A New Resource Allocation Method Based on PSO in Cloud Computing

Davoud Bahrepour[a], Nastaran Evaznia[b*], Tahereh Khodabakhshi[c]
Department of Computer, Mashhad Branch, Islamic Azad University, Mashhad, Iran; bahrepour@mshdiau.ac.ir[a], nastaran_avaznia@yahoo.com[b], ta.khodabakhshi@gmail.com[c]

ABSTRACT

Cloud computing has emerged as a pivotal technology for managing and processing data, with a primary objective to offer efficient resource access while minimizing expenses. The allocation of resources is a critical aspect that can significantly reduce costs. This process necessitates the continuous assessment of the current status of each resource to design algorithms that optimize allocation and enhance overall system performance. Numerous algorithms have been developed to address the challenge of resource allocation, yet many fail to satisfy requirements of time efficiency and load balancing in cloud computing environments. This paper introduces a novel approach that classifies tasks according to their resource demands, employs a modified particle swarm optimization (PSO) algorithm, and incorporates load balancing strategies. The proposed method initially clusters tasks based on their resource requirements, subsequently utilizes the PSO algorithm to determine the best task-to-resource assignments, and finally implements a load balancing algorithm to reduce costs through balanced load distribution. The validity of the proposed method is tested and simulated using the Cloudsim tool. The simulation results indicate that the proposed method achieves lower average response time, waiting times, and energy consumption than existing baseline methods.

Keywords— cloud computing, resource allocation, modified particle swarm optimization, response time, energy consumption.

1. Introduction

The swift advancement of distributed computing technology has facilitated inexpensive and robust access to computational resources through the Internet, surpassing the capabilities of previous eras [1]. This advancement has culminated in the model known as cloud computing, wherein users can lease and subsequently release resources such as processors and storage devices as a service through simple Internet requests [1, 2]. Consequently, these systems enable users to utilize resources over a network. Cloud computing is a burgeoning technology that has garnered the interest of researchers across diverse fields due to its benefits [3, 4]. Within the cloud framework, proficiency in optimally managing resources directly correlates with effectiveness in cost reduction and productivity enhancement [4]. Cloud computing platforms allow users to access desired resources over the Internet [5]. This technology has been particularly beneficial for numerous companies, as it allows them to procure necessary resources at a low cost precisely when needed [5, 6]. The advent of cloud computing has spurred rapid growth in various research domains, including virtualization, distributed systems, clustering, and grid computing [7, 8]. Virtualization technology within cloud data centers has enabled the scheduling of user requests on a reduced number of physical machines, thereby enhancing resource efficiency in these data centers [9-11]. Upon receiving requests from cloud users, service providers employ scheduling algorithms to allocate these incoming requests [10]. Nonetheless, a novel challenge in this milieu pertains to the optimal allocation of resources to requests and tasks [12, 13]. Indeed, task scheduling within a cloud environment is a critical and pivotal concern, as end users expect to access resources anytime and anywhere. Given the diversity of resources offered by cloud data centers and the real-time, demand-driven nature of the cloud computing model, the issue of resource allocation is paramount. The variability and frequency of requests may lead to inefficiencies in resource allocation [13]. In essence, resource allocation can be economically defined as mapping suitable processors, memory, and bandwidth [12, 14]. This approach entails the efficient mapping of tasks to available resources to minimize costs and execution times [15-18].

http://dx.doi.org/10.22133/ijwr.2024.457539.1216

Citation D. Bahrepour, N. Evaznia, T. Khodabakhshi, "A New Resource Allocation Method Based on PSO in Cloud Computing", *International Journal of Web Research*, vol.7, no.2,pp.13-21, 2024, doi: http://dx.doi.org/10.22133/ijwr.2024.457539.1216.

**Coressponding Author*
Article History: Received: 14 December 2023; Revised: 25 February 2024; Accepted: 13 March 2024.

Consequently, focusing on the allocation process can contribute to cost reduction.

The primary contributions of this study are as follows:

- We classify tasks according to CPU, memory, and I/O demands.
- We employ a Modified Particle Swarm Optimization (MPSO) algorithm that takes into account a multi-criteria objective function.
- We utilize a load balancing algorithm among hosts to prevent both overloading and underloading of resources, thereby aiming to reduce energy consumption.

The paper is structured as follows: The second section provides a review of the relevant background literature. The third section offers an overview of the PSO algorithm. The proposed method is described in detail in the fourth section. The fifth section presents and discusses the simulation results, and the final section concludes the paper.

2. Related Works

The principal challenge in cloud data centers is the scarcity of resources, making the maximization of resource utilization and the minimization of execution time critical concerns. Various allocation algorithms have been proposed to address this issue. Asif et al. [19] applied the Whale Optimization Algorithm (WOA) for task scheduling, utilizing an Integer Linear Programming (ILP) model to compute fitness. The Cloudsim simulation results indicate a decrease in cost and execution time. Reference [20] introduces an adaptive resource allocation technique for load balancing in fog-cloud environments, employing hybrid multi-criteria decision-making methods such as the Fuzzy Analytic Hierarchy Process (FAHP) and Fuzzy Technique for Order Performance by Similarity to Ideal Solution (FTOPSIS). This technique targets resource allocation in fog-cloud environments, considering the limited resources of fog devices and the need for low latency and rapid response times. The findings suggest that the proposed adaptive multi-criteria-based load balancing technique effectively enhances load balancing, response time, resource utilization, and energy efficiency in fog-cloud environments. Study [21] explores novel algorithms that use real-time resource monitoring, predictive analytics, and adaptive decision-making for intelligent workload allocation in cloud computing. The primary objective is to increase resource utilization, minimize response time, and improve overall system performance. Traditional load balancing methods often fall short of addressing the dynamic and heterogeneous nature of cloud environments. Consequently, this study investigates new algorithms that utilize real-time resource monitoring, predictive analytics, and adaptive decision-making for intelligent workload allocation. The research encompasses the design, implementation, and evaluation of multiple load balancing algorithms, examining their impact on energy consumption and environmental sustainability. Efficient load balancing algorithms have been developed to align resource allocation with energy consumption patterns, contributing to reduced energy usage. Manikandan et al. [22] proposed a hybrid whale optimization method for task scheduling in cloud data centers, focusing on multiple objectives to maximize host utilization and lifespan. The algorithm results in a reduction in execution and completion times. Mangalampalli et al. [23] introduced a multi-objective confidence level approach, using the whale algorithm to model the scheduling problem. The results revealed improvement in execution time, reliability, and energy consumption. Aruna et al. [24] employed an enhanced Firefly algorithm with a multi-criteria objective function for cloud center scheduling, aiming to achieve load balancing and reduce task execution times. Li et al. [25] presented a PSO approach for allocating hardware resources and assigning real-time computing tasks to virtual machines, to minimize system energy consumption. This involves partitioning hardware resources into equal parts and utilizing virtual machine technology to create distinct virtual machines that consume a portion of hardware resources. The PSO method developed in this paper for resource allocation and task assignment to virtual machines demonstrates reduced energy consumption and improved efficiency through simulation results. Beegom et al [26] proposed a multi-objective Integer-PSO algorithm that maintains the algorithm's randomness and eliminates duplicate assignments by modifying the particle position update equation. The resulting charts indicate improvements in the presented method. Beegom et al. [27] introduced a scheduling method according to non-dominated sorting and a PSO approach, using a graph to represent task relationships. This algorithm aims to reduce completion time and cost, with simulation results illustrating improvements in these metrics compared to other algorithms. Saad et al. [28] proposed a hybrid GA-PSO algorithm for optimizing multi-objective tasks in fog computing, achieving notable improvements in execution time, response time, and completion time over traditional methods. The combined approach leverages the strengths of GA and PSO for effective exploration and exploitation of the search space, with results indicating improvements in execution time, response, and completion. Manikandan et al. [29] presented a solution with k-means clustering, black widow algorithms, and fish swarm optimization for resource scheduling , aiming to reduce costs and energy consumption. Clustering is used to categorize requests, while the black widow algorithm and fish swarm optimization are employed for request-to-resource assignment. The simulation results demonstrate reductions in time, cost, and energy consumption.

3. Particle Swarm Optimization algorithm

Particle swarm optimization (PSO) is employed as an optimization technique to identify suitable or multiple solutions [30]. In this framework, particles are initially assigned a velocity based on which they navigate through the solution space, and results are determined by a fitness function. Subsequently, particles are directed towards the optimal point characterized by a more suitable and superior fitness function. This algorithm is inspired by the social behavior of animals. Initially, the algorithm is initiated with a randomly generated group of particles, randomly, and then, which then update their solution as better options are discovered. Within this structure, each position updates its position using the two values that represent the best solutions for that particular parameter. The first value is the position that the particle has reached so far, known as Pbest. The second-best value is derived by the population of particles and is denoted by Gbest. In this algorithm, each particle has a velocity *v*, which is determined by the velocity vector (Pbest, Gbest), and its position is updated using Equ(1-2) [30].

$$S_i(t+1) = ws_i\,(t) + c_1r_1\big(pbest - y_i(t)\big) + c_2r_2(gbest - y_i(t))) \qquad (1)$$

$$y_i(t+1) = y_i(t) + v_i(t+1) \qquad (2)$$

Equ(1) calculates the future velocity of a particle, w is the inertia coefficient, an input weight that influences the particle's previous velocity in the new calculations. A higher value of this parameter increases the global search, while a lower value enhances the local search. Additionally, c1 and c2 are constants greater than zero, with c1 representing the best personal experience and c2 the best social experience. r1 and r2 are random numbers within the range [0, 1].

Equ(2) adds the current position of the particle to its future velocity to determine its future position.

4. Proposed Method

The solution proposed in this paper is a method designed to reduce costs, including response time and waiting time, while preventing an increase in energy consumption through load balancing. This is achieved by categorizing tasks based on their resource requirements and assigning them to suitable hosts using the Modified Particle Swarm Optimization (MPSO) algorithm. Initially, tasks are classified according to their need for processor, memory, and I/O resources. Following this categorization, the MPSO algorithm is applied with a defined objective function, and then the results are used as input for the subsequent step. The next step involves load balancing to prevent energy consumption from rising by balancing the load and avoiding both overloading and underloading of resources. Figure 1 illustrates the flowchart of the proposed method.

Figure 1 outlines the flowchart and steps of the method, considering multiple sources. In this method, tasks are initially divided into three categories based on their resource demands: processor-intensive, memory-intensive, and I/O- intensive. Subsequently, MPSO is employed to map tasks to virtual machines following the objective function. The method for determining task type is based on Equ(3) [31].

$$T_{ih} = \max(C.O.M) = \left(\frac{C_j}{C_s}.\frac{O_j}{O_S}.\frac{M_j}{M_s}\right) \qquad (3)$$

According to Equ(3), Cj, Oj, and Mj represent the resources required by the task, while Cs, Os, and Ms denote the system capacity. If Tjh = C, the task is categorized as processor-intensive; if Tjh = O, it is I/O-intensive; and if Tjh=M, it is memory-intensive. After tasks are categorized, resources are mapped based on the MPSO and the specified objective function. Figure 2 depicts the process of the proposed solution.

Figure 2, categorized tasks are randomly assigned to virtual machines. For instance, if there are 30 tasks, a matrix with 30 columns is created for the position, in which positions are randomly selected, representing resources or virtual machines. A matrix with 30 columns for velocity is then generated, filled with random numbers between zero and one. The fitness function is calculated and stored, with the best value obtained for the objective function saved as the

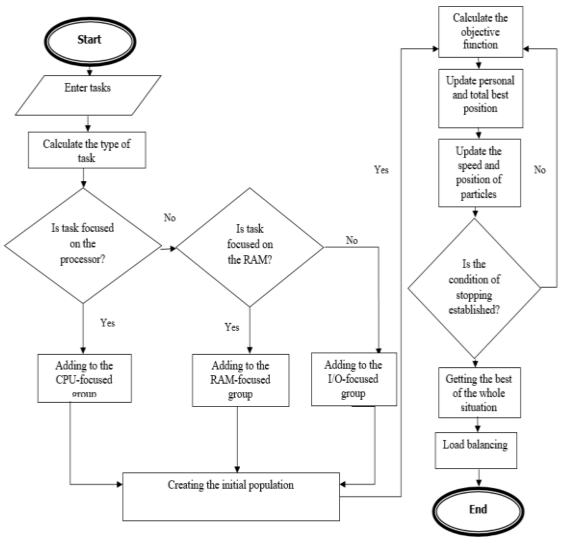
Start
Enter tasks
Calculate the type of task
Is task focused on the processor?
No
Yes
Is task focused on the RAM?
No
Yes
Adding to the CPU-focused group
Adding to the RAM-focused group
Adding to the I/O-focused group
Creating the initial population
Yes
Calculate the objective function
Update personal and total best position
Update the speed and position of particles
No
Is the condition of stopping established?
Getting the best of the whole situation
Load balancing
End

Figure. 1. Flowchart of the proposed method.

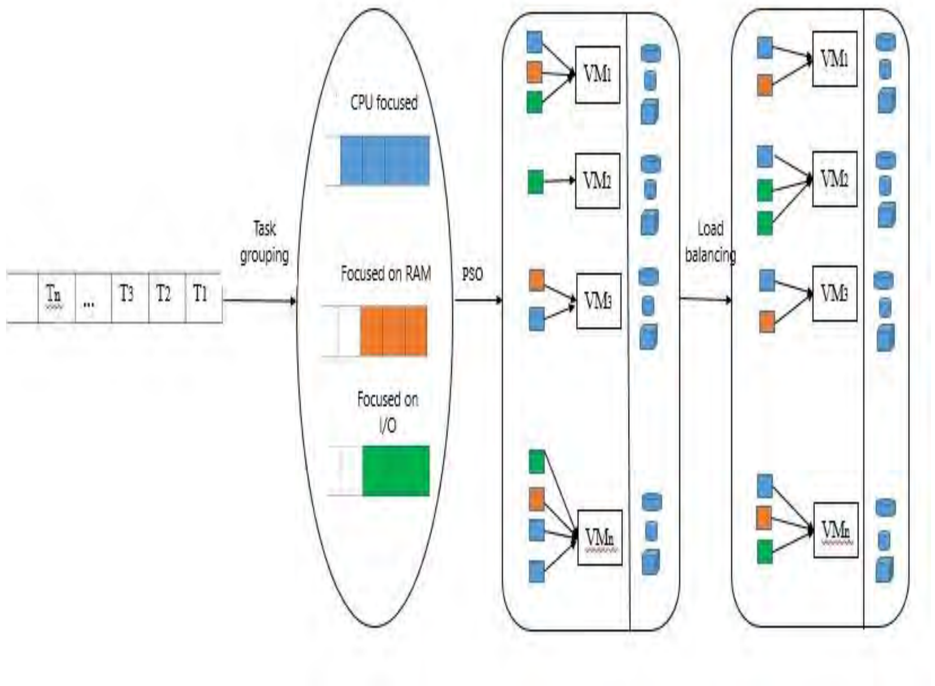

Figure. 2. Process of the proposed method.

best state, along with the location where the best state is achieved. In each iteration, the velocity and position of each particle are updated, and the value of the objective function is calculated at the new position. This value is compared with the particle's best state, and if an improvement is detected, the best state is updated. It is then compared with the overall best state, and if it surpasses the current overall best, the function value in the overall best state is also updated. The algorithm repeats for a predetermined number of iterations, and the best overall state is returned as the solution. Subsequently, a load balancing algorithm is used to prevent hosts from becoming overloaded or underloaded, thereby contributing to cost reduction, including energy consumption.

4.1. Objective Function Calculation

In the MPSO, the movement of particles is governed by the objective function, which dictates how particles search for the optimal point. The value of the objective function can be either minimized or maximized based on the requirements; for instance, if the function represents profit or efficiency, maximization is the goal, whereas, for cost, time, or error, minimization is sought. In the proposed method, our main goal is to reduce the primary objective is to minimize the average response time and waiting time while preventing an increase in energy consumption.

In this paper, the calculation of the objective function considers several factors. Initially, a variable *d* is defined and set to zero. For each task, the virtual machine on which it is executed and its group affiliation are determined. If the task belongs to the processor-intensive group, *d* is calculated using Equ(4).

$$d = d + \frac{c_j}{c_i} \qquad (4)$$

Equ(4) defines c_j as the processing requirement of task$_j$ and c_i as the processing powers of the host *i* where the task is executed.

For tasks in the memory-intensive group, *d* is computed using Equ(5), where m_j is the memory requirement of task$_\mathrm{j}$ and m_j is the memory capacity of host i.

$$d = d + \frac{m_j}{m_i} \qquad (5)$$

If the task is I/O-intensive, d is calculated using Equ(6), with $\mathrm{o_j}$ representing the I/O requirement of task$_\mathrm{j}$ and $\mathrm{o_i}$ the I/O capacity of host *i*.

$$d = d + \frac{o_j}{o_i} \qquad (6)$$

The expectation is that d will decrease; for processor-intensive tasks, a lower d indicates assignment to a host with greater processing power; for memory-intensive tasks, a smaller d suggests allocation to a host with more memory; and for I/O-intensive tasks, a reduced d means assignment to a machine with higher I/O capabilities.

Equ(7) computes the average *d* across all tasks, where *n* is the number of tasks. The fitness function is the sum of the average response time and average *d*, aiming for a decrease in both. Equ(8) is used to calculate the fitness function.

$$z = \frac{d}{2 * n} \quad (7)$$

$$f = z + Average\ Response\ Time \quad (8)$$

Once the fitness function is established, the MPSO algorithm is executed, yielding the best state or the optimal arrangement. This optimal arrangement is stored in a matrix called the mapping matrix, whose dimensions correspond to the number of tasks and components of those virtual machines. For example, if there are five tasks and two resources, the mapping matrix can be Table 1.

Table 1 indicates that tasks $Task_0$ and $Task_3$ are mapped to Vm_0, while $Task_1$, $Task_2$, and $Task_4$ are mapped to Vm_1.

4.2. Load Balancing

Following the objective function described in Section 4.1, the particle optimization algorithm is first executed, and the best mappings are stored in the mapping matrix. The mapping matrix, which is the output of the previous step, is then utilized as the input for the current step. Using the information from the mapping matrix, a separate matrix for virtual machines (VMs) is constructed, with its elements representing the percentage usage for each VM. To calculate this value for each VM, the total processing requirements of all tasks mapped to a particular VM are summed and then divided by the processing capacity of that VM. For instance, using the mapping matrix in Table 1, to determine the processor usage for virtual machine number zero, one would add the processing requirements of tasks Task0 and Task 3 and divide the sum by the processor power of virtual machine number zero.

To achieve load balancing across the VMs, the VM with the highest load and the one with the lowest load are identified. A task is then randomly selected from the overloaded VM and transferred to the underloaded VM. This process is repeated, with tasks being moved randomly from overloaded to underloaded VMs, until the mapping matrix undergoes regular adjustments to reach a balanced state.

5. Evaluation of the proposed method

To validate the proposed method against the approaches presented in previous studies [24] and [29] the Cloudsim simulator [32], a standard cloud simulator, is employed. Cloudsim is a free tool and library that offers various classes for simulating cloud environments. It provides essential classes for defining data centers, including physical machines, virtual machines, tasks, and methods and policies for managing different system components. Version 3.0.3 of the Cloudsim Tools is utilized to simulate the proposed method and compare it with the methods from earlier papers. The implementations are carried out on a Windows 10 system equipped with a core i7 processor and 8 GB of memory.

The dataset consists of a set of randomly generated tasks with varying specifications as outlined in Table 2[31].

Table 2 details the task specifications based on length (MIPS), file size (MB), and output size (MB).

The characteristics of virtual machines, including memory, processor power, and bandwidth, are randomly selected from Table 3 [24].

Table 3 presents the virtual machine specifications based on bandwidth (gigabit per second), memory (MB), and processor (MIPS).

The parameters of the particle optimization algorithm used in the simulation are listed in Table 4.

5.1. Test metrics

Three evaluation indicators are used to assess tests. The first is the average response time, or completion time, which is a key metric for evaluating the proposed approach. The second criterion is the waiting time, and the third is energy consumption,

Table 1. Mapping Matrix

$Task_0$	$Task_1$	$Task_2$	$Task_3$	$Task_4$
Vm_0	Vm_1	Vm_1	Vm_0	Vm_1

Table 2. Task specifications

Parameter	***Value***
Length (CPU)	(500-1000) MIPS
File size	(300-1000) MB
Output size	(20-50) MB

Table 3. Virtual machine specifications

Bandwidth (Gbit/s)	***RAM (MB)***	***CPU (MIPS)***
1	128	512
1	256	1024
1	512	2048
1	1024	4096
1	2048	8192

Table 4. Particle swarm algorithm parameters

Parameter	*Value*
W	9
C1	2
C2	2
Number of repetitions	50
Number of particles	100

which verifies the load balance of resources. Effective load balancing, by preventing overloading and underloading of hosts, contributes to reducing energy consumption.

- **Response time**

The purpose of the proposed method is to minimize costs, including response time and waiting time, while also preventing an increase in energy consumption through load balancing.

Response time is defined as the interval from when a task is submitted to when the result is returned. This time is formulated in Equ(9) [31].

$$RT = \frac{\sum(wt_j + ct_j)}{N} \tag{9}$$

Equ(9) defines the response time (RT) as the sum of the the waiting time for task j (wt_j) and the completion time (ct_j). The average response time is calculated using Equ(9), where N is the number of tasks.

- **Energy consumption**

Reducing energy consumption can be effective in cost reduction. Balancing the load among hosts through optimal allocation and preventing overloading and underloading can lead to energy savings. The energy metric is described by Equ(10) [29].

$$E = \int_{stime}^{etime} P(u(t))dt \tag{10}$$

Equ(10) calculates energy consumption based on the efficiency of the host processor, which varies over time. The energy consumed is represented as an integral between the start time (stime) and the end time (etime), with processor efficiency as a function of time, denoted as u(t).

5.2. Results

We conducted a comparative analysis of the proposed method with the approaches described in papers [24] and [29], as depicted in the tested graphs. The methods under study did not emphasize task classification for enhanced mapping through a multi-criteria objective function. Additionally, they did not address the establishment of load balancing to prevent host overloading and underloading, which could lead to better performance in energy consumption reduction.

In the first test, the number of resources was set to a constant value of 10, while the number of tasks varied from 200 to 350 in increments of 25. In the graphs, the term BWFSO (Black-widow and Fish Swarm optimization) is used to represent paper [29], and FA (Firefly Algorithm) is used for [24]. The response time was calculated, and the evaluation results are presented in Figure 3.

As illustrated in Figure 3, as the number of tasks increases, the average response time also increases. However, the proposed method demonstrates a reduction in average response time by 27% compared to the FA [24] and by 22% compared to the BWFSO [29]. In the second evaluation, the number of tasks is held constant at 300, while the number of virtual machines varies from 5 to 11 in increments of 1. This improvement is attributed to the consideration of the objective function and the load balancing parameter. Figure 4 shows a graph comparing the average response time with an increasing number of resources.

According to Figure 4, as resources increase, the response time decreases. In this scenario, the proposed method achieves a 36% decrease in the average response time compared to the FA [24] and a 25% decrease compared to the BWFSO [29]. By utilizing the defined objective function and implementing load balancing, we were able to effectively reduce and improve the response time. Next, we examined the average waiting time for tasks, is examined, with the results illustrated in Figure 5 and 6.

As shown in Figure 5, the waiting time in the proposed method decreases relative to basic papers as the number of resources increases. The proposed method performs 34% better than the FA [24] and 23% better than the BWFSO [29]. In the subsequent analysis, the number of tasks was increased while the number of resources remained constant. The results are depicted in Figure 6.

In Figure 6, the proposed method exhibits a decrease in average response time by 29% compared to the FA [24] and by 24% compared to the BWFSO [29]. Finally, Figure 7 and Figure. 8Figure 8, examine the energy consumption.

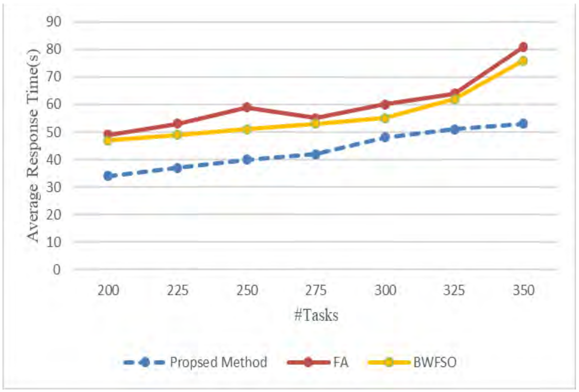

Figure. 3. Comparison graph of average response time with increasing tasks.

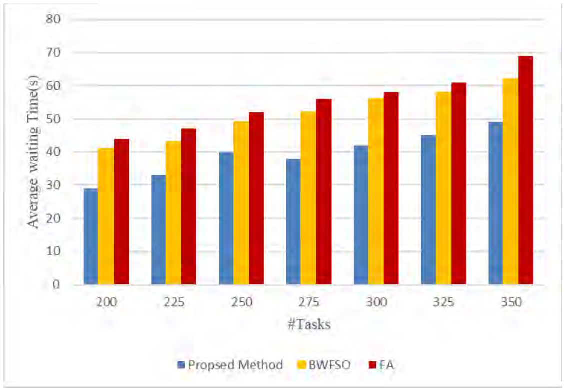

Figure. 6. Comparison graph of average waiting time with increasing tasks.

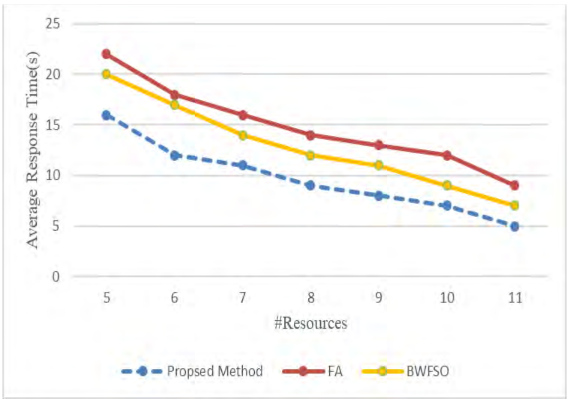

Figure. 4. Comparison graph of average response time with increasing resources.

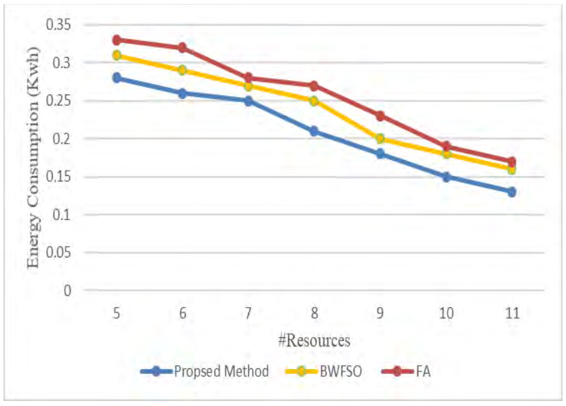

Figure. 7. Comparison graphs of energy consumption with increasing resources.

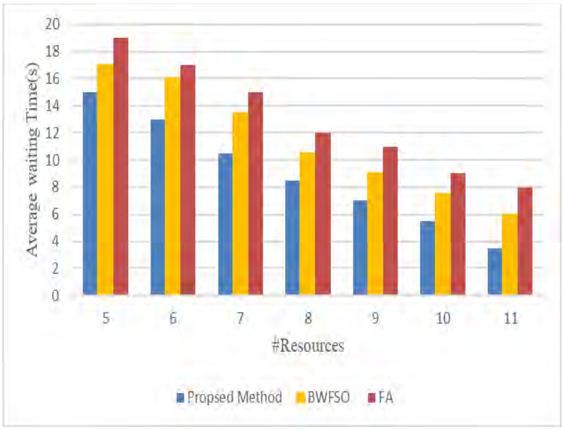

Figure. 5. Comparison chart of average waiting time with increasing resources.

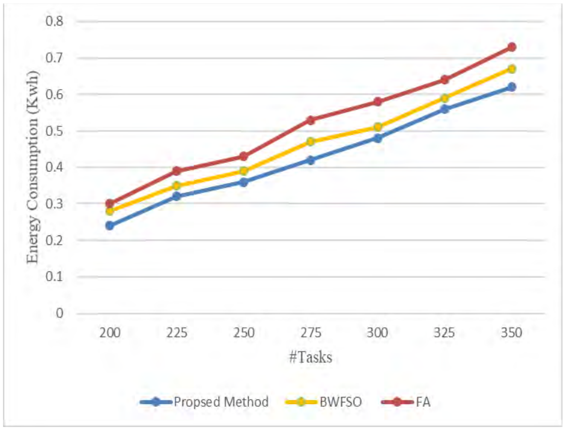

Figure. 8. Comparison graphs of energy consumption with increasing tasks.

In Figure 7 and 8, the energy consumption of the proposed method is lower than that of the two comparison methods. This reduction is due to the addition of a load balancing step, which creates balanced loads on the hosts and prevents overloading and underloading. The proposed method shows a decrease of 19% compared to the first paper in Figure. 7 and 13% compared to the second paper. In Figure 8, the improvements are 17% and 9%, respectively.

6. Conclusion

The vast and dynamic nature of the cloud infrastructure, coupled with the ever-changing demands of user requests, makes effective resource management and scheduling critical in this domain. Allocation and scheduling methods must be adaptable and swift, responding to changes or increased workloads to sustain system efficiency. This paper introduced a new method for task allocation in cloud environments. The method is comprised of three steps: initially, tasks are categorized according to their resource requirements. Subsequently, the MPSO algorithm is employed with a multi-criteria objective function to assign each task to a suitable host. Finally, load balancing is achieved by migrating tasks away from overloaded and underloaded hosts, ensuring a balanced system load that helps curb energy consumption. Simulation results obtained using the Cloudsim simulator and comparisons with related studies demonstrate that the proposed method significantly reduces average response and waiting times, both with an increasing number of tasks and resources, while also minimizing energy consumption.

Future research will explore the integration of quality of service parameters and cost factors, which were not addressed in this study. Additionally, the proposed method will be combined with other meta-heuristic algorithms, and the potential of fog computing to reduce latency for users will be investigated.

Declarations

Funding

The authors did not receive support from any organization for the submitted work.

Conflict of interest

The authors have no conflict of interest.

References

[1] M. M. E. Mahmoud, J. J. P. C. Rodrigues, K. Saleem, J. Al-Muhtadi, N. Kumar, and V. Korotaev, "Towards energy-aware fog-enabled cloud of things for healthcare," *Comput. Electr. Eng.*, vol. 67, pp.58–69, 2018. https://doi.org/10.1016/j.compeleceng.2018.02.047.

[2] M. S. A. Khan and R. Santhosh, "Task scheduling in cloud computing using hybrid optimization algorithm," *Soft Comput.*, vol. 26, no. 23, pp. 13069–13079, 2022. https://doi.org/10.1007/s00500-021-06488-5.

[3] N. S. S. Fatemi, M. T. Zahmatkesh, and D. Bahrepour, "Energy Efficiency and Establishing Service Level Agreement using Fuzzification of Virtual Machine Selection Policies for Migrating in Cloud Computing," *2023 9th International Conference on Web Research (ICWR)*, Tehran, Iran, Islamic Republic of, pp. 201-207, 2023. https://doi.org/10.1109/ICWR57742.2023.10138982.

[4] N. Evaznia and R. Ebrahimi, "Providing a Solution for Optimal Management of Resources using the Multi-objective Crow Search Algorithm in Cloud Data Centers," in *2023 9th International Conference on Web Research (ICWR)*, IEEE, pp. 179–184, 2023. https://doi.org/10.1109/ICWR57742.2023.10139192.

[5] P. Neelima and A. R. M. Reddy, "An efficient load balancing system using adaptive dragonfly algorithm in cloud computing," *Cluster Comput.*, vol. 23, no. 4, pp. 2891–2899, 2020. https://doi.org/10.1007/s10586-020-03054-w.

[6] Y. Sun, J. Li, X. Fu, H. Wang, and H. Li, "Application research based on improved genetic algorithm in cloud task scheduling," *J. Intell. Fuzzy Syst.*, vol. 38, no. 1, pp. 239–246, 2020. https://doi.org/10.3233/JIFS-179398

[7] D. Yu, Z. Ma, and R. Wang, "Efficient smart grid load balancing via fog and cloud computing," *Math. Probl. Eng.*, vol. 2022, no. 1, p. 3151249, 2022. https://doi.org/10.1155/2022/3151249.

[8] S. A. Hashmi, C. F. Ali, and S. Zafar, "Internet of things and cloud computing-based energy management system for demand side management in smart grid," *Int. J. Energy Res.*, vol. 45, no. 1, pp. 1007–1022, 2021. https://doi.org/10.1002/er.6141.

[9] K. Perumal, S. Mohan, J. Frnda, and P. B. Divakarachari, "Dynamic resource provisioning and secured file sharing using virtualization in cloud azure," *J. cloud Comput.*, vol. 11, no. 1, p. 46, 2022. https://doi.org/10.1186/s13677-022-00326-1.

[10] N. Almurisi and S. Tadisetty, "Cloud-based virtualization environment for iot-based wsn: solutions, approaches and challenges," *J. Ambient Intell. Humaniz. Comput.*, vol. 13, no. 10, pp. 4681–4703, 2022. https://doi.org/10.1007/s12652-021-03515-z.

[11] A. Hameed *et al.*, "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems," *Computing*, vol. 98, pp. 751–774, 2016. https://doi.org/10.1007/s00607-014-0407-8.

[12] A. Belgacem, "Dynamic resource allocation in cloud computing: analysis and taxonomies," *Computing*, vol. 104, no. 3, pp. 681–710, 2022. https://doi.org/10.1007/s00607-021-01045-2.

[13] A. Abid, M. F. Manzoor, M. S. Farooq, U. Farooq, and M. Hussain, "Challenges and Issues of Resource Allocation Techniques in Cloud Computing.," *KSII Trans. Internet Inf. Syst.*, vol. 14, no. 7, pp. 2815-2839, 2020. http://doi.org/10.3837/tiis.2020.07.005.

[14] Z. Amini, M. Maeen, and M. R. Jahangir, "Providing a load balancing method based on dragonfly optimization algorithm for resource allocation in cloud computing," *Int. J. Networked Distrib. Comput.*, vol. 6, no. 1, pp. 35–42, 2018. https://doi.org/10.2991/ijndc.2018.6.1.4.

[15] H. Attaran, N. Kheibari, and D. Bahrepour, "Toward integrated smart city: A new model for implementation and design challenges," *GeoJournal*, vol. 87, no. Suppl 4, pp. 511–526, 2022. https://doi.org/10.1007/s10708-021-10560-w.

[16] H. F. Farimani, D. Bahrepour, and S. R. K. Tabbakh, "Reallocation of virtual machines to cloud data centers to reduce service level agreement violation and energy consumption using the FMT method," *J. Inf. Syst. Telecommun.*, vol. 4, no. 28, p. 316, 2020. https://doi.org/10.7508/jist.2019.04.007.

[17] L. M. Haji, S. Zeebaree, O. M. Ahmed, A. B. Sallow, K. Jacksi, and R. R. Zeabri, "Dynamic resource allocation for distributed systems and cloud computing," *TEST Eng. Manag.*, vol. 83, no. May/June 2020, pp. 22417–22426, 2020.

[18] J. Chen, T. Du, and G. Xiao, "A multi-objective optimization for resource allocation of emergent demands in cloud computing," *J. Cloud Comput.*, vol. 10, pp. 1–17, 2021. https://doi.org/10.1186/s13677-021-00237-7.

[19] R. Asif, K. A. Alam, K. M. Ko, and S. U. R. Khan, "Task scheduling in a cloud computing environment using a whale optimization algorithm," in *Proceedings of the First International Workshop on Intelligent Software Automation: ISEA 2020*, Springer, 2021, pp. 37–52. https://doi.org/10.1007/978-981-16-1045-5_4.

[20] A. A. A. Gad-Elrab, A. S. Alsharkawy, M. E. Embabi, A. Sobhi, and F. A. Emara, "Adaptive multi-criteria-based load balancing technique for resource allocation in fog-cloud environments," *arXiv Prepr. arXiv2402.01326*, 2024. https://doi.org/10.48550/arXiv.2402.01326

[21] A. Nandwal and R. Jain, "Optimizing of resource allocation in cloud computing with advanced load balancing

algorithm,” *Int. J. Eng. Sci. Math.*, vol. 12, no. 7, pp. 82–90, 2023.

[22] N. Manikandan, N. Gobalakrishnan, and K. Pradeep, “Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment,” *Comput. Commun.*, vol. 187, pp. 35–44, 2022. https://doi.org/10.1016/j.comcom.2022.01.016.

[23] S. Mangalampalli, G. R. Karri, and U. Kose, “Multi Objective Trust aware task scheduling algorithm in cloud computing using Whale Optimization,” *J. King Saud Univ. Inf. Sci.*, vol. 35, no. 2, pp. 791–809, 2023. https://doi.org/10.1016/j.jksuci.2023.01.016.

[24] M. Aruna, D. Bhanu, and S. Karthik, “An improved load balanced metaheuristic scheduling in cloud,” *Cluster Comput.*, vol. 22, no. Suppl 5, pp. 10873–10881, 2019. https://doi.org/10.1007/s10586-017-1213-9.

[25] S. Li, Z. Yan, and B. Hu, “A PSO-based Resource Allocation and Task Assignment Approach for Real-Time Cloud Computing-based Robotic Systems,” in *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Jinghong, China, IEEE, pp. 2305–2310,2022. https://doi.org/10.1109/ROBIO55434.2022.10011855.

[26] A. S. A. Beegom and M. S. Rajasree, “Integer-pso: a discrete pso algorithm for task scheduling in cloud computing systems,” *Evol. Intell.*, vol. 12, pp. 227–239, 2019. https://doi.org/10.1007/s12065-019-00216-7.

[27] A. S. Ajeena Beegom and M. S. Rajasree, “Non-dominated sorting based PSO algorithm for workflow task scheduling in cloud computing systems,” *J. Intell. Fuzzy Syst.*, vol. 37, no. 5, pp. 6801–6813, 2019. https://doi.org/10.3233/JIFS-190355.

[28] M. Saad, R. N. Enam, and R. Qureshi, “Optimizing multi-objective task scheduling in fog computing with GA-PSO algorithm for big data application,” *Front. big Data*, vol. 7, p. 1358486, 2024. https://doi.org/10.3389/fdata.2024.1358486.

[29] N. Manikandan, P. Divya, and S. Janani, “BWFSO: hybrid Black-widow and Fish swarm optimization Algorithm for resource allocation and task scheduling in cloud computing,” *Mater. Today Proc.*, vol. 62, pp. 4903–4908, 2022. https://doi.org/10.1016/j.matpr.2022.03.535.

[30] M. A. El-Shorbagy and A. E. Hassanien, “Particle swarm optimization from theory to applications,” *Int. J. Rough Sets Data Anal.*, vol. 5, no. 2, pp. 1–24, 2018. https://doi.org/10.4018/IJRSDA.2018040101.

[31] L. Zuo, S. Dong, L. Shu, C. Zhu, and G. Han, “A multiqueue interlacing peak scheduling method based on tasks’ classification in cloud computing,” *IEEE Syst. J.*, vol. 12, no. 2, pp. 1518–1530, 2016. https://doi.org/10.1109/JSYST.2016.2542251.

[32] N. Mansouri, R. Ghafari, and B. M. H. Zade, “Cloud computing simulators: A comprehensive review,” *Simul. Model. Pract. Theory*, vol. 104, p. 102144, 2020. https://doi.org/10.1016/j.simpat.2020.102144.

Davoud Bahrepour was born in Mashhad, Iran, in 1982. He received the M.S. and Ph.D. degrees in computer engineering from Science and Research Branch, Islamic Azad University, Tehran, Iran, in 2007 and 2012, respectively. Since 2013, he has been an Assistant Professor with the Computer Engineering Department, Islamic Azad University of Mashhad, Mashhad. His current research interests include cloud computing, IoT, and smart city.

Nastaran Evaznia received the M.S. degree in Computer Engineering from Azad University, Mashhad, Iran. Since 2013, she has been a university lecturer at Azad University of Mashhad, with several years of experience in the IT industry. She has developed expertise in cloud computing, IoT, fog computing, and blockchain. Her research interests include cloud computing, fog computing, IoT, and blockchain technologies.

Tahereh Khodabakhshi was born in Neyshabur, Iran, in 1986. She earned her M.S. degree in Information Technology Engineering from Islamic Azad University in Mashhad, Iran, in 2020. Since 2021, she has been a Computer Teacher in the computer department at Rashed Institute. Her current research interests include cloud computing, blockchain, and fog computing.