), 1125–1130. <https://doi.org/10.4236/jsea.2010.312131>

About the authors

Timofey Yakovlevich Shevgunov

Candidate of Sciences (Technology);

Associate Professor, Moscow Aviation Institute (National Research University), 4 Volokolamskoe Hwy., Moscow 125993, Russia;

Associate Professor, Department of Business Informatics, Graduate School of Business, HSE University, 26–28 Shabolovka St., Moscow 119049, Russia;

E-mail: shevgunov@gmail.com

ORCID: 0000-0003-1444-983X

Anna Alexandrovna Kroshilina

Master's student, Department of Business Informatics, Graduate School of Business, HSE University, 26–28 Shabolovka St., Moscow 119049, Russia;

E-mail: ankrosh@vk.com