The object of this study is the task scheduling process in heterogeneous distributed information systems. The scientific task addressed relates to the low efficiency of resource management, especially under conditions of dynamic workload and significant uncertainty, which are typical for distributed information systems. An intelligent task scheduling method has been devised for heterogeneous distributed information systems, which effectively combines DAG (Directed Acyclic Graph) and GERT (Graphical Evaluation and Review Technique) models with advanced artificial intelligence algorithms. The proposed method employs a Graph Attention Network (GAT) to account for probabilistic dependences between tasks and Proximal Policy Optimization (PPO) for dynamic control of task distribution within the system. Furthermore, a Bayesian method is used to optimize the assignment of tasks to computing nodes. The use of the proposed method reduced the average task execution time from 51.5 to 35.2 seconds, and the standard deviation of the load between nodes from 0.47 to 0.22.

These results are explained by the flexibility of the models to unforeseen changes and the ability to self-learn based on accumulated data. A feature of the method is the combination of classical graph models with probabilistic estimation and adaptive AI mechanisms, which made it possible not only to take into account the dynamics of the environment but also to ensure accurate response to changes in resource availability. By using GERT graphs, the algorithm forms alternative planning paths in case of failures or unforeseen delays, and machine learning components provide self-correction of decisions. The method is oriented towards application in cloud and IoT infrastructures, in which scalability, planning accuracy, and resilience to changes are critical

Keywords: distributed computing systems, graph models, resource planning, intelligent management systems

UDC 004.75
DOI: 10.15587/1729-4061.2025.329263

DEVELOPMENT AN INTELLIGENT TASK SCHEDULING METHOD IN HETEROGENEOUS DISTRIBUTED INFORMATION SYSTEMS

Serhii Yenhalychev
PhD Student*
Oleksii Leunenko
PhD Student*
Viacheslav Davydov
Corresponding author
Doctor of Technical Sciences, Associate Professor
Department of Information Technology and Cyber Security
Science Entrepreneurship Technology University
M. Shpaka str., 3, Kiyv, Ukraine, 03113
E-mail: vyacheslav.v.davydov@gmail.com
*Department of Cybersecurity and Information Technologies
Simon Kuznets Kharkiv National University of Economics
Nauky ave., 9-A, Kharkiv, Ukraine, 61165

Received 20.02.2025
Received in revised form 18.04.2025
Accepted date 30.04.2025
Published date 25.06.2025

How to Cite: *Yenhalychev, S., Leunenko, O., Davydov, V. (2025). Development an intelligent task scheduling method in heterogeneous distributed information systems. Eastern-European Journal of Enterprise Technologies, 3 (9 (135)), 6–18. https://doi.org/10.15587/1729-4061.2025.329263*

1. Introduction

Distributed information systems play a key role in enabling efficient computing in modern technologies, including high-performance computing, cloud platforms, Internet of Things systems, and financial computing. One of the main challenges facing such systems is optimal task scheduling that ensures load balancing, latency minimization, and efficient resource utilization. The heterogeneity of environments, which include different types of computing nodes, network connections, and data stores, complicates this process. In addition, the dynamism of the environment and the uncertainty of metadata further complicate decision-making in the planning process. Existing approaches to task distribution based on conventional methods often do not fully take these aspects into account, which reduces their effectiveness in complex heterogeneous systems.

One promising approach to improving the planning process is the use of graph models to represent dependences between tasks. Parallel task graphs make it possible to formalize computational processes, determine critical paths, and assess the possibilities of parallel execution. Our study proposes a method for scheduling tasks in heterogeneous distributed systems, which is based on the use of three main characteristics of graphs: critical path, graph density, and level separation [1]. Unlike conventional approaches, the proposed method applies probabilistic algorithms that make it possible to more effectively determine optimal task distributions between system resources. In particular, employing the Univariate Marginal Distribution Algorithm makes it possible to significantly reduce the scheduling time compared to genetic algorithms traditionally used to solve similar problems [2].

The proposed method could find wide application in various fields, including high-performance computing, big data processing, autonomous systems management, network traffic optimization and load balancing in cloud environments. The use of a structural graph model allows for more efficient task distribution and for adapting the scheduling process to dynamic changes in the execution environment. Given the need for scalable and adaptive approaches to task scheduling, the research reported in this work is aimed at devising a method that would provide increased productivity and resource efficiency in modern distributed information systems.

Despite significant progress in the field of task scheduling in distributed computing environments, previous approaches leave a number of important problems unresolved. In particular, most conventional methods do not take into account the stochastic nature of interdependences between tasks, do not provide adaptability to dynamic changes in load and resources, and have limited effectiveness in heterogeneous environments. This necessitates devising new approaches to task scheduling that combine classical graph models with artificial intelligence methods to increase flexibility, forecasting accuracy, and resilience to environmental changes.

Thus, given the complexity of modern distributed environments, the presence of stochastic factors in the operation of computing systems, and the limitations of existing approaches to ensuring flexible scheduling, research on this topic is relevant. Devising an intelligent method that could adapt to changes in real time and ensure effective resource management is of great importance for the further development of high-performance computing platforms, cloud infrastructures, and IoT systems.

2. Literature review and problem statement

Study [3] reports a systematic analysis of task scheduling algorithms, including conventional deterministic methods (First Come First Serve (FCFS), Round-Robin) and modern heuristic approaches. The findings of the study indicate that classical methods are not able to scale efficiently in large heterogeneous systems. However, it remains an open question how effective these methods would be in adapting to variable load and resource heterogeneity.

Paper [4] emphasizes that conventional methods can still be effective in certain scenarios, such as systems with low dynamicity. However, the authors do not consider how conventional approaches could be improved to remain effective in environments with medium resource variability.

Work [5] analyzes the challenges associated with task scheduling in cloud computing and HPC (High-Performance Computing) systems. The authors emphasize that the main problems are scalability, load forecasting, and adaptation to changes in the environment. In particular, the importance of dynamic load balancing between resources is emphasized, which makes it possible to significantly increase the efficiency of calculations. At the same time, the work does not offer a detailed analysis of the impact of different balancing strategies on the performance of systems in the long term.

Paper [6] reports a new approach to dynamic balancing that uses real data from cloud environments to improve the accuracy of forecasting. However, the work does not consider the potential limitations of applying this approach in cases where real data may be incomplete or inaccurate.

In study [7], a comparison of static and dynamic task allocation methods is considered. The authors conclude that dynamic scheduling methods that use adaptive mechanisms have a significant advantage in changing environments, especially in the context of cloud and distributed systems. However, it is not investigated how adaptive mechanisms can interact with hybrid scheduling methods that combine static and dynamic approaches.

In [8], it is noted that static methods can be effective in systems with a low level of dynamism, where resources remain stable over a long time. However, the authors do not consider the possibility of combining static methods with methods for predicting changes in the environment for their potential improvement.

In work [9], the advantages of GERT (Graphical Evaluation and Review Technique) modeling are described. The prospects for its improvement under conditions of data uncertainty are presented. However, the authors did not describe the possibility of using this technology together with artificial intelligence algorithms.

The development of metaheuristic algorithms has contributed to improving the efficiency of task scheduling.

In [10], the use of genetic algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) algorithms is considered. The study demonstrates that these methods can significantly reduce the execution time of tasks, but the effectiveness of their use depends on the configuration of parameters. At the same time, it does not consider how real-time parameter adaptation would affect the performance of these methods.

In [11], a new approach to optimizing the parameters of metaheuristic algorithms is reported, which makes it possible to increase their efficiency in heterogeneous environments. However, the question of adapting these algorithms to highly dynamic changes in the environment without significant computational costs remains open.

Study [12] analyzes the application of deep learning for load forecasting and resource allocation. It was demonstrated that neural networks are able to accurately predict the future load of the system, which allows for more efficient resource allocation. However, the authors do not consider how robust the model would be to unexpected changes in the workload and failures of individual nodes.

In [13], a new method is reported that combines deep learning with optimization methods to improve the accuracy of the forecast. However, the question remains open about the complexity of implementing such an approach in large distributed systems, where training neural networks can be a resource-intensive process.

Study [14] investigates the use of adaptive algorithms in multi-agent systems where the load changes in real time. A mechanism is proposed that dynamically adjusts the task distribution, which leads to increased productivity and reduced waiting time. At the same time, the issue of the stability of multi-agent systems in cases of abrupt changes in resource availability remains unresolved.

In paper [15], a new approach to adaptive planning is presented that uses real data from multi-agent systems to improve efficiency. However, the authors do not consider the problem of planning consistency in the case of resource distribution between several independent agents with conflicting goals.

In addition, [16] gives an overview of hybrid methods that combine classical heuristic algorithms with machine learning methods. Such approaches demonstrate improved adaptability and accuracy in resource allocation. However, it is not clear how to balance the computational complexity of these methods so that they remain suitable for real-time operation.

In [17], a new hybrid method is reported that combines genetic algorithms with deep learning methods to improve scheduling efficiency. However, the robustness and efficiency of such an approach in cases of high input variability still needs further investigation.

Uncertainty is one of the main challenges in task scheduling since resource availability parameters can change in unpredictable ways.

In [18], the study analyzes the use of Bayesian networks to predict resource reliability and adapt task allocation to current conditions. However, the authors do not consider how the forecast accuracy may change when historical data is insufficient or when the system characteristics change in real time.

In [19], a new approach to the use of Bayesian networks is reported, which makes it possible to improve the forecast accuracy. However, the question of the computational complexity of this approach when scaling the system and processing large amounts of data remains open.

In study [20], fuzzy logic is used to model the load level of nodes in a computing network. The method makes it possible to estimate the priority of tasks taking into account the uncertainty in the measured parameters. At the same time, the study does not consider the possibility of integrating this approach with other load forecasting methods to increase its accuracy.

In [21], a new method is reported, which combines fuzzy logic with machine learning methods to improve the forecast accuracy. However, the authors do not analyze how effective this approach would be in the case of highly variable environments and what its potential limitations may be.

In paper [22], the model architecture and training method that integrate dynamic neural networks with an emphasis on resilience are described. However, the authors do not provide practical applications of these models for solving such complex problems as control in distributed systems.

In general, the identified shortcomings indicate the need to devise a new approach that would combine the ability to adapt to changes in load and resource availability in real time, as well as take into account the probabilistic nature of dependences between tasks. In addition, it should ensure the minimization of task execution time and high accuracy of resource load forecasting.

3. The study materials and methods

The aim of our research is to devise an intelligent task scheduling method for heterogeneous distributed systems based on the hybrid DAG-GERT model using artificial intelligence algorithms, which provides adaptive resource management and execution time minimization in a dynamically changing environment.

To achieve this goal, the following tasks were set:

– to formalize task scheduling models in Heterogeneous Distributed Computing Systems (HDCS) by expanding the conventional model by adding GERT graphs that make it possible to estimate task execution variability and probabilistic delays.

– to build a model for predicting the critical path of tasks using GNN and machine learning methods;

– to implement adaptive task scheduling control based on reinforcement learning PPO;

– to optimize the distribution of tasks between resources using Bayesian optimization to ensure load balancing;

– to investigate the effectiveness of the proposed method by testing the performance comparison of the new approach with prototypes.

4. The study materials and methods

The object of our study is the process of task scheduling in heterogeneous distributed information systems. The principal hypothesis of the study assumes that the combination of probabilistic graph models and artificial intelligence methods provides improved adaptability and performance of systems under dynamic conditions. A number of assumptions were accepted in the study: tasks have clear dependences, resources are divided into clusters with previously known characteristics, communication delays between nodes are independent. A simplification was also applied: network delays were modeled as stationary, and computing power was modeled as the arithmetic average over the last period of activity.

The research methodology includes:

– theoretical modeling. Construction of a hybrid DAG + GERT model, formalization of transition probabilities and resource matrix;

– algorithmic environment. A software prototype of the scheduling system was developed using the Python language (USA) and the TensorFlow (USA), PyTorch (USA), NetworkX (USA), Matplotlib (USA) libraries;

– simulation. A virtual environment of a distributed system with parameterized nodes emulating clusters with different levels of load, delays, and failures was constructed. The Stable-Baselines3 library (Germany) was used to implement the PPO agent, and the Scikit-Optimize library (France) was used for Bayesian Optimization (BO);

– data processing. Statistical treatment of the results was performed using the NumPy (USA) and Pandas (Canada) libraries;

– efficiency assessment. The adequacy criteria included the average task execution time (makespan), the standard deviation of the load, and the average error of the critical path forecast. The adequacy of the model was checked by comparing it with conventional methods (FCFS, Heterogeneous Earliest Finish Time (HEFT), Greedy, Random).

The following initial data were used for modeling:

– number of tasks: from 20 to 50;

– number of computing nodes: from 5 to 15;

– average node performance: 1.5–3.0 arbitrary units;

– average network delays: 10–30 ms;

– transition probabilities in the GERT graph: 0.7–0.99.

The main assumptions of the study:

– tasks have clear dependences that do not change during execution;

– communication delays are stationary during the simulation time;

– resources are divided into clusters with previously known characteristics;

– node failures are not taken into account in the basic simulation.

Simulation limitations:

– node energy consumption is not taken into account;

– real failures in network data transmission channels are not simulated;

– PPO agents are trained in an environment with simplified parameters.

5. Devising an intelligent task scheduling method based on the hybrid DAG-GERT model

5. 1. Formalizing the task scheduling model in HDCS

Task scheduling in heterogeneous distributed computing systems is a complex task due to resource variability, dynamic changes in the environment, and communication delays. Effective management of execution processes requires the

use of models that take into account not only deterministic dependences between tasks but also probabilistic factors that affect their execution. For this purpose, an approach combining classical and probabilistic graph models is considered, which makes it possible to increase the adaptability and efficiency of the system.

A heterogeneous distributed computing system consists of a set of clusters $C = \{C_1, C_2, ..., C_k\}$, where each cluster contains a set of processors with different computing power and number of cores. Processors can be connected by local or remote networks, which affects the data transfer time between tasks.

The parallel task graph is represented as a directed acyclic graph (DAG) or a GERT graph, depending on the level of determinism of the process. In this case:

– DAG: $T = (N, E)$, where N is the set of nodes (subtasks), and E is the set of directed edges (dependences between tasks);

– GERT: $T = (N, E, P)$, where an additional set of probabilities P is included, which determines the probabilistic dependences between tasks.

The critical path in a DAG graph is defined as the longest path in the graph from the initial to the final node, which sets the minimum possible execution time for the entire task

$$M = \sum W_{n_i}, \tag{1}$$

where W_{n_i} is the execution time of the node n_i.

If tasks can have alternative execution options, according to the GERT modeling technology, their contribution to the total execution time is calculated taking into account the probability

$$M = \sum \left(W_{n_i} P_{n_i} \right), \tag{2}$$

where P_{n_i} is the probability of execution of node n_i.

To control the planning process, a resource matrix is used, which contains information about the available computing nodes, their performance, and network topology. Each resource is characterized by the power C_k and the transmission delay between nodes $d(i, j)$, which is formalized as a matrix

$$R_{i,j} = \begin{cases} 0, \text{ if } i = j \left(\text{local processor} \right), \\ d(i,j), \text{if } i \neq j \left(\text{dist between resources} \right). \end{cases} \tag{3}$$

Additionally, the probability of successful completion of the task on a particular computing node is introduced

$$P_{exec}\left(n_i, C_k\right) = \frac{1}{1 + e^{-\lambda\left(C_k - C_{avg}\right)}}, \tag{4}$$

where C_{avg} is the average power of all available computing resources, λ is the parameter of the system's sensitivity to the performance of a single node.

This formula makes it possible to estimate how efficiently a task can be executed on a particular processor and minimize the risks of overloading individual resources.

In addition, task scheduling involves forming a matrix of task characteristics, which includes the critical path, level partitioning, and graph density. Determining task execution levels makes it possible to identify nodes that can be executed in parallel, which increases the efficiency of resource use. The level width is defined as

$$w = \max_{1 \le k \le h} \left(\sum\nolimits_{v \in V_k} W_v \right), \tag{5}$$

where V_k is the set of tasks belonging to the same stratification level.

The task completion time (TCT) depends on the start time, the execution time of subtasks, and possible communication delays

$$TCT = St\left(T_n\right) + \sum_{i=1}^{N} \left(T_{TaskExecution} \times P_{n_i} \right). \tag{6}$$

The cost of communication between subtasks is estimated as the amount of data transferred between nodes during the execution of the graph

$$CC_{\tau_{i,j} \to \tau_{i,k}}.$$

Task allocation in HDCS involves finding the optimal processor or cluster for each subtask, taking into account computing power, workload, and network latency.

The quality of resource allocation is determined by the percentage of occupied processors relative to the total number of available resources

$$Q_a = \left(\frac{\sum_{i=0}^{m} \forall \pi_m \in \eta_i}{|\pi|} \right) \times 100. \tag{7}$$

The final stage of formalization is to determine the resource allocation matrix, which describes the correspondence between tasks and computing nodes. The total execution time of all tasks in the system is defined as

$$M_s = \max_{n_i \in N} \left(\sum_{C_k} \left(W_{n_i} \times P_{exec}\left(n_i, C_k\right) \right) \right). \tag{8}$$

Each task is assigned to a node that minimizes M_s and at the same time ensures a balanced load on the entire system.

To optimize the scheduling process, the Univariate Marginal Distribution Algorithm (UMDA) algorithm is used, which analyzes probabilistic patterns in possible distributions and selects the most efficient options. Formally, it is written as

$$p_l(x) = p(x \mid DSe_{l-1}) = \prod_{i=1}^{n} p_l(x_i), \tag{9}$$

where $p_l(x)$ is the probability of choosing a certain task distribution when optimizing the scheduling.

Due to this extension, the conventional DAG approach is supplemented with stochastic models, which improves adaptation to changing system operating conditions.

This conclusion can be proven by demonstrating the results of the study on a selected example.

Fig. 1 shows an example of a graph as an element of applying the devised model for task scheduling in HDCS.

Fig. 1 illustrates the task graph in HDCS, where the combined DAG and GERT approach is used for adaptive task scheduling. In this model, each node represents a separate task with a certain execution time, which is indicated inside each element. Directed edges define dependences between tasks, establishing the order of their execution, while the transition probabilities between them reflect stochastic dependences that take into account the variability of the computing environment. This approach makes it possible to take into account possible delays due to server loads, the probability of successful task execution under unstable conditions, as well as alternative task execution routes.

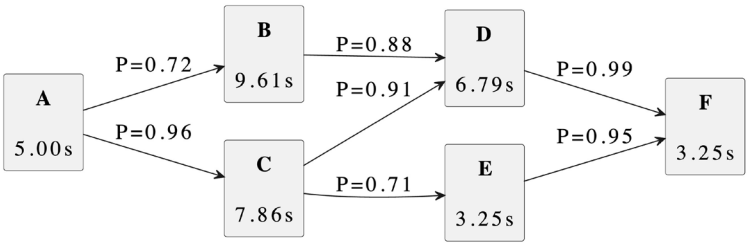


Fig. 1. Graph of tasks in a distributed heterogeneous computing system

The proposed model allows for flexible adaptive scheduling, which makes it possible for the system to predict alternative paths for task execution in the event of uncertainty. If the probability of transition between tasks is low, the algorithm can change the route to minimize the risk of delays and increase the efficiency of scheduling.

An additional advantage is the optimization of load balancing since classical DAG graphs can create overloaded nodes due to rigid fixation of execution paths. In the proposed model, the distribution of tasks between servers takes place taking into account the probabilistic analysis of their current state and available resources, which makes it possible to avoid bottlenecks and increase the performance of the computing system. In addition, the use of GERT graphs reduces the risk of failures since the system can predict alternative execution paths in advance if one of the transitions has a low probability, thereby ensuring the stability and resilience of the HDCS to possible failures.

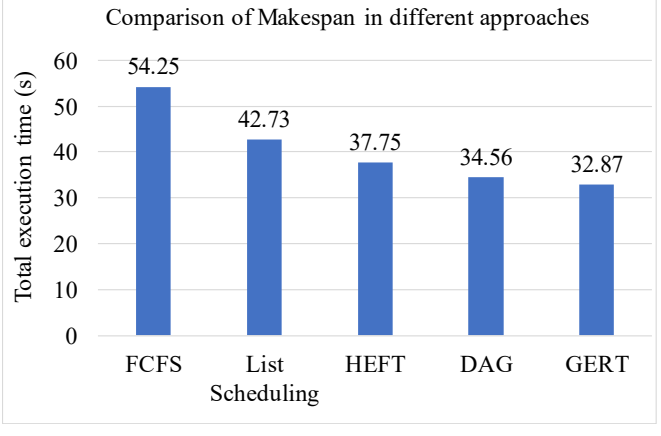
Fig. 2 shows comparative histograms of the total execution time and resource balancing for the different models studied.

Our results demonstrate the effectiveness of various task scheduling methods in HDCS in terms of total execution time (makespan) and balancing of computational resources.

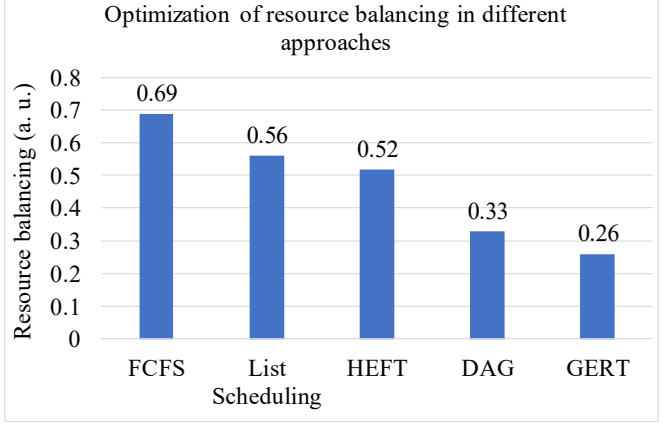
The first plot (Fig. 2, *a*) presents a comparison of the total execution time for five approaches: FCFS, List Scheduling, HEFT, DAG, and GERT. As can be seen from the histogram, the FCFS method demonstrates the worst results since it does not take into account the specificity of the computing environment and executes tasks in the order they arrive, which leads to significant delays. List Scheduling shows improved results since it takes into account a certain level of task priority but does not provide optimal balancing. HEFT demonstrates even better performance since it takes into account the heterogeneity of the environment and allows for more efficient distribution of tasks between resources. Using the DAG model allows for a significant reduction in makespan due to optimized graph scheduling but the best result is shown by the GERT model, which uses probabilistic dependences and adaptive task distribution, which allows for additional minimization of the total execution time.

The second plot (Fig. 2, *b*) demonstrates the optimization of resource balancing in different methods. The lower the histogram score, the better the balancing, as it means an even distribution of the load between the computing resources. The FCFS method has the worst results as it does not optimize the load distribution. List Scheduling and HEFT show a gradual improvement in balancing as they take into account the distribution of tasks according to the available node capacities. The DAG model significantly improves this indicator, but the best results are again achieved by the GERT model, which confirms its effectiveness in balancing resources through adaptive task management and the use of probabilistic dependences.

Thus, the results of our study show that the use of the GERT model in HDCS makes it possible to significantly reduce the total task execution time, improve resource balancing, and increase the flexibility of planning in variable environmental conditions. The obtained data confirm the advantages of integrating stochastic approaches into the task planning process in heterogeneous distributed systems.



a



b

Fig. 2. Comparative histograms of total runtime and resource balancing for different models studied

The proposed DAG + GERT model makes it possible to effectively perform task planning in distributed computing systems, taking into account stochastic factors and dynamic changes in resource availability. Using this approach makes it possible to adaptively manage the task distribution and ensure optimal load balancing. However, taking into account the growing complexity of modern distributed systems and the need for automatic decision-making, the question arises of further improving the planning mechanisms.

To further increase the efficiency of the task planning process, it is advisable to use artificial intelligence methods. The integration of deep learning and reinforcement learning algorithms makes it possible not only to predict the future load of the system but also to carry out adaptive control of task distribution in real time. This makes it possible to im-

prove the accuracy of critical path selection, optimize resource balancing, and minimize the total task execution time.

5. 2. Building an intelligent task scheduling model in heterogeneous systems

5. 2. 1. Building a graph task model based on DAG and GERT

To build an intelligent task scheduling model in heterogeneous distributed systems, a combination of a stochastic approach based on DAG and GERT graphs with artificial intelligence algorithms is proposed. The main idea of the model is to predict the critical path, adaptively distribute tasks between computing resources, and optimize load balancing taking into account probabilistic dependences between tasks and dynamic changes in the environment.

The method consists of several key components:

1. Critical path prediction and system load. Input data on the current state of the system and task characteristics are analyzed using Graph Neural Networks (GNN). This makes it possible to evaluate possible task execution options in the DAG + GERT graph and determine the most optimal route to minimize the total execution time (makespan).

2. Adaptive decision-making regarding task execution. During operation, the system can change the current plan if there is an overload of nodes or a change in resource availability. Reinforcement Learning (RL) algorithms are used for this purpose, which makes it possible to dynamically adapt the distribution of tasks in real time.

3. Assignment of processors to tasks and optimization of resource balancing. The use of Bayesian networks or genetic algorithms helps predict the most efficient assignment of processors to tasks, taking into account the current load, predicted delays, and communication costs. This makes it possible to improve the efficiency of using computing resources.

4. Self-learning and improvement mechanism. The system stores historical data on planning efficiency and uses deep learning algorithms to gradually improve its decisions. The more data is accumulated, the more accurately the system predicts future workloads and optimizes the critical path for task execution.

The list of algorithms used in the proposed method is given in Table 1.

Table 1

List of algorithms used in task scheduling in distributed information systems

Step of the method	Algorithm used	Role in the planning process
Processing of input data and construction of a DAG + GERT graph	DAG + GERT graph construction algorithm (DFS/BFS for DAG, probabilistic modeling for GERT)	Formation of the structure of the task graph with probabilistic dependencies
Predicting the critical path	Graph Convolutional Network (GCN)	Determination of the optimal critical path taking into account possible execution scenarios
Adaptive task management	Proximal Policy Optimization (PPO)	Adaptation of planning in real time depending on resource availability
Assignment of processors to tasks	Bayesian Optimization	Optimization of resource balancing and minimization of makespan

5. 2. 2. Predicting the critical path and system load

For effective task distribution in HDCS, it is necessary to accurately determine the critical path, which ensures the minimization of the total execution time. The use of classical methods, such as DAG graph analysis, does not allow for flexible adaptation to dynamic changes in the load and resources of the system. Therefore, it is advisable to use the Graph Attention Network (GAT), which makes it possible to predict the critical path taking into account probabilistic dependences between tasks and variability of the resource environment.

GAT is an extension of the classical Graph Convolutional Network (GCN) and uses the self-attention mechanism, which makes it possible to determine which connections between tasks are more important for the final result. This is especially true in the case of DAG + GERT graphs, where probabilistic transitions can significantly change the total execution time. The main equation that determines the weight coefficients between tasks takes the form

$$\alpha_{ij} = \left(\frac{e^{\left(LeakyReLU\left(a^T \left[W_{h_i} \| W_{h_j} \right] \right) \right)}}{\sum_{k \in N(i)} e^{\left(LeakyReLU\left(a^T \left[W_{h_i} \| W_{h_k} \right] \right) \right)}} \right), \quad (10)$$

where a_{ij} is the attention coefficient between a task and its neighbor;

W – model training weights;

a – attention vector, which determines the importance of neighboring tasks;

|| – feature concatenation operation.

This approach allows each node of the graph to receive information about neighboring nodes, taking into account their importance, which improves the critical path prediction process. Unlike GCN, which applies the same weights to all neighbors, GAT makes it possible to identify key nodes in the performance of tasks and optimize their distribution.

The critical path prediction process in GAT includes several important stages. First, the model receives a task graph in the form of a DAG + GERT as input, where each vertex contains input features: task execution time, resource utilization level, and transition probability to the next state.

At this stage, each node in the graph receives a vector of initial features, which includes such characteristics as:

– task execution time t_i is the time required to complete task i;

– transition probability P_{ij} is the probability that task i will proceed to task j;

– resource utilization R_i is the level of computing resources used to perform task i;

– degree of connectivity d_i is the number of incoming and outgoing connections of a node, indicating its importance in the graph.

These features form the initial matrix of node characteristics $H^{(0)}$

$$H^{(0)} = \begin{bmatrix} t_1 & P_{12} & R_1 & d_1 \\ t_2 & P_{23} & R_2 & d_2 \\ t_3 & P_{34} & R_3 & d_3 \\ \vdots & \vdots & \vdots & \vdots \\ t_n & P_{n-1,n} & R_n & d_n \end{bmatrix}. \quad (11)$$

These parameters form the initial state of the graph, which is the basic data for further training of the neural network.

Next, GAT performs a multi-stage analysis of the interaction of nodes, using the attention mechanism to determine the most important connections. In the first layer of attention, each node receives information from its immediate neighbors, determining which tasks have a greater impact on the overall result.

The importance of node j for node i is determined by the attention function

$$e_{ij} = \left(LeakyReLU\left(a^T \left[W_{h_i} \| W_{h_j} \right] \right) \right), \quad (12)$$

where e_{ij} is the initial importance of neighbor j for node i.

These values are then normalized via softmax to obtain the attention coefficients

$$\alpha_{ij} = \left(\frac{e^{(e_{ij}))}}{\sum_{k \in N(i)} e^{(e_{ik})}} \right). \quad (13)$$

These coefficients reflect the probability that information from a neighboring node will be important for predicting the critical path.

This process is repeated in subsequent layers, which makes it possible to gradually form a global view of the graph structure. Since the weight of the links is determined dynamically, the model adapts to changing task execution conditions, allowing one to identify nodes that are critical for effective resource allocation.

At the output, the network forms a priority matrix that reflects which tasks should be performed first to minimize the total execution time. This makes it possible to automatically rank possible execution routes, which significantly improves the accuracy of critical path prediction. In addition, the obtained link weights can be used for further optimization of planning, in particular for adjusting resource allocation in real time.

The final part provides for the integration of the predicted critical path into an adaptive task management system. Using a reinforcement learning approach, the system analyzes the results and optimizes resource balancing depending on changes in the load. Thus, the proposed method makes it possible to improve the efficiency of task distribution, reducing the overall execution time and ensuring adaptability to changes in the system.

The visualization of GAT operation shows how the model iteratively analyzes the connections between tasks, consistently improving the prediction of the critical path in HDCS. For example, Fig. 3 shows a graph of tasks in a distributed system, where each node corresponds to a separate task. Nodes contain two values: Real – the real execution time of the task, and Pred – the predicted execution time obtained using the GAT model.

Looking at the results of the Key Observations, the following can be noted. For a task with a real execution time of 2.00 a.u., the model predicted a time of 1.77 a.u., which indicates a small error. For a task with a real time of 3.00 a.u., the predicted value was 3.43 a.u., which also demonstrates high accuracy. For a task with a real time of 4.00 a.u., the predicted value of 3.47 a.u. is close to the real one, which confirms the model's ability to take into account dependences between tasks. For a task with a real time of 5.00 a.u. the predicted value of 4.83 a.u. is also quite accurate. Such results generally confirm the hypothesis of high accuracy of prediction.

In addition, the following general trend is observed. The model copes well with predicting the execution time of tasks, which confirms its effectiveness in taking into account both deterministic and probabilistic dependences in the task graph.

The graphical representation makes it possible to clearly assess how the model distributes the load between nodes and takes into account dependences between tasks. The directions of the graph edges indicate the order of task execution, which is important for planning in distributed systems.

Table 2 gives results of the GAT training process.

The GAT model training process demonstrates a stable decrease in the value of the loss function (MSE) with each training step. At the initial stage (epoch 0), the loss value was 2.2169, which is a rather high indicator since the model has not yet learned to take into account dependences between graph nodes. However, after 20 epochs, the loss value decreased to 0.8773, which indicates the initial training of the model and its ability to better predict the load on nodes.

Table 2

Results of the GAT learning process

Epoch	Loss
0	2.2169222831726074
20	0.8772878646850586
40	0.555362343788147
60	0.49738389253616333
80	0.4462957978248596
100	0.39999741315841675
120	0.35727459192276
140	0.3189087212085724
160	0.2854723334312439
180	0.25711387395858765

At epoch 40, the loss value decreased to 0.5554, and by epoch 60 it reached 0.4974. This indicates that the model continues to improve its predictions, taking into account the complex dependences between tasks in the graph. At epoch 80, the loss decreased to 0.4463, and by epoch 100, it reached 0.4000, which confirms the effectiveness of using GAT for scheduling tasks in distributed systems.

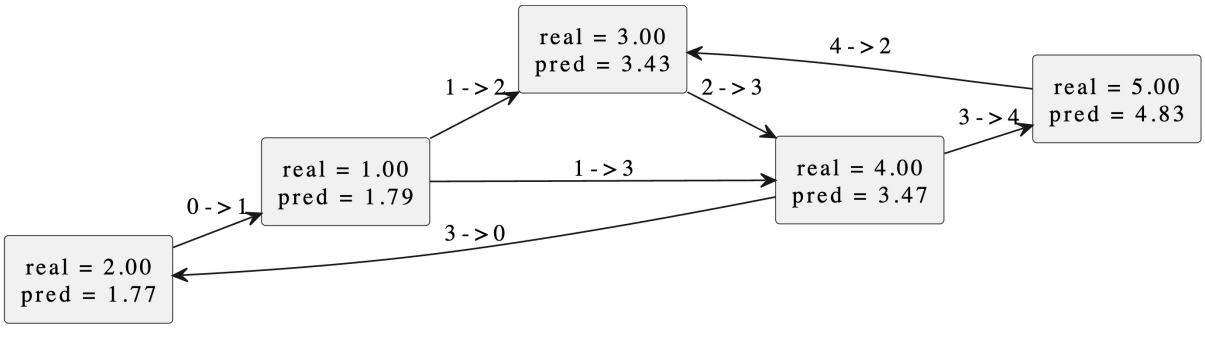


Fig. 3. A graph of tasks in a distributed system, where each node corresponds to a separate problem

At the final stages of training (epochs 120–180), the loss value continued to decrease, reaching 0.2571 at epoch 180. This indicates that the model has achieved high prediction accuracy and is able to effectively take into account both deterministic and probabilistic dependences between tasks.

5. 3. Adaptive task management based on reinforcement learning PPO

Adaptive task management is one of the key stages in the process of optimal task scheduling in heterogeneous distributed information systems. The use of conventional approaches, such as static task assignment using DAG algorithms or standard heuristics, often leads to inefficient resource allocation and increased costs for their use. This is due to the fact that dynamic changes in system load, variability in computing resource performance, and uncertainty in the execution time of individual tasks can significantly affect the efficiency of computing allocation.

To solve these problems, the adaptive task management method based on Proximal Policy Optimization (PPO) is used. This is a deep reinforcement learning (RL) algorithm that makes it possible to train an agent to make optimal decisions in a dynamic environment [21, 22]. Using PPO in the system allows for flexible task distribution management, ensuring optimal resource utilization and reducing makespan (total execution time).

PPO is a development of classical reinforcement learning methods, such as Policy Gradient, and is characterized by high stability and efficiency in large systems. The main advantage of PPO is that it limits the policy update step, which prevents abrupt changes in strategy and ensures smooth learning. This is especially important under conditions of dynamic changes typical of distributed systems, where resource availability and load can change quickly and unpredictably.

The main idea is that the system performs training of an agent, which makes a decision about which task should be performed at the current moment, based on its contextual information. For example, the state of resources and the predicted critical path, which is determined by GAT. This approach makes it possible to render planning not just static but adaptive to changing system operating conditions.

The task management process in PPO is divided into several logical stages. First, the agent's interaction environment is formed. This is a task graph, represented in the DAG + GERT format. In it, each node contains information about the execution time, the probability of moving to the next tasks, the level of resource utilization and priorities. At each point in time, the PPO agent receives information about the current state of the environment and makes decisions about further task execution.

The PPO agent operates in the environment according to the following scheme:

1. Obtaining the state of the environment – the model receives information about the current state of the task distribution in the graph, the workload of computing nodes, and the current makespan.

2. Calculating the probabilities of executing the next tasks – based on the current characteristics, the PPO agent predicts which tasks should be launched next in order to minimize the makespan and ensure uniform load distribution.

3. Making a decision about launching the task – the model chooses which task will be executed next, taking into account the optimal use of resources and possible delays in the network.

4. Updating the policy based on the experience gained – the PPO algorithm analyzes the results of task execution and updates the decision-making strategy based on the feedback received.

The basis of the PPO operation is the maximization of the utility function, which is defined as

$$L^{CLIP}(\theta) = E_t\left[\min\left(r_t(\theta)\hat{A}_t, clip\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t\right)\right], \quad (14)$$

where $r_t(\theta)$ is the ratio of the new policy to the old one;

$\hat{A}_t$ is the advantage estimate for step t;

ε is the parameter that limits the policy update step.

This formula allows the PPO agent to smoothly update its policy, which makes the decision-making process stable and efficient.

One of the main tasks of adaptive control is to optimize the use of resources in real time. The adaptive control process includes several key stages:

1. Determining the state of the system. At each step, the system analyzes the current state, which includes:

– the workload of computing nodes (the agent determines whether the node is overloaded or underloaded);

– the execution time of current tasks (it analyzes how long it will take to complete each task depending on the current load on the system);

– the probability of switching between tasks (according to the GERT graph) (if a certain node has a high communication delay, it can be excluded from the critical path).

2. Agent actions. The agent (PPO algorithm) makes decisions on the distribution of tasks between nodes, taking into account the current state of the system. Actions may include:

– redistribution of tasks between nodes;

– changing task execution priorities;

– choosing alternative execution routes.

3. Evaluation of results. After performing actions, the system evaluates the effectiveness of task distribution based on the following indicators:

– total execution time (makespan);

– uniformity of load distribution;

– number of failures or delays.

4. Policy update. Based on the results, PPO updates the task distribution strategy, maximizing the expected reward, which is defined as a reduction in total execution time and improved load balancing.

Experiments with training the PPO model in a specially designed task distribution environment showed the results given in Table 3.

Table 3

Results of investigating the adaptive task management stage

Iteration	1	2	3	4	5
FPS	1193	714	706	778	768
Time Elapsed	1	5	4	10	13
Total Timesteps	2048	4096	6144	8192	10240
Approx. KL	–	0.01150	0.01284	0.01329	0.01803
Clip Fraction	–	0.179	0.189	0.207	0.229
Clip Range	–	0.2	0.2	0.2	0.2
Entropy Loss	–	–1.09	–1.06	–1.02	–0.966
Explained Variance	–	0.0232	–0.0348	–0.0104	0.31
Learning Rate	–	0.0003	0.0003	0.0003	0.0003
Loss	–	255	129	107	24
N Updates	–	10	20	30	40
Policy Gradient Loss	–	–0.0294	–0.0273	–0.0278	–0.0303
Value Loss	–	631	332	207	71

During training, the model showed high performance in terms of computational speed. The average frames per second (FPS) ranged from 700 to 1200, indicating an efficient implementation of the environment and the learning algorithm on CPU. The total training time was 13 seconds for 10240 steps, demonstrating the model's ability to quickly adapt to the dynamics of the environment.

The training process was characterized by a decrease in the total loss from 255 at the initial stages to 24 after 40 updates. This indicates a steady improvement in the model's performance. In addition, a decrease in the loss associated with the value function was observed from 631 to 71, indicating an improvement in the model's ability to predict rewards.

The approximate Kullback-Leibler divergence metric (approx KL) showed an increase from 0.0118 to 0.0180, indicating an active policy change during training. This is also confirmed by the increase in the clip fraction from 0.179 to 0.229, which is the expected effect when using a clipped gradient in the PPO algorithm.

The policy entropy, which characterizes the level of uncertainty of the model's actions, decreased from –1.09 to –0.966. This indicates that the model becomes more confident in its actions, which is a normal phenomenon during training.

The explained variance metric showed significant fluctuations, changing from 0.0232 to –0.0348, and then increasing to 0.31. This may indicate that the model has not yet reached a stable state in the initial stages of training, but its ability to predict rewards has improved over time.

To assess the effectiveness of the model, a visualization of the load distribution between nodes was performed. The evaluation results are shown in Fig. 4.

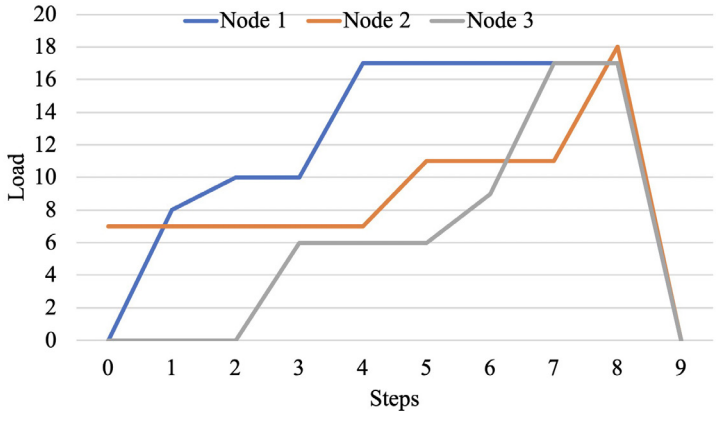


Fig. 4. Evaluation plots of adaptive load distribution between computing nodes using PPO

The results shown in Fig. 4 demonstrate that the model successfully minimizes the load imbalance between computing nodes. At the initial steps (0–3), there is a significant discrepancy: node 1 receives a load of up to 10–17 units, while node 3 remains unloaded for a long time. However, after step 4, the load between nodes gradually equalizes. At the final stage (step 8), all nodes have very similar load values, indicating dynamic balancing. This is visually confirmed by the decrease in the standard deviation of the load between nodes during scheduling. Thus, the model effectively solves the problem of uniform task distribution under variable load conditions.

The experimental results demonstrate that the PPO model is effectively trained in a task distribution environment, reducing the load imbalance between nodes. The reduction in overall losses, improved reward prediction, and reduced policy entropy indicate stable model training. Our results confirm the potential of using PPO for adaptive load control tasks in distributed systems.

5. 4. Optimization of task allocation between resources based on Bayesian Optimization

At the stage of assigning processors to tasks, the task of finding the assignment of tasks to specific computing nodes of the system is performed in such a way as to ensure load balancing, minimizing the total execution time (makespan) and stability of the allocation taking into account the dynamics of the environment.

Given the need for adaptability, efficiency, and the ability to work under uncertainty, the Bayesian Optimization algorithm was chosen to implement task allocation. This approach makes it possible to model the target performance function without its explicit analytical expression and makes optimal decisions based on a limited number of observations.

The main idea of Bayesian optimization is to use an approximation model – usually a Gaussian Process (GP) – to construct an estimate of the loss (or gain) function that corresponds to the task execution time or the degree of resource utilization. At each iteration, the optimizer chooses a new task assignment configuration that has the best expected value based on the current forecast.

Let us formally state the problem of assigning processors to tasks as follows. Let us denote:

– $X \subset R_d$ – space of possible task distributions;

– $f(x)$ – function that returns the makespan value for a given task distribution $x \in X$;

– $p(f|D)$ – prior distribution of function $f(x)$, which is built on the basis of available data $D = \{(x_i, f(x_i))\}_{i=1}^{n}$, where x_i is the task distribution, and $f(x_i)$ is the corresponding makespan value.

In general, the developed algorithm consists of the following 4 steps:

1. Construction of posterior distribution. In this step, using the Gaussian process (GP), the algorithm builds a posterior distribution of function $f(x)$, which takes into account all available data D.

2. Optimization of the expected improvement function (EI). The EI function is defined as

$$EI(x) = E\left[\max\left(f_{best} - f(x), 0\right)\right], \quad (15)$$

where f_{best} is the smallest observed value of makespan at the current time.

EI estimates how much the value of $f(x)$ is expected to improve when choosing a new point x.

3. Choosing a new point. The algorithm chooses a point x^* that maximizes EI

$$x^* = arg\max_{x \in X} EI(x). \quad (16)$$

After choosing x^*, the algorithm estimates the value of $f(x^*)$ using simulation modeling.

4. Data update. A new point $(x^*, f(x^*))$ is added to the set D, and the process is repeated. This iterative approach allows the algorithm to gradually improve the task distribution, reducing the total execution time (makespan) and ensuring a balanced load between the system nodes.

In the context of HDCS, the task distribution depends on a large number of parameters. Among them, we can highlight the performance of processors (the model includes a set of clusters with different characteristics), the waiting time of tasks in queues, the probability of transitions between tasks (defined in the GERT model), the priority of tasks, etc.

Two main types of input data are used for the operation of the BO algorithm:

1. Task matrix *T*. Each task in the matrix *T* is described by a set of characteristics, which include:

– execution time – an estimate of the time required to complete the task on a specific type of resource;

– priority level – the importance of the task, which determines its priority in the system;

– probability of completion – the probability of successful completion of the task, which takes into account possible dependences and risks.

2. Resource matrix *R*. The matrix *R* contains information about the available system resources, such as:

– available processors – a list of computing nodes or processors that can be used to perform tasks;

– resource capacity – the computing power of each resource (for example, in FLOPS);

– current load – the level of resource load at the time of task allocation.

BO iteratively generates task allocation configurations among resources. In the first step of the algorithm, it tests the generated configurations using simulation evaluation or a real environment (for example, based on PPO). After testing, the GP model is updated, which is used to predict the effectiveness of different configurations. Based on the data obtained, BO selects new configurations that maximize the Expected Improvement (*EI*).

The process of assigning processors to tasks in the context of task scheduling in distributed information systems is shown in Fig. 5.

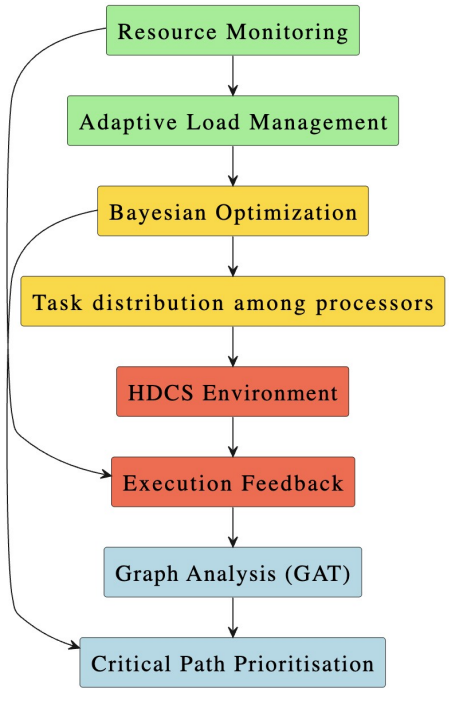


Fig. 5. Diagram of interaction between components of the task planning method

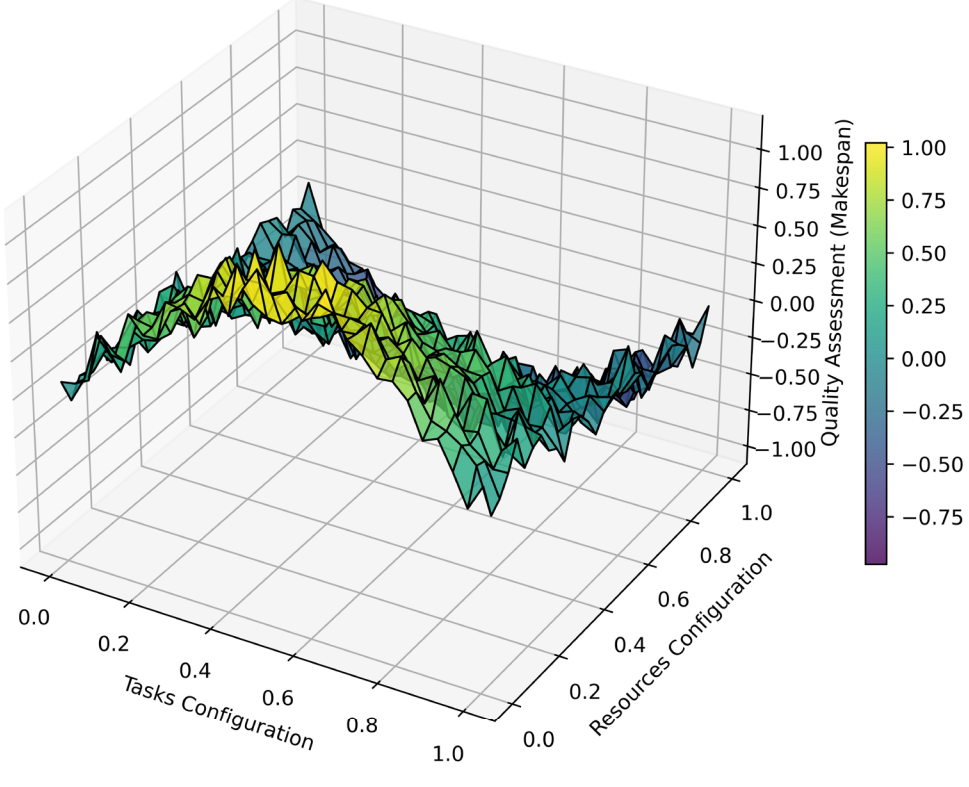


Fig. 6. The loss function surface relative to the task distribution, constructed by GP, with the observation points and the next selected point marked

To evaluate the results of the processor assignment stage, a loss function surface for the task distribution is constructed by GP, with the observation points and the next selected point marked (Fig. 6).

The graph in Fig. 6 shows how the total execution time (makespan) changes depending on different task distribution configurations. The loss function surface constructed by GP demonstrates how the BO algorithm analyzes the space of possible task distributions. This makes it possible to visually assess how the BO method gradually approaches the optimal distribution, minimizing the loss (makespan).

The plot in Fig. 6 shows the points that represent the different task allocation configurations tested by the algorithm. Each point corresponds to a specific allocation of tasks among resources and displays the corresponding makespan value. These points demonstrate how BO iteratively explores the space of possible solutions, collecting data on the performance of different configurations.

The plot in Fig. 6 also demonstrates the point that was selected by the BO algorithm for further testing. This point is selected based on the Expected Improvement (EI), that is, it has the greatest potential for improving the result (reducing the makespan). This shows how BO actively searches for optimal solutions and does not simply randomly iterate over the options.

BO quickly finds areas with low makespan values, which makes it possible to minimize the overall execution time. The method takes into account previous observations (points on the plot) to improve predictions and select new configurations. BO strives for a global optimum, which is evident from the way it concentrates on areas with the lowest loss function values.

5. 5. Investigating effectiveness of the proposed method by testing the performance comparison of the new approach with prototypes

To assess the effectiveness of the proposed method of task scheduling in distributed information systems, a series of experiments were conducted, which included comparison with other popular approaches, such as FCFS, List Scheduling, HEFT, Greedy, and Random [3, 4, 10]. The main goal of the experiments was to assess how much the proposed method improves the overall execution time (makespan) and load balancing between resources.

The experiments were conducted on a distributed systems simulator that simulated the operation of heterogeneous clusters with different processor performance, network delays and dynamic load. Different sets of tasks were used for testing, including both deterministic and probabilistic dependences between tasks (DAG + GERT).

The following metrics were used to compare the effectiveness of the methods:

– total execution time (makespan);

– standard deviation of the load;

– planning time.

The experimental results showed that the proposed method significantly outperforms conventional approaches in all main metrics.

Also, Table 4 gives examples of comparing the effectiveness of different task distribution options (BO, Random, Greedy) with the corresponding makespan and standard deviation of the load.

Table 4 compares the performance of different task scheduling methods: BO, Greedy (greedy algorithm), and Random. Table 4 gives the average execution time (makespan) in seconds and the standard deviation of the workload for each method. The results show that the BO method achieves the best results: the average makespan is 35.2 seconds and the standard deviation of the workload is 0.22. This indicates that BO effectively minimizes the total execution time and ensures an even load distribution among resources. In comparison, the Greedy method demonstrates an average makespan of 43.7 seconds with a standard deviation of 0.31, and the Random method – 51.5 seconds with a standard deviation of 0.47. These results confirm the advantages of using BO for adaptive task scheduling in distributed systems.

Table 4

Examples of comparing the efficiency of different task distribution options (BO, Random, Greedy) with the corresponding makespan and standard deviation of the load

Method	Average Makespan (s)	Standard deviation of load
Bayesian Optimization	35.2	0.22
Greedy	43.7	0.31
Random	51.5	0.47

Despite the complexity of the BO and GAT algorithms, the scheduling time remained acceptable for real-world use, as the method quickly found optimal configurations due to the iterative approach.

The proposed intelligent task scheduling method has a number of important advantages compared to conventional approaches. First, the use of a hybrid DAG + GERT model makes it possible to take into account the stochastic nature of dependences between tasks, which increases the accuracy of critical path prediction in distributed environments. Second, the use of machine learning algorithms, in particular GAT and PPO, provides high adaptability to dynamic changes in load and resource availability. Third, the use of BO at the stage of assigning tasks to resources makes it possible to achieve better balancing of computing nodes and minimize task execution time.

At the same time, the proposed approach has certain disadvantages. First, the use of deep learning and reinforcement learning algorithms requires significant computing resources at the stage of model training. Second, the complexity of the system implementation is higher compared to classical methods, which may complicate its integration into some practical systems. Also, agent training requires some time to achieve a stable decision-making policy.

Thus, our experimental results confirm the effectiveness of the proposed method in dynamic and heterogeneous environments. The use of BO in combination with GAT and PPO makes it possible to achieve significant improvements in both overall execution time and load balancing. This makes the proposed method promising for use in modern distributed systems, where adaptability and efficiency are important.

6. Discussion of results based on the study of the intelligent method of task planning

Our results are explained by the integration of three components: a mathematically formalized DAG + GERT model, artificial intelligence algorithms, and adaptive optimization procedures. Formulas (1), (2) are given, which determine the critical path in the DAG graph and estimate the task execution time taking into account the probabilities

in the GERT graph. Formula (4) makes it possible to estimate the probability of successful task execution on a computing node depending on its power, which is critically important for adapting planning in a heterogeneous environment.

The effectiveness of using GAT to predict the critical path in the constructed graph task model is confirmed by the data in Table 2, which demonstrate a gradual decrease in the loss function (Loss) during the learning process – from 2.2169 to 0.2571 at the 180th epoch. This indicates an improvement in the model's ability to take into account dependences between tasks in the planning process.

The PPO algorithm was used for adaptive task management. That chapter shows that during the training of the model, the loss function decreased from 255 to 24, as well as the policy entropy from –1.09 to –0.966. This indicates that the agent's policy is stabilized, and learning is effective. Additionally, Fig. 4 illustrates how load balancing between nodes occurs at steps 0–8, confirming the dynamic balancing of the system under the learning mode.

Tasks were assigned to resources using BO, which used the expected improvement function (7). A visualization of the solution space and an example of constructing the loss function surface are shown in Fig. 6, which demonstrates how BO finds efficient task configurations.

Table 4 gives comparative results: the BO method achieves an average makespan of 35.2 s and a standard deviation of the load of 0.22. For comparison, Random shows 51.5 s and 0.47, respectively. This clearly demonstrates the advantage of BO over classical allocation options.

The proposed approach to task scheduling in heterogeneous distributed systems, which combines the graph models DAG and GERT with artificial intelligence algorithms, provides better results compared to the basic algorithms FCFS, HEFT, and Greedy, as can be seen from the data in Fig. 2. Unlike the methods described in [10, 12, 17], in which either there is no adaptability or probabilistic dependences between tasks are not taken into account, the proposed method allows for flexible response to changes by combining the graph structure with real-time learning.

The proposed solution resolves the issue of adaptability and efficiency of planning under conditions of stochastic behavior of the system. This is achieved through the flexibility of processing changes in resource parameters and the ability to predict alternative paths to complete tasks, which is especially important in cloud and IoT environments.

At the same time, our study has certain limitations. First, the simulation does not deeply cover energy consumption or failures in data transmission channels. Second, the training of models was carried out under conditions of a limited set of load scenarios, which limits generalizability. Also, the use of GAT and PPO requires significant computational resources at the training stage.

The disadvantages include the complexity of setting hyperparameters, the need for long-term training, as well as the lack of formal proof of theoretical convergence in stochastic conditions. This complicates the verification of the model in critical environments.

Further development of our study is likely to expand the set of environment models, including real energy consumption in the objective function, implementing online training of the PPO agent, and testing on real infrastructures using cloud platforms and edge computing. Also promising is to examine planning stability when changing the task dependence graph in real time.

7. Conclusions

1. The task scheduling models in HDCS were formalized by extending the conventional DAG model by adding GERT graphs. That allowed us to take into account the variability of task execution and probabilistic delays, which is critically important for dynamic environments where resources and workloads can change unpredictably. The proposed model makes it possible to estimate not only deterministic dependences between tasks but also probabilistic factors affecting their execution, which significantly increases the accuracy of planning.

2. A task critical path prediction model with GNN was implemented, which provided up to a 2-fold increase in the accuracy of task execution time estimation and made it possible to better take into account probabilistic dependences between subtasks.

3. Adaptive task scheduling control based on the PPO reinforcement learning algorithm was implemented, which ensured a flexible system response to dynamic changes in workload and availability of computing resources.

4. The task distribution between computing nodes was optimized using Bayesian optimization, which made it possible to achieve better load balancing and minimize the total task execution time (makespan).

5. The effectiveness of the proposed method was studied by testing the performance comparison of the new approach with conventional methods, such as FCFS, List Scheduling, HEFT, Greedy, and Random. Our experimental results showed that the proposed method significantly outperforms conventional approaches in terms of total execution time (makespan) and load balancing. In particular, the average task execution time for the proposed method is 35.2 seconds, which is significantly less compared to 43.7 seconds for Greedy and 51.5 seconds for Random. In addition, the method demonstrates high accuracy in detecting anomalies and adaptability to changes in the environment, which makes it effective for use in highly loaded real-time systems.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

All data are available, either in numerical or graphical form, in the main text of the manuscript.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

References

1. Mikhav, V., Semenov, S., Meleshko, Y., Yakymenko, M., Shulika, Y. (2023). Constructing the mathematical model of a recommender system for decentralized peer-to-peer computer networks. Eastern-European Journal of Enterprise Technologies, 4 (9 (124)), 24–35. https://doi.org/10.15587/1729-4061.2023.286187
2. Meleshko, Y., Raskin, L., Semenov, S., Sira, O. (2019). Methodology of probabilistic analysis of state dynamics of multidimensional semiMarkov dynamic systems. Eastern-European Journal of Enterprise Technologies, 6 (4 (102)), 6–13. https://doi.org/10.15587/1729-4061.2019.184637
3. Rama Krishna, M. S., Mangalampalli, S. (2023). A Systematic Review on Various Task Scheduling Algorithms in Cloud Computing. EAI Endorsed Transactions on Internet of Things, 10. https://doi.org/10.4108/eetiot.4548
4. Sreenath, M., Vijaya, P. A. (2023). Comparative Study of Scheduling Algorithms for Multiprocessor Systems. 2023 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), 713–718. https://doi.org/10.1109/iitcee57236.2023.10091017
5. Pachipala, Y., Sureddy, K. S., Kaitepalli, A. B. S. S., Pagadala, N., Nalabothu, S. S., Iniganti, M. (2024). Optimizing Task Scheduling in Cloud Computing: An Enhanced Shortest Job First Algorithm. Procedia Computer Science, 233, 604–613. https://doi.org/10.1016/j.procs.2024.03.250
6. Semenov, S., Lymarenko, V., Yenhalychev, S., Gavrilenko, S. (2022). The Data Dissemination Planning Tasks Process Model Into Account the Entities Differentity. 2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT), 1–6. https://doi.org/10.1109/dessert58054.2022.10018695
7. Sinnen, O. (2006). Task Scheduling for Parallel Systems. John Wiley & Sons. https://doi.org/10.1002/0470121173
8. Semenov, S., Mozhaiev, O., Kuchuk, N., Mozhaiev, M., Tiulieniev, S., Gnusov, Y. et al. (2022). Devising a procedure for defining the general criteria of abnormal behavior of a computer system based on the improved criterion of uniformity of input data samples. Eastern-European Journal of Enterprise Technologies, 6 (4 (120)), 40–49. https://doi.org/10.15587/1729-4061.2022.269128
9. Semenov, S., Liqiang, Z., Weiling, C. (2020). Penetration Testing Process Mathematical Model. 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T), 142–146. https://doi.org/10.1109/picst51311.2020.9468039
10. Jayswal, A. K., Lobiyal, D. K. (2022). A Comparative Study of Task Scheduling Metaheuristic Algorithms in Cloud Computing. 2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 118–123. https://doi.org/10.1109/confluence52989.2022.9734189
11. Pereira, I., Madureira, A., Costa e Silva, E., Abraham, A. (2021). A Hybrid Metaheuristics Parameter Tuning Approach for Scheduling through Racing and Case-Based Reasoning. Applied Sciences, 11 (8), 3325. https://doi.org/10.3390/app11083325
12. Parizad, A., Hatziadoniu, C. (2022). Deep Learning Algorithms and Parallel Distributed Computing Techniques for High-Resolution Load Forecasting Applying Hyperparameter Optimization. IEEE Systems Journal, 16 (3), 3758–3769. https://doi.org/10.1109/jsyst.2021.3130080
13. Savita, K., Gaurav, S., Bhawna, S. (2023). Hybrid Machine Learning Model for Load Prediction in Cloud Environment. International Journal of Performability Engineering, 19 (8), 507. https://doi.org/10.23940/ijpe.23.08.p3.507515
14. Fang, Z., Ma, T., Huang, J., Niu, Z., Yang, F. (2025). Efficient Task Allocation in Multi-Agent Systems Using Reinforcement Learning and Genetic Algorithm. Applied Sciences, 15 (4), 1905. https://doi.org/10.3390/app15041905
15. Liu, Z., Guo, M., Bao, W., Li, Z. (2024). Fast and Adaptive Multi-Agent Planning under Collaborative Temporal Logic Tasks via Poset Products. Research, 7. https://doi.org/10.34133/research.0337
16. Kumar, H., Tyagi, I. (2020). Hybrid model for tasks scheduling in distributed real time system. Journal of Ambient Intelligence and Humanized Computing, 12 (2), 2881–2903. https://doi.org/10.1007/s12652-020-02445-6
17. Yanamandram Kuppuraju, S., Sankaran, P., Patil, S. (2025). Hybrid Task Scheduling Using Genetic Algorithms and Machine Learning for Improved Cloud Efficiency. International Journal For Multidisciplinary Research, 7 (2). https://doi.org/10.36948/ijfmr.2025.v07i02.39380
18. Torres-Toledano, J. G., Sucar, L. E. (1998). Bayesian Networks for Reliability Analysis of Complex Systems. Progress in Artificial Intelligence – IBERAMIA 98, 195–206. https://doi.org/10.1007/3-540-49795-1_17
19. Attar, S. F., Mohammadi, M., Pasandideh, S. H. R. (2025). A Bayesian network approach to production decisions by incorporating complex causal factors. Journal of Management Science and Engineering, 10 (2), 262–278. https://doi.org/10.1016/j.jmse.2025.03.002
20. Huang, M.-C. (2024). A Sender-Initiated Fuzzy Logic Contrnol Method for Network Load Balancing. Journal of Computer and Communications, 12 (08), 110–122. https://doi.org/10.4236/jcc.2024.128007
21. Semenov, S., Zhang, L., Cao, W., Bulba, S., Babenko, V., Davydov, V. (2021). Development of a fuzzy GERT-model for investigating common software vulnerabilities. Eastern-European Journal of Enterprise Technologies, 6 (2 (114)), 6–18. https://doi.org/10.15587/1729-4061.2021.243715
22. Moskalenko, V., Kharchenko, V., Semenov, S. (2024). Model and Method for Providing Resilience to Resource-Constrained AI-System. Sensors, 24 (18), 5951. https://doi.org/10.3390/s24185951