UDC 004.7 **doi:** https://doi.org/10.20998/2522-9052.2025.4.11

Nina Kuchuk[1], Oleksandr Zakovorotnyi[1], Yuliia Andrusenko[2], Viacheslav Radchenko[2], Dmytro Lysytsia[1]

[1] National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine
[2] Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

LOAD BALANCING OF A MULTIPROCESSOR COMPUTER SYSTEM USING THE METHOD PARTICLE SWARM OPTIMIZATION

Abstract. The relevance of this research is determined by the increasing demands on the performance of multiprocessor computer systems, which are widely used for processing large-scale data and solving complex computational tasks. Uneven load distribution among processors often leads to resource underutilization, overload of certain nodes, and, consequently, a decrease in overall system efficiency. **The subject of the study** is the process of load balancing in multiprocessor computer systems using metaheuristic optimization methods. **The purpose of the work** is to develop and analyze a mathematical model of load balancing based on the Particle Swarm Optimization (PSO) method, aimed at improving system performance and resource utilization efficiency. **The paper presents** a mathematical model of the optimization process for task distribution across processors, considering their performance and current workload. The results of simulation experiments confirm a reduction in the average execution time of computational tasks and an improvement in load uniformity when applying PSO, compared to traditional approaches. The conclusions highlight that the use of PSO is an effective and feasible solution to the load balancing problem in multiprocessor computer systems. The proposed approach can be applied in cloud infrastructures, distributed environments, and high-performance computing systems, where efficient resource allocation is a critical requirement.

Keywords: optimization; computer system; load balancing; particle swarm optimization (PSO); multiprocessor system; mathematical model.

Introduction

Problem relevance. Modern computer systems and cloud infrastructures are characterized by high levels of complexity and dynamic computing processes [1–3]. In such environments, the load on system nodes constantly fluctuates due to varying user request rates, uneven resource utilization, and unexpected peak loads [4–6]. Without an effective balancing mechanism, some servers are overloaded while others remain underutilized [7]. This leads to a number of critical issues [8]:

– decreased system performance due to delays in processing tasks;

– increased power consumption due to inefficient resource utilization;

– risk of service failure due to overloaded individual nodes;

– decreased system scalability and flexibility;

– degraded quality of service (QoS).

Load balancing is a complex optimization problem. It involves distributing computing resources to minimize system response time, maximize hardware utilization, and maintain the stability of the entire infrastructure [9].

In cloud environments, this problem becomes even more pressing due to the dynamic migration of virtual machines, the heterogeneity of tasks, and the limited resources of physical servers [10, 11]. Therefore, the development and implementation of effective load balancing algorithms, particularly those based on heuristic and metaheuristic approaches, is key to ensuring the reliability and efficiency of modern computer systems. Traditional methods are not always able to find optimal or even acceptable solutions within a reasonable timeframe [12–16]. Therefore, metaheuristic approaches are becoming increasingly popular. For example, the particle swarm optimization (PSO) method, which models the collective behavior of biological systems, is used [17, 18].

PSO, proposed by Kennedy and Eberhardt in 1995, remains one of the most popular metaheuristic approaches to solving optimization problems. In recent decades, it has gained widespread adoption across various industries, including engineering, machine learning, signal processing, telecommunications, and cloud computing.

Literature review. The literature review considered hybridization with other methods. A significant number of works are devoted to combining PSO with genetic algorithms, differential evolution and machine learning methods to improve accuracy and convergence speed [19, 20].

To avoid premature convergence and falling into local minima, strategies are being developed for adaptively changing the inertial weight parameters and the coefficients of the cognitive and social components [21

Researchers point out that traditional PSO has limited effectiveness in solving problems with a large number of local extrema, so modifications are being created that focus on finding a set of solutions [22].

PSO is increasingly used for load balancing in cloud computing and virtualized environments, which improves performance and resource efficiency [23].

Modern research shows the high performance of PSO in the tasks of optimizing neural networks, tuning deep learning parameters, and controlling autonomous cyber-physical systems [24].

A literature review has shown that the particle swarm method remains relevant due to its ease of implementation, versatility, and scalability. Algorithm improvements are aimed at increasing robustness to local minima, adapting to dynamic environments, and applying it to computationally complex problems.

Therefore, **the goal of this article** is to study PSO in solving the balancing problem in a computer system.

To this end, we propose solving the following problems: load balancing problem in a multiprocessor system; virtual machine placement problem in a cloud infrastructure.

1. Particle Swarm Optimization algorithm

To clearly illustrate the operating principle of the particle swarm optimization (PSO), a generalized flowchart of the algorithm is provided (Fig. 1).

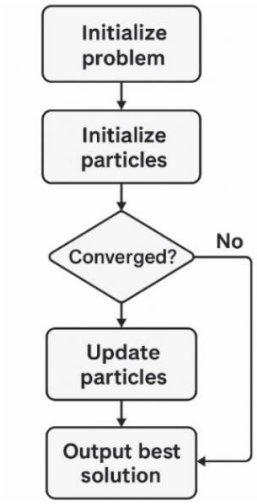

Fig. 1. Block diagram of the PSO

It reflects the main stages of computation: from problem initialization and the formation of the initial particle population to the process of iteratively updating their positions. Verification of the convergence condition determines the completion of the algorithm, after which the initial solution is formed. This structure enables an adaptive search for the optimal task distribution, oriented both to the individual experience of each particle and to the global knowledge of the entire population.

Let's consider the main stages of the algorithm.

1. At the initial stage, the task is initialized and the objective function is determined, which in the case of load balancing reflects the maximum execution time among all processors.

2. Next, an initial population of particles is created, where each of them encodes a certain variant of task distribution.

3. At each iteration, the algorithm checks whether the convergence of the solution has been achieved or another stopping condition has been met. If not, then the positions of the particles are updated taking into account both their individual best experience and the globally optimal solution.

4. After the iterations are completed, the best solution found is selected and displayed, which ensures a more even load distribution between the processors.

The essence of the PSO method is to use a set of agents (particles) that move in a multidimensional search space. Each particle represents a possible solution to the problem and has its own speed and position. Each particle remembers the best solution it found (pbest). The particles are oriented to the best solution found by the entire swarm (gbest). The movement of particles occurs according to a formula that takes into account their previous speed, the distance to the personally found best solution and the globally best. Thus, the method balances between exploitation (searching near already known solutions) and exploration (exploring new areas of space).

The PSO is an effective tool for finding rational solutions in complex multidimensional spaces. Table 1 shows the results of a theoretical study of the advantages and disadvantages of using this method to solve the load balancing problem in a multiprocessor system.

Table 1 – **Advantages and disadvantages of using PSO in load balancing**

Aspect	Advantages	Disadvantages
Simplicity	Ease of implement-tation and minimal number of parame-ters to configure	Dependence on the correct choice of parameters (inertial weight, cognitive and social coefficients)
Speed	High speed of convergence to acceptable solutions	Possibility of premature convergence to local minima
Scalability	Efficient operation in different environments - from small systems to cloud computing	Increasing computational costs with increasing dimensionality of the problem
Flexibility	Can be used for different criteria (latency, energy consumption, bandwidth)	Complexity in multi-criteria problems without modifications
Resilience	Less chance of getting "stuck" in local extrema due to collective search	Stochastic nature leads to different results with multiple runs
Dynamism	Ability to adapt to load changes in real time	Need for reconfiguration when the environment changes quickly
Extensibility	Easily combined with other evolutionary and heuristic methods	Lack of mathematical guarantee of global optimality

2. Mathematical model PSO

In the context of optimization problems, a swarm is represented by a set of particles, where each particle corresponds to a possible solution to the problem.

A particle is characterized by two main parameters:

• current position in the multidimensional search space $x_i(t)$;

• speed of movement $v_i(t)$, which determines the direction and intensity of position change.

The particle state is updated according to the following equations [25]:

– speed update:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot \left(p_{best\_i} - x_i(t)\right) + c_2 \cdot r_2 \cdot \left(g_{best\_i} - x_i(t)\right), \quad (1)$$

where w – inertia coefficient that adjusts the effect of previous speed; c_1, c_2 – learning coefficients (cognitive and social components, respectively); r_1, r_2 – random numbers uniformly distributed in the interval [0; 1]; p_{best_i} – best position of particle iii for the entire search time; g_{best_i} – best global position among the entire swarm;

– position update:

$$x_i(t+1) = x_i(t) + (t+1) \cdot v_i(t+1), \quad (2)$$

Thus, the trajectory of each particle is formed under the influence of its own experience and the experience of the entire population. This allows the swarm to balance local search (exploitation of known solutions) and global search (exploration of new areas of space).

Analysis of the advantages and disadvantages of the particle swarm method shows that this approach is one of the most flexible and effective tools for load balancing problems, especially in dynamic and scalable computing environments. Its strengths are simplicity of implementation, high convergence speed and the ability to adapt to various objective functions. At the same time, the method has a number of limitations associated with the risk of premature convergence, parameter dependence and stochastic nature of the results. Thus, the use of PSO is appropriate in cases where it is necessary to quickly find an approximate rational solution, however, to increase the reliability and stability of the results, the use of modified or hybrid versions of the algorithm is often recommended.

3. The problem of load balancing in a multiprocessor system

Consider a computer system with multiple processors (or servers) (Fig. 2), each of which can perform a certain amount of tasks [26]. The goal is to distribute incoming tasks among the processors in such a way as to minimize the execution time (make span) and avoid overloading individual nodes.

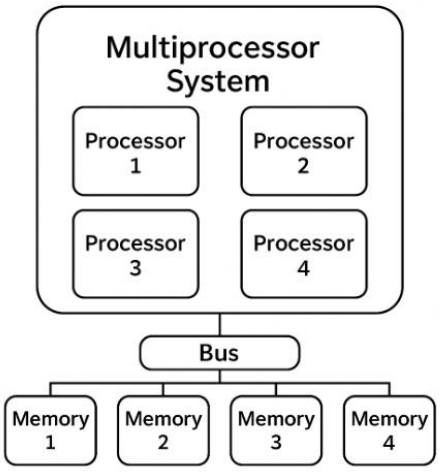

Fig. 2. Computer system with multiple processors

To create a mathematical model of the problem, it must be formalized.

Let us have m processors and n tasks.

Let us denote the execution time of the j-th task on the i-th processor as T_{ij}.

We need to find the following mapping of "tasks → processors" to minimize the maximum load:

$$\min\left(\max_{i\in 1..m}\sum_{j\in J_i}^{n} T_{ij}\right), \quad (2)$$

where J_i – set of tasks assigned to a processor i.

Solution coding. In this model, each particle is a vector of length n, where the element of the vector x_j corresponds to the number of the processor assigned to the j-th task. For example, the vector [2, 1, 3, 2, 1] corresponds to this distribution:

tasks 1 → processor 2;
tasks 2 → processor 1;
tasks 3 → processor 3;
tasks 4 → processor 2;
tasks 5 → processor 1.

Fitness function. The quality of the solution is evaluated by the maximum load among all processors. The smaller this value, the more balanced the task distribution.

Swarm update. Particles change their distribution, focusing on both their own best experience and the globally optimal solution found by the entire population.

Example of work. Consider a system with three processors and 5 tasks with durations: T = [10, 7, 6, 5, 8].

The initial distribution may be of the form:

P1 = [10, 5] → 15;
P2 = [7, 6] → 13;
P3 = [8] → 8,

where the maximum load is 15.

In the process of several PSO iterations, it is possible to find other options, for example:

P1 = [10] → 10;
P2 = [7, 8] → 15;
P3 = [6, 5] → 11,

which again gives the maximum value 15.

The optimal distribution in this case will be as follows Fig. 3:

P1 = [10, 5] → 15,
P2 = [7, 8] → 15,
P3 = [6] → 6,

which provides an even balance between processors with the same maximum of 15, but with a reduction in the disparity between the loads.

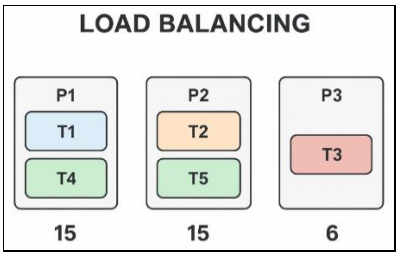

Fig. 3. Optimal load distribution

In more complex cases with dozens of processors and hundreds of tasks, the particle swarm optimization method shows high efficiency, as it quickly finds a near-optimal balance.

Thus, PSO allows you to automatically search for the optimal distribution of tasks in the system, which reduces execution time and increases the efficiency of resource use.

The rapid development of cloud computing and the growth of the number of users necessitates the effective management of computing resources. One of the key tasks in cloud infrastructure is the optimal placement of

virtual machines (VMs) on physical servers, taking into account the limited resources of the processor, RAM, disk space and network bandwidth [27, 28]. Inefficient distribution leads to overload of individual nodes, reduced system performance, increased energy consumption and reduced reliability of services [29, 30].

Optimization methods, in particular heuristic and meta-heuristic approaches, are widely used to solve the VM placement problem. Among them, the Particle Swarm Optimization (PSO) method occupies a special place due to its ability to quickly find solutions close to optimal in complex multidimensional spaces. PSO imitates the collective behavior of a swarm, where each "particle" is a candidate solution, and the interaction between them allows the system to adaptively balance the speed of convergence and the quality of the obtained results [31].

Applying PSO to the problem of virtual machine placement allows you to reduce the energy consumption of data centers, avoid server overloads, and ensure a more even distribution of resources. This makes the method an effective tool for supporting the scalability and reliability of cloud services. Thus, the study of the use of the particle swarm method in the context of optimizing VM placement is relevant and has significant scientific and practical interest.

Fig. 4 shows a flowchart of the operation of the particle swarm method (PSO) for placing virtual machines in a cloud infrastructure.

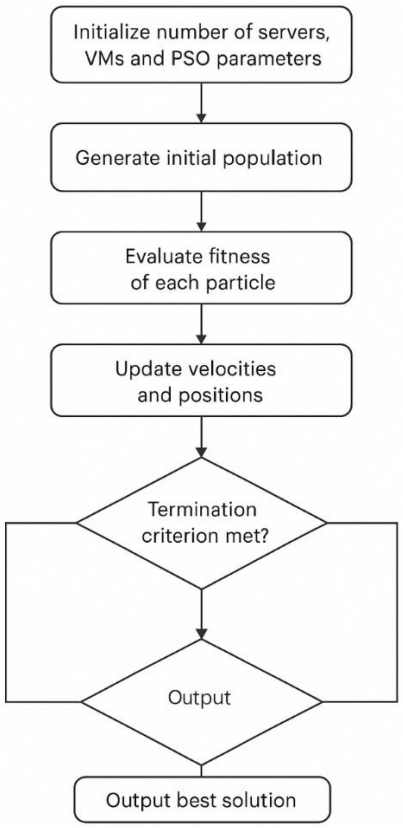

Fig. 4. Algorithm of the PSO method for placing virtual machines in a cloud infrastructure

The algorithm begins with the initialization of the number of physical servers, virtual machines, and PSO parameters. Next, an initial population of particles is formed, each of which encodes a possible option for distributing VMs between servers.

The next step is to evaluate the fitness function for each particle, taking into account server overload, resource utilization balance, and energy consumption. After that, the particle velocities and positions are updated according to the best individual and global solutions. The algorithm continues the iterative process until the termination criterion is met (reaching the maximum number of iterations or swarm convergence). As a result, an initial solution is formed that corresponds to the optimal or close to optimal distribution of virtual machines within the available server resources.

Problem statement for placing virtual machines in a cloud infrastructure. A typical cloud infrastructure is considered, consisting of a set of physical servers (hosts) and a set of virtual machines (VMs), each of which has its own resource requirements:

– $H = \{h_1, \ldots, h_m\}$ – a set of physical servers; each server h has limited resources: CPU, memory, disk, network bandwidth, which we will denote by a capacity vector $C_h = (C_h^{cpu}, C_h^{mem}, \ldots)$;

– $V = \{v_1, \ldots, v_n\}$ – multiple virtual machines; each VM v requires resources $r_v = (r_v^{cpu}, r_v^{mem}, \ldots)$;

– the placement is specified by binary variables $x_{v,h} \in \{0, 1\}$: $x_{v,h} = 1$ if VM v hosted on the server h, else $x_{v,h} = 0$;

– for each VM:

$$\sum_{h \in H} x_{v,h} = 1; \tag{3}$$

– capacity constraints: for each server h and each resource type r:

$$\sum_{v \in V} r_v^{(r)} \cdot x_{v,h} \le C_h^{(r)}. \tag{4}$$

Objective function and optimization criteria. The placement task has several goals at once:

1. ***Minimize server overload***. One option is to minimize the maximum CPU load (total time/load):

$$\min \max L_h^{cpu}, \quad L_h^{cpu} = \sum_{v \in V} r_v^{cpu} \cdot x_{v,h}. \tag{5}$$

2. ***Balancing resource usage*** (CPU, memory). This can be formalized as minimizing the variance/variation of load between servers or minimizing the sum of deviations from the mean:

$$\min \left(\begin{array}{l} \sum_{h \in H} \left(L_h^{cpu} - \bar{L}_h^{cpu} \right)^2 + \\ + \alpha \sum_{h \in H} \left(L_h^{cmem} - \bar{L}_h^{mem} \right)^2 \end{array} \right), \tag{6}$$

where $\bar{L}$ — average load and α— weighting factor for memory.

3. ***Minimizing energy consumption***. One approach – minimize the number of active servers

$$A = \sum_{h \in H} y_h, \quad y_h \in \{0, 1\}. \tag{7}$$

shows whether the server is enabled and to connect y_h with $x_{v,h}$ using a condition

$$\sum_v x_{v,h} \le M \cdot y_h \quad (8)$$

(for big M). Then the energy component of the goal can be, for example:

$$\min \begin{pmatrix} E = \beta \cdot \sum_{h \in H} P_h^{idle} \cdot y_h + \\ +\gamma \cdot \sum_{h \in H} f(L_h) \end{pmatrix}, \quad (9)$$

where P_h^{idle} is base capacity of an active but partially loaded server; $f(L, h)$ is additional power from the load; β, γ is weights.

The trade-off between objectives can be implemented as a multi-criteria optimization or an aggregated objective function:

$$\min \begin{pmatrix} w_1 \cdot max\_load + w_2 \cdot us\_balance + \\ +w_3 \cdot min\_energy \end{pmatrix}, \quad (10)$$

where w_i – вагові коефіцієнти, що відображають пріоритети оператора хмарної платформи.

Notes on practical implementation:

• taking into account the discreteness of the distribution and combinatorial complexity, the problem is NP-hard, so for large instances heuristic and metaheuristic methods are used (for example, PSO, GA, Tabu Search, Simulated Annealing);

• when designing an algorithm, it is important to consider adaptability to dynamic changes (appearance/ disappearance of VMs, variable loads) and the overhead of VM migration (movement cost);

• solution evaluation metrics should include not only the time characteristics of the load, but also the costs of migration and QoS restoration time.

Formalization of the problem of placing virtual machines in a cloud infrastructure

In cloud data centers, it is important to ensure a rational distribution of virtual machines (VMs) between physical servers (hosts) [32]. Each server has a limited number of resources, in particular, processing power (CPU), random access memory (RAM), disk memory, and network bandwidth [33]. The problem of optimal VM placement can be presented in the form of combinatorial optimization. Позначення:

$H = \{h_1, h_2, \ldots, h_m\}$ – multiple physical servers;

$V = \{v1, v2, \ldots, vn\}$ – multiple virtual machines;

$C_h = (C_h^{cpu}, C_h^{mem}, \ldots)$ – server resources h;

$R_v = (r_v^{cpu}, r_v^{mem}, \ldots)$ – resources needed VM v;

$x_{v,h} \in \{0, 1\}$ – location variable, equal to 1 if VM v is located on server h;

$y_h \in \{0, 1\}$ – server activity variable: $y_h = 1$ if the server is in use.

Limitation. Each virtual machine must be hosted on the same server:

$$\sum_{h \in H} x_{v,h} = 1, \quad \forall v \in V. \quad (11)$$

The load on the server should not exceed its resources:

$$\sum_{v \in V} r_v^{cpu} \le C_h^{cpu} \cdot y_h, \quad \forall h \in H. \quad (12)$$

$$\sum_{v \in V} r_v^{mem} \le C_h^{mem} \cdot y_h, \quad \forall h \in H. \quad (13)$$

Objective function. The goal is to simultaneously achieve load balance and reduce energy consumption. For this, a combined function is used:

$$\begin{pmatrix} F = w_1 \cdot \max_{h \in H} \left(L_h^{cpu} \right) + w_2 \cdot \sum_{h \in H} y_h + \\ +w_3 \cdot \sum_{h \in H} \left(L_h^{cpu} - \bar{L}^{cpu} \right)^2 \end{pmatrix} \to \min, \quad (14)$$

where $L_h^{cpu} = \sum_{v \in V} r_v^{cpu} x_{v,h}$ – server CPU load; L_h^{cpu} – average load across all servers; $\sum_{h \in H} y_h$ – number of active servers (proportional to power consumption); w_1, w_2, w_3 – weighting factors for balancing goals.

Thus, the model allows to minimize the overload of individual nodes, achieve more uniform resource utilization, and reduce energy consumption by turning off inactive servers.

Conclusions

The particle swarm optimization method is an effective metaheuristic optimization technique that simulates the collective behavior of biological systems. It allows to find acceptable and close to optimal solutions in problems with a large number of variables and complex constraints. In the load balancing problem, PSO showed the ability to quickly find a uniform distribution of tasks or virtual machines between system resources, minimizing the overload of individual nodes.

Unlike classical methods, PSO does not require derivatives of the objective function, which makes it universal for various types of optimization problems - from computer systems to cloud computing.

In the example with a cloud environment, the PSO algorithm allowed to balance the use of resources (CPU, memory), as well as reduce the number of active servers, which can directly reduce the energy consumption of data centers. Despite its advantages, the particle swarm optimization method has some disadvantages: the possibility of getting stuck in local minima and the dependence on the choice of parameters (w,c1,c2). Therefore, it is often combined with other optimization methods to improve accuracy.

In general, PSO can be considered a promising tool for resource management problems in computer systems and cloud platforms, as it provides good convergence, scalability, and ease of implementation.

REFERENCES

1. Shi, Q. and Zhao, F. (2024), "Research on Computer Cloud Intelligent System Based on Intelligent Virtualization Technology", *2024 IEEE 3rd Int. Conf. on Eebda 2024*", pp. 1245–1250, doi: https://doi.org/10.1109/EEBDA60612.2024.10485750
2. Mao, C. (2023), "Design of Computer Storage System Based on Cloud Computing", *Lecture Notes in Electrical Engineering*, 1037 LNEE, pp. 651–659, doi: https://doi.org/10.1007/978-981-99-1983-3_59
3. Kuchuk, H., Mozhaiev, O., Kuchuk, N., Tiulieniev, S., Mozhaiev, M., Gnusov, Y., Tsuranov, M., Bykova, T., Klivets, S. and Kuleshov, A. (2024), "Devising a method for the virtual clustering of the Internet of Things edge environment", *Eastern-*

European Journal of Enterprise Technologies, vol. 1, no. 9 (127), pp. 60–71, doi: https://doi.org/10.15587/1729-4061.2024.298431
4. Taneja, M. and Davy, A. (2017), "Resource aware placement of IoT application modules in fog-cloud computing paradigm", *Proc. 2017 IFIP/IEEE Symposium on Integrated Network and Service Management*, INSM, pp. 1222–1228, doi: https://doi.org/10.23919/INM.2017.7987464
5. Semenov, S., Mozhaiev, O., Kuchuk, N., Mozhaiev, M., Tiulieniev, S., Gnusov, Yu., Yevstrat, D.,Chyrva, Y. and Kuchuk, H. (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*, vol. 6 (4, 120), pp. 40–49, doi: https://doi.org/10.15587/1729-4061.2022.269128
6. Lee, B.M. (2025), "Efficient Resource Management for Massive MIMO in High-Density Massive IoT Networks", *IEEE Transactions on Mobile Computing*, vol. 24 (3), pp. 1963–1980, doi: https://doi.org/10.1109/TMC.2024.3486712
7. Rajammal, K. and Chinnadurai, M. (2025), "Dynamic load balancing in cloud computing using predictive graph networks and adaptive neural scheduling", *Scientific Report*s, vol. 15(1), 22181, doi: https://doi.org/10.1038/s41598-025-97494-2
8. Kuchuk, N., Mozhaiev, O., Semenov, S., Haichenko, A., Kuchuk, H., Tiulieniev, S., Mozhaiev, M., Davydov, V., Brusakova, O. and Gnusov, Y. (2023), "Devising a method for balancing the load on a territorially distributed foggy environment", *Eastern-European Journal of Enterprise Technologies*, vol. 1(4, 121), pp. 48–55, doi: https://doi.org/10.15587/1729-4061.2023.274177
9. Farahi, R. (2025), "A comprehensive overview of load balancing methods in software-defined networks", *Discover Internet of Things*, vol. 5(1), 6, doi: https://doi.org/10.1007/s43926-025-00098-5
10. Kuchuk, H., Husieva, Y., Novoselov, S., Lysytsia, D. and Krykhovetskyi, H. (2025), "Load Balancing of the layers Iot Fog-Cloud support network", *Advanced Information Systems*, vol. 9, no. 1, pp. 91–98, doi: https://doi.org/10.20998/2522-9052.2025.1.11
11. Kaistha, T. and Ahuja, K. (2025), "Cloud Heterogeneous Networks: Cooperative Random Spatial Local Best Particle Swarm Optimization for Load Balancing", IJMEMS, vol. 10(5), pp. 1585–1603, doi: https://doi.org/10.33889/IJMEMS.2025.10.5.075
12. Ibrahimov, B., Hashimov, E. and Ismayılov, T. (2024), "Research and analysis mathematical model of the demodulator for assessing the indicators noise immunity telecommunication systems", *Advanced Information Systems*, vol. 8, no. 4, pp. 20–25, doi: https://doi.org/10.20998/2522-9052.2024.4.03
13. Hunko, M., Tkachov, V., Kuchuk, H., Kovalenko, A. (2023), "Advantages of Fog Computing: A Comparative Analysis with Cloud Computing for Enhanced Edge Computing Capabilities", *2023 IEEE 4th KhPI Week on Advanced Technology*, *KhPI Week 2023 – Conf. Proc.*, 02-06 October 2023, Code 194480, doi: https://doi.org/10.1109/KhPIWeek61412.2023.10312948
14. Raj, S.P., Kalpana, D., Arun, M., Manthena, K. V., Krishna, K.S. and Krishnan, V.G. (2025), "Performance Optimization in Wireless Sensor Networks using REAMR Protocol for IoT Applications", *2025 International Conference on Emerging Smart Computing and Informatics Esci 2025*, doi: https://doi.org/10.1109/ESCI63694.2025.10988270
15. Kuchuk, H., Kalinin, Y., Dotsenko, N., Chumachenko, I. and Pakhomov, Y. (2024), "Decomposition of integrated high-density IoT data flow", *Advanced Information Systems*, vol. 8, no. 3, pp. 77–84, doi: https://doi.org/10.20998/2522-9052.2024.3.09
16. Kalinin, Y., Kozhushko, A., Rebrov, O., and Zakovorotniy, A. (2022), "Characteristics of Rational Classifications in Game-Theoretic Algorithms of Pattern Recognition for Unmanned Vehicles", *2022 IEEE 3rd Khpi Week on Advanced Technology Khpi Week 2022 Conference Proceedings*, 03-07 October 2022, doi: https://doi.org/10.1109/KhPIWeek57572.2022.9916454
17. Li, Z. and Xiong, J. (2024), "Reactive Power Optimization in Distribution Networks of New Power Systems Based on Multi-Objective Particle Swarm Optimization", *Energies*, vol. 17(10), 2316, doi: https://doi.org/10.3390/en17102316
18. Sajith, P.J. and Nagarajan, G. (2022), "Intrusion Detection System Using Deep Belief Network & Particle Swarm Optimization", *Wireless Personal Communications*, vol. 125(2), pp. 1385–1403, doi: https://doi.org/10.1007/s11277-022-09609-x
19. Clerc, M. (2006), *Particle Swarm Optimization*, ISTE Press, 244 p., available at: https://www.wiley.com/en-us/Particle+Swarm+Optimization-p-9781905209040
20. Fan, Y.-A. and Liang, C.-K. (2022), "Hybrid Discrete Particle Swarm Optimization Algorithm with Genetic Operators for Target Coverage Problem in Directional Wireless Sensor Networks", *Applied Sciences Switzerland*, vol. 12(17), 8503, doi: https://doi.org/10.3390/app12178503
21. Alam, M.S., Lusaf, M.J., Munna, K.Y.A., Ushno, J.A. and Hasan, M.D.T. (2023), "An elitist particle swarm optimization with mutation operator and adaptive inertia weight for global optimization", *Aip Conference Proceedings*, 2788(1), 040008, doi: https://doi.org/10.1063/5.0148658
22. Sun, B., Wang, Y., He, L. and Chen, Y. (2025), "Niche particle swarm optimization algorithm with integrated adaptive weibull flight", *Proceedings of the 2025 5th International Conference on Applied Mathematics Modelling and Intelligent Computing Cammic 2025*, pp. 149–153, doi: https://doi.org/10.1145/3745533.3745558
23. Handur, V. S., Deshpande S. L. and Marakumb P. R. (2021), "Particle swarm optimization for load balancing in cloud computing: A survey", *Turkish Journal of Computer and Mathematics Education*, vol. 12(15), pp. 257–265, available at: https://turcomat.org/index.php/turkbilmat/article/view/1766
24. Capel, M.I., Holgado-Terriza, J.A., Galiana-Velasco, S. and Salguero, A.G. (2024), "A Distributed Particle Swarm Optimization Algorithm Based on Apache Spark for Asynchronous Parallel Training of Deep Neural Networks", *ACM International Conference Proceeding Series*, pp. 76–85, doi: https://doi.org/10.1145/3677333.3678158
25. Chen, A.C.H. (2024), "Exploring the Optimized Hyperparameter Values Based on Mathematical Models for PSO", *Proceedings of International Conference on Circuit Power and Computing Technologies*, ICCPCT 2024, pp. 431–433, doi: https://doi.org/10.1109/ICCPCT61902.2024.10673319
26. Kuchuk, N., Kashkevich, S., Radchenko, V., Andrusenko, Y. and Kuchuk, H. (2024), "Applying edge computing in the execution IoT operative transactions", *Advanced Information Systems*, vol. 8, no. 4, pp. 49–59, doi: https://doi.org/10.20998/2522-9052.2024.4.07
27. Petrovska, I., Kuchuk, H. and Mozhaiev, M. (2022), "Features of the distribution of computing resources in cloud systems", *2022 IEEE 4th KhPI Week on Advanced Technology*, KhPI Week 2022 – Conference Proceedings, 03-07 October 2022, Code 183771, doi: https://doi.org/10.1109/KhPIWeek57572.2022.9916459
28. Dun B., Zakovorotnyi, O. and Kuchuk, N. (2023), "Generating currency exchange rate data based on Quant-Gan model", *Advanced Information Systems*, vol. 7, no. 2, pp. 68–74, doi: http://dx.doi.org/10.20998/2522-9052.2023.2.10

29. Niu, Y.-F., Yan, Y.-F. and Xu, X.-Z. (2025), “A new MC-based method for the resource-constrained multi-distribution multi-state flow network reliability optimization problem”, RESS, 265, 111499, doi: https://doi.org/10.1016/j.ress.2025.111499
30. Mozhaev, O., Kuchuk, H., Kuchuk, N., Mykhailo, M. and Lohvynenko, M. (2017), “Multiservice network security metric”, *2nd International Conference on Advanced Information and Communication Technologies*, AICT 2017 – Proceedings, pp. 133–136, doi**:** https://doi.org/10.1109/AIACT.2017.8020083
31. Li, J., Ma, Z., Wang, J., Song, S. and Wang, R. (2025), “Application Analysis of PSO Wavelet Neural Network in Power System Energy Balance Problem”, *2025 4th International Conference on Energy Power and Electrical Technology*, ICEPET 2025, pp. 819–824, doi: https://doi.org/10.1109/ICEPET65469.2025.11047229
32. Kuchuk, G., Nechausov, S. and Kharchenko, V. (2015), “Two-stage optimization of resource allocation for hybrid cloud data store”, *International Conference on Information and Digital Technologies*, Zilina, Slovakia, pp. 266–271, doi: http://dx.doi.org/10.1109/DT.2015.7222982
33. Mageed, K.A., Hieba, A.A. and Amer, G.M. (2024), “Optimal Network Reconfiguration with DG-Allocation Using Parallel-PSO Technique”, *2024 Int. Telecommunications Conference ITC Egypt*, pp. 445–450, doi: http://dx.doi.org/10.1109/ITC-Egypt61547.2024.10620573

Received (Надійшла) 28.05.2025
Accepted for publication (Прийнята до друку) 20.08.2025

ВІДОМОСТІ ПРО АВТОРІВ/ ABOUT THE AUTHORS

Кучук Ніна Георгіївна – доктор технічних наук, професор, професорка кафедри обчислювальної техніки та програмування, Національний технічний університет “Харківський політехнічний інститут”, Харків, Україна;
Nina Kuchuk – Doctor of Technical Sciences, Professor, Professor of Computer Engineering and Programming Department, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: nina_kuchuk@ukr.net; ORCID Author ID: http://orcid.org/0000-0002-0784-1465;
Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=57196006131.

Заковоротний Олександр Юрійович – доктор технічних наук, професор, завідувач кафедри комп’ютерної інженерії та програмування, Національний технічний університет “Харківський політехнічний інститут”, Харків, Україна;
Oleksandr Zakovorotnyi – Doctor of Technical Sciences, Professor, Head of Computer Engineering and Programming Department, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: Oleksandr.Zakovorotnyi@khpi.edu.ua; ORCID ID: http://orcid.org/0000- 0003-4415-838X;
Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=57201613700.

Радченко В'ячеслав Олексійович – старший викладач кафедри Електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Viacheslav Radchenko – Senior Lecturer of Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: viacheslav.radchenko@nure.ua; ORCID Author ID: https://orcid.org/0000-0001-5782-1932;
Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=57189376280.

Андрусенко Юлія Олександрівна – асистент кафедри Електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Yuliia Andrusenko – Assistant Professor of Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: yuliia.andrusenko@nure.ua; ORCID Author ID: https://orcid.org/0000-0001-7844-2042.

Лисиця Дмитро Олександрович – кандидат технічних наук, доцент кафедри комп’ютерної інженерії та програмування, Національний технічний університет “Харківський політехнічний інститут”, Харків, Україна;
Dmytro Lysytsia – Candidate of Technical Sciences, Associate Professor of Computer Engineering and Programming Department, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: Dmytro.Lysytsia@khpi.edu.ua; ORCID Author ID: https://orcid.org/0000-0003-1778-4676;
Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=57220049627.

Балансування навантаження багатопроцесорної КС з використанням методу рою частинок

Н. Г. Кучук, О. Ю. Заковоротний, Ю. О. Андрусенко, В. О. Радченко, Д. О. Лисиця

Анотація. Актуальність дослідження зумовлена зростаючими вимогами до продуктивності багатопроцесорних комп’ютерних систем, що використовуються для обробки великих обсягів даних і виконання складних обчислювальних завдань. Нерівномірний розподіл навантаження між процесорами призводить до простою частини ресурсів, перевантаження інших і, як наслідок, до зниження ефективності системи. **Предметом дослідження** є процес балансування навантаження у багатопроцесорних комп’ютерних системах з використанням метаевристичних методів оптимізації. **Метою роботи** є розробка та аналіз математичної моделі балансування навантаження із застосуванням методу рою частинок (Particle Swarm Optimization, PSO), що дозволяє підвищити продуктивність системи та ефективність використання її ресурсів. У роботі наведено математичну модель процесу оптимізації розподілу завдань між процесорами з урахуванням їхньої продуктивності та поточного завантаження. Результати моделювання підтверджують зменшення середнього часу виконання обчислень і підвищення рівномірності завантаження процесорів при використанні методу рою частинок у порівнянні з традиційними підходами. **У висновках** відзначено, що застосування PSO є доцільним для вирішення задач балансування навантаження у багатопроцесорних системах. Запропонований підхід може знайти застосування в хмарних інфраструктурах, розподілених та високопродуктивних обчислювальних середовищах, де критично важливим є оптимальний розподіл ресурсів.

Ключові слова: оптимізація; комп’ютерна система; балансування навантаження; метод рою частинок (PSO); багатопроцесорна система; математична модель.