

Multi-objective tasks scheduling using bee colony algorithm in cloud computing

Mehdi Salehi Babadi¹, Mohammad Ebrahim Shiri¹, Mohammad Reza Moazami Goudarzi²,
Hamid Haj Seyyed Javadi³

¹Department of Computer Engineering, Borujerd Branch, Islamic Azad University, Borujerd, Iran

²Department of Mathematics, Borujerd Branch, Islamic Azad University, Borujerd, Iran

³Department of Mathematics and Computer Science, Shahed University, Tehran, Iran

Article Info

Article history:

Received Feb 20, 2021

Revised Mar 10, 2022

Accepted Apr 8, 2022

Keywords:

Artificial bee colony algorithm

Cellular automata

Cloud computing

Resource allocation

ABSTRACT

Due to the development of communication device technology and the need to use up-to-date infrastructure ready to respond quickly and in a timely manner to computational needs, the competition for the use of processing resources is increasing nowadays. The scheduling tasks in the cloud computing environment have been remained a challenge to access a quick and efficient solution. In this paper, the aim is to present a new tactic for allocating the available processing resources based on the artificial bee colony (ABC) algorithm and cellular automata for solving the task scheduling problem in the cloud computing network. The results show the performance of the proposed method is better than its counterparts.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Mohammad Ebrahim Shiri

Department of Computer Engineering, Borujerd Branch, Islamic Azad University

Borujerd, Iran

Email: shiri@aut.ac.ir

1. INTRODUCTION

With the dramatic increase in the variety of information technology, equipment and services, the management of services offered in this area has also faced many challenges. Managing problems and requests, managing equipment and resources related to technical support services, and allocating them to users, as well as monitoring, controlling, and scheduling are among the causes that force information technology (IT) [1]. Managers to provide useful and efficient tools themselves. There is also a need for people to do their heavy computing work without having expensive hardware and software through services [2]. Cloud computing has been the latest technology response to these needs. The National Institute of Technology and Standards defines cloud computing as: "cloud computing is a model for providing easy access based on the demand of user by the network to a set of changeable and configurable computing resources (e.g., networks, servers, storage, applications, and services) that this access can be provided or release with minimal need for resources management or the need for direct intervention by the service provider". Sharing "intangible and consumable" computing power among several tenants can improve productivity rates because other servers are not idle for any reason with this effective way and computers are being used more because cloud computing clients do not need to calculate and determine their maximum load [3].

A cellular automaton is a mathematical model for representing systems in which objects called cells together model system behavior, which can be defined in one or more dimensional formats [4]. The

homogenous and parallel structure of cellular automata makes it suitable for modeling different types of physical systems. To optimally model a physical system, the simple structure of cellular automata is used as finite and local interactions between cells. The simple structure is defined as a one-dimensional cellular automaton and each cell with two states (1.0) with a uniform ternary neighbor (itself and its right and left neighbors). The most important feature of a cellular automata structure is its modularity. According to interactions [4], it is observed that provide one-dimensional three-neighbor cellular automata provides the best efficiency for modeling physical systems.

In this paper, a new task scheduling problem in a cloud environment, a highly distributed computing platform, is concentrated. A time-aware scheduling algorithm based on artificial bee colony (ABC) is proposed to solve this problem. The primary objective of the proposed algorithm is to achieve a good trade-off between response time, schedule length ratio, and efficiency to complete the tasks in the Cloud system. Our approach was experimentally evaluated and on many datasets of various sizes. The results prove that the proposed algorithm achieved better results and was more optimal.

In [5], a new different optimization algorithm, evolution by combining elite to schedule tasks in the cloud computing environment is voluntary, which is called multi-objective ant lion optimizer (MOALO). MOALO algorithm has performed better than other algorithms. In [6], a new load balancing was also, which proposed by combining a cuckoo search algorithm with a bee colony called a self-adaptive artificial bee colony (SABC). It increases the utilization of resources, and execution time is reduced. Best task-to-virtual machine mapping is computed by the proposed algorithm. Speed of virtual machines (VM) processing and submitted workload length has an effect on it.

In [7], proposes an in-depth reinforcement learning model based on learning the quality of service (QoS) feature to optimize data center resource planning. In the in-depth learning phase, we propose a QoS feature learning method based on the enhancement of automatic voice removal encoders to extract stronger QoS feature information. The study [8] provides a technical analysis of cloud service deployment approaches in internet of things (IoT) systems. The key point of this technical analysis is to identify the basic studies in service placement approaches that need more attention to develop more efficient and effective strategies in IoT locations.

In [9], a new cloud computing task-scheduling algorithm that introduces min-min and max-min algorithms to generate initialization population selects task completion time and load balancing as double fitness functions and improves the quality of initialization population, algorithm searchability, and convergence speed was proposed. In [10], for task scheduling in the cloud, we use particle swarm optimization (PSO), firefly algorithm (FA), bat algorithm (BA), and grasshopper optimize cloud algorithm (GOA), which are swarm-based algorithms. The experimental results indicate that the improved GOA can optimize task scheduling problems by effective utilization of available resources.

In [11] provides a new approach for improving the task scheduling problem in a cloud-fog environment in terms of execution time (makespan) and operating costs for bag-of-tasks applications. A task scheduling evolutionary algorithm has been proposed. A single custom representation of the problem and a uniform intersection is built for the proposed algorithm. In [12], a task scheduler based on a genetic algorithm for the cloud computing system was invented that used the genetic algorithm [13] to minimize the time to complete tasks in the cloud computing environment.

In [14], an ant algorithm for symmetric scheduling of tasks on cloud computing was presented. The goal is to allocate the optimal resource to each task according to the source and task characteristics. Each task is regarded as an ant, and the weight of the resources is regarded as a pheromone. In other words, the higher the source weight, the higher the pheromone. The results show the optimum completion time and balance. In [15] to enhance the performance of cloud computing and reduce delay time in the queue waiting for jobs. The proposed algorithm tries to avoid some significant challenges that throttle from developing applications of clouding computing. Our experimental result of the proposed job scheduling algorithm shows that the proposed schemes possess outstanding enhancing rates with a reduction in waiting time for jobs in the queue list.

Paper [16], apply the latest whale optimization metamorphosis (WOA) algorithm to schedule cloud work with a multi-objective optimization model, with the aim of improving the performance of a cloud system with given computing resources. Accordingly, we propose an advanced approach called WOA cloud scheduling (IWC) Improvement to further improve the ability to search for the optimal WOA-based approach. In [17] paper, a modified henry gas solubility optimization (HGSO) is presented, which is based on the WOA and comprehensive opposition-based learning (COBL) for optimum task scheduling. The proposed method is named henry gas solubility whale cloud (HGSWC). HGSWC is validated on a set of thirty-six optimization benchmark functions, and it is contrasted with conventional HGSO and WOA.

In [18] proposes three main contributions to solve this load balancing problem. First, it proposes a heterogeneous initialized load balancing (HILB) algorithm to perform a good task scheduling process that improves the makespan in the case of homogeneous or heterogeneous resources and provides a direction to

reach optimal load deviation. Second, it proposes a hybrid load balance based on a genetic algorithm (HLBGA) as a combination of HILB and genetic algorithm (GA). Third, a newly formulated fitness function that minimizes the load deviation is used for GA [19] proposes an efficient algorithm traffic-aware adaptive server load balancing (TAASLB) to balance the flows to the servers in a data center network. It works based on two parameters, residual bandwidth and server capacity. It detects the elephant flows and forwards them towards the optimal server where they can be processed quickly.

In [20], proposed hybrid electro search with a genetic algorithm (HESGA) to improve the behavior of task scheduling by considering parameters such as makespan, load balancing, utilization of resources, and cost of the multi-cloud. The proposed method combined the advantage of a genetic algorithm and an electro search algorithm. The genetic algorithm provides the best local optimal solutions, whereas the electro search algorithm provides the best global optima solutions. The proposed algorithm outperforms existing scheduling algorithms such as hybrid particle swarm optimization genetic algorithm (HPSOGA), GA, evolution strategy (ES), and ant colony optimization algorithm (ACO). In [21], an efficient hybridized scheduling algorithm that replicates the parasitic behavior of the cuckoo and food gathering habit of the crow bird, named the cuckoo crow search algorithm (CCSA), had been presented for improvising the task scheduling process. The crow bird always stares at its neighbors, looking for a better food source than the one it currently possesses.

In [22], a resource scheduling optimization model based on service level agreement based on a random planning approach in cloud computing is presented. In [23], the flexible task scheduling problem in a cloud computing system is studied and solved by a hybrid discrete artificial bee colony (ABC) algorithm, where the considered problem is first modeled as a hybrid flow shop scheduling (HFS) problem. Both a single objective and multiple objectives are considered. In [24], a cloud computing multi-objective task scheduling optimization based on a fuzzy self-defense algorithm is proposed. Select the shortest time, the degree of resource load balance, and the cost of multi-objective task completion as the goal of cloud computing multi-objective task scheduling, establish a mathematical model to measure the effect of multi-objective task scheduling and construct the objective function of cloud computing multi-objective task scheduling.

2. METHOD

2.1. Problem statement

The cloud management program manages all cloud resources using various cloud modules such as network module, operating system image module, cost module, and endorsement module. The tasks are distributed in different data centers (DC's) available through the cloud infrastructure. Each data center divides the user tasks into several "sub-task" and makes them available to processing elements (PE) [25].

2.2. Proposed schema

The proposed scheduling module will have the task of assigning the right job to the right source at the right time in the framework of cloud. In Figure 1, DC represents a data center and PE represents the processing elements. In this model, a new cloud is regarded as a set of user tasks that its complex computing is performed using cloud resources. Suppose UserJob= (U_1, U_2, \dots, U_N) is a set of user programs that enters (execution-request) at a given time. Each UserJob (U_i) is represented by a double $\langle ai, di \rangle$, where ai represents the time of entry, and di represents the time limit of the UserJob. If a task fails within its time limit, it is designated as a failed task and must be re-entered into a new scheduling queue. In the scheduling process, user tasks are assigned to available data centers (D_1, D_2, \dots, D_M), where $M \leq N$, means that the number of data centers may be less than the number of tasks requested. Each data center (D_i) is represented by a double $\langle Ci, mi \rangle$. In this dual, Ci is the cost of executing tasks in the data center per unit time, and mi is the number of PEs available for executing user tasks. Each data center has a number of processing elements $\{P_{E1}, P_{E2}, \dots, P_{Ek}\}$ to execute users' work. Each processing element with a PE's characteristic also means processing speed[26].

2.3. Problem parameters

2.3.1. Response-time problem analysis

These tasks are embedded in one of the existing datacenters called D , which has a certain number of PE processing units, and these tasks must be distributed among these processing units. Each U has a required processing volume, i.e., P and an allowed time i.e., T . Each PE also has a PE_j processing speed and a processing cost. If the execution time of T_k in PE_j is shown by τ_k , then the termination time can be stated: this processing time is proportional to the difference in processing volume required for P_k in PE_j .

$$P_{k\text{remaining}} = PK_{\text{initial}} - \tau_k * PE_j \leq \tau_k = \frac{P_k}{PE_j} \quad (1)$$

If $P_k > P_{ej}$ is, then the T_k spent for U_k task processing is decreased from the amount of processing required, and some processing remains. Naturally, we want this processing time to be less than the time allowed to perform the full task of the user task, i.e., Tim. In general, assuming the parallel distribution of U_k tasks among P_{ej} processors, the time of completing task U is equal to the time of completing the longest tasks.

$$Makespan = \max\{Finish(\tau_k)\} \tag{2}$$

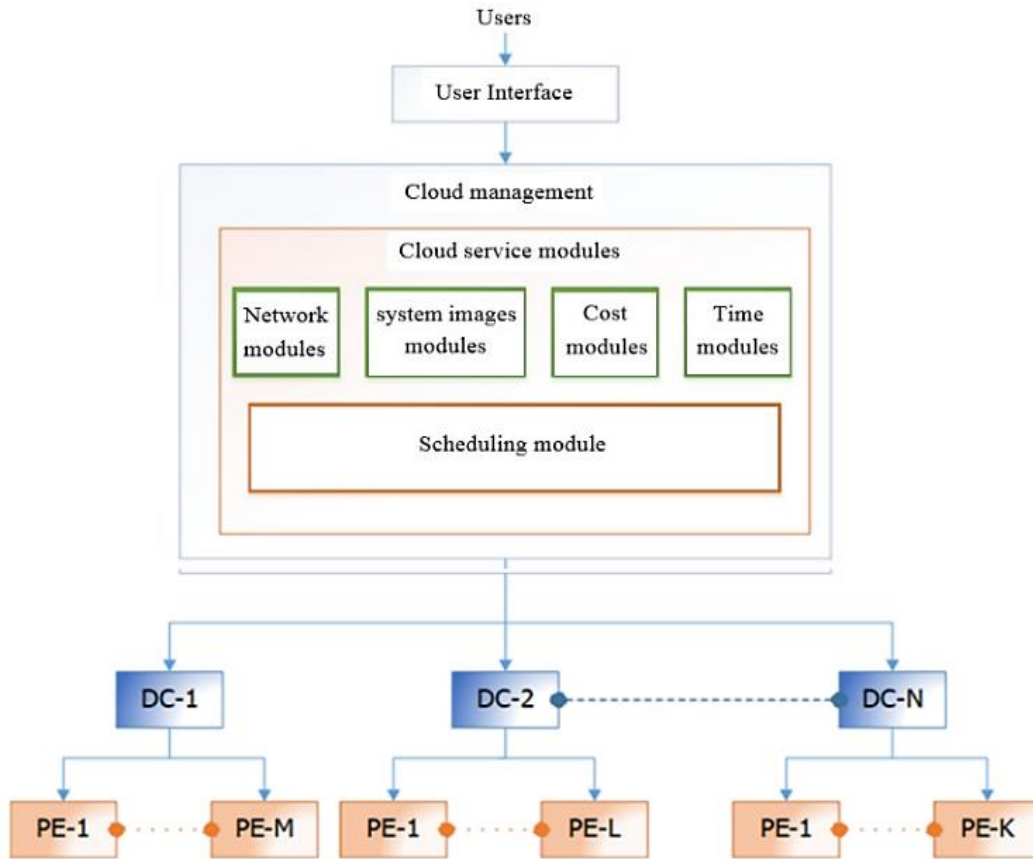


Figure 1. Management structure of a cloud processor [26]

2.3.2. Energy consumption problem analysis

If the unit processing cost per P_{ej} is equal to PC_j , then the energy consumed to perform the whole task U is equal to (3).

$$E_U = \sum_{k=1}^n (\tau_k * PC_j) \tag{3}$$

This is equal to the cost spent for the UserJob in the D datacenter.

2.3.3. The efficiency formulation

Here, a new relationship must be established to determine the efficiency of the task distribution algorithm. This productivity can be obtained by comparing the amount of processing that could do with the whole time of D (data) with U (UserJob) with the amount of processing that actually performed. That's mean:

$$eff = \frac{\sum_{k=1}^n (\tau_k * PE_{jk})^{-1}}{Makespan * \sum_{j=1}^J PE_j} \tag{4}$$

This equation is equal to the inverse of the sum of the time that each processor unit was occupied by a particular task divided by the total time U task multiplied by the total available processing power.

Therefore, the lower the total execution time of the task and the more optimized processing power is distributed among the tasks, the higher the efficiency of the algorithm. The schedule length ratio (SLR) parameter can also be obtained from this (5).

$$SLR = \frac{\max\{Finish(\tau_k)\}}{\min\{Finish(\tau_k)\}} \quad (5)$$

Now suppose each task U has linear cell automata that correspond to the order of T subtasks, and each cell automata have a bee with greedy selection. At each m period of the processing resource allocation algorithm operation, we have a number of P_k residual from different U_k , each of which has a bee with greedy selection. To continue task for each P_k , one power of choice is defined which is the ability of the bee assigned to it, which is equal to the amount of processing operations residual from the U_k task. In short, any remaining or new task in any given period will choose the most powerful of the available processing resources, depending on the selection power it has, and this process will continue until all tasks are completed. If we show the periods of allocating tasks by m , each bee's selection power is (6).

$$POW_k = P_{k_{initial}} - \sum_{m=1}^M \tau_k(m) * PES_j(m) \quad (6)$$

That is, the selection power of each bee for its own task is equal to the initial amount of processing required for that task minus the processing power allocated to that task at different times. However, with each iteration of the resource allocation algorithm of task U_k , that has the most selection power to obtain the highest P_{ej} . However, the energy consumption equation is also rewritten as (7).

$$E_U = \sum_{m=1}^M \sum_{k=1}^n (\tau_{km} * PC_{jm}) \quad (7)$$

And the efficiency of the algorithm is also equal to (8).

$$eff = \frac{\sum_{m=1}^M \sum_{k=1}^n (\tau_k * PE_{jm})^{-1}}{\text{Makespan} * \sum_{j=1}^J PE_j} \quad (8)$$

The unknown inputs of the problem are the number of tasks available and the processing volume of each one, the number of processors and processing power, and the processing cost of each, and the outputs of the problem, as well as the total processing time, total energy consumed, and algorithm efficiency that are compared in two states of distribution with greedy selection and constant random distribution. In the constant random distribution, each task is assigned as a processing resource at the beginning of the task, and this allocation does not change anymore. One remaining point is that given the extension of servers and processing resources in cloud systems, the probability that the number of tasks assigned is greater than the number of processors is very low, usually $J \geq K$.

3. RESULTS AND DISCUSSION

3.1. Test data of the problem

After introducing the overview, examining the technical terms, and reviewing similar work, we were able to present a new theory for the distribution of cloud computing resources among existing tasks. It is time to look at the results and judge the validity and effect clouds of the proposed technique. First, we implement the simulation with a 12-sub task process on a cloud processor with 20 hardware sources. Naturally, a virtual machine is defined simultaneously for each task, but how each virtual machine accesses the hardware is set by the proposed algorithm.

The first step is to determine the number of different hardware resources with different processing power. The reason for the random selection is that according to the use of cloud servers from discrete processing resources in a very large area of the Internet network and connection and disconnection of many of these resources, it cannot be assured from existing processing power. The processing values required in each sub-task are also unknown and randomly determined in a reasonable range.

Also, the processing cost depends not only on the processing power but also on other parameters such as the distance, and position of the processor. This issue should also be determined randomly. After specifying the tasks and hardware resources available for each task, automata and a bee is assigned, and their performance is such that the location of each subtask may vary depending on the selection power of the bee during the execution of the program, as shown in the Table 1. Whole task execution using cellular automata to divide training resources and tasks in this example took 48 time periods, i.e., 12 parallel automata changed 48 times.

Table 1. Change of the 12-cell automata in the first 15 iterations of the algorithm

1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
7	14	11	6	2	4	12	8	16	10	20	12
7	14	11	6	2	4	12	8	16	10	20	12
7	14	11	6	2	4	12	8	16	10	20	12
7	14	11	6	2	4	12	20	8	10	16	12
7	14	11	6	2	4	12	16	20	10	8	12
7	14	11	6	2	4	12	16	20	10	8	12
7	14	11	12	2	4	6	16	20	10	8	12
7	14	11	6	2	12	4	16	20	10	8	12
7	14	11	4	2	12	6	16	20	10	8	12
7	14	12	4	2	6	11	16	20	10	8	12
7	14	6	4	2	11	12	16	20	10	8	12
7	14	11	12	2	4	6	16	20	10	8	12
7	14	4	12	2	6	11	16	20	10	8	12
7	14	12	6	2	11	4	16	20	10	8	12

3.1.1. The response times

As we said, the whole task is processed by the available cloud resources once using the variable cellular automata and once by the allocation of random processing resources. Now we compare the output parameters between these two modes. As shown in Figure 2, processing time of whole tasks with variable automata in a left column equals 48 seconds, and random allocation of resources in a right column equals 220 seconds the difference is very significant. As it can be seen, the proposed algorithm in this study reduces the processing time by more than four times.

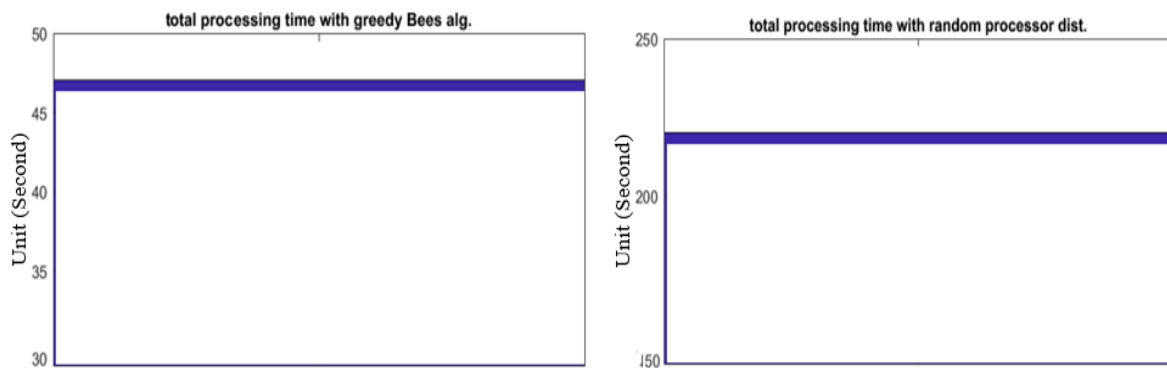


Figure 2. Processing time of whole tasks with variable automata

3.1.2. Energy consumption criterion

In Figure 3 shows energy consumption criterion of fault data in favorable case. The energy consumption in the two methods used in the cellular automata method was about 450,000 joules and in the random allocation method was about 3.6 million joules. It can be seen that the proposed method reduces energy consumption by eight times compared to the random allocation method 1. In Table 1 change of the 12-cell automata in the first 15 iterations of the algorithm. After these 15 iterations, due to zero of most processors remained, other automata elements were not changed until the last remaining processing.

3.1.3. The efficiency criterion

Figure 4 shows the efficiency criterion. Comparison of processing efficiency in two random distribution modes and cellular automata with astronomical differences is shown in Figure 4. It is observed that the improvement of all system performance parameters in the use of cellular automata with greedy bee selector is more significant than the random allocation of resources.

3.1.4. The SLR criterion

Figure 5 show comparison of SLR parameter in mentioned methods. It can be seen that the ratio of the longest process to the shortest one in cellular automata has been 45 times and in random distribution method 220 times. It can be seen that the improvement of all system performance parameters in the use of

cellular automata with a greedy bee selector compared to random allocation of resources is very significant. It seems that the proposed method can have a significant impact on improving cloud computing services.

In order to investigate the performance of the algorithm under different conditions, we are cloud the simulation in 45 sub-tasks with a maximum load of 4,000 processing units and distributed among 65 hardware sources with a processing capability of 1,000 units and a processing cost of 10,000 units. The results are Figure 6, show the time taken to perform all tasks in the presented method was only eight periods, and in the random allocation method, the tasks were over 1,400 periods, which the difference is very significant. The cost of processing in the proposed method was nearly two thousand times less than the random allocation method as shown in Figure 7.

Figure 8 show the processing efficiency seems to have decreased here, unlike the previous simulation, and it is less than the random distribution of resources. In the case of random distribution, some processing resources seem to have spent much more time than other processing sources, and this has been influenced in the following graph. Figure 9 shows the SLR parameter, which shows the time difference of working among the cloud processor, is much less than the random distribution method in the presented method, which indicates the balanced distribution of tasks in the current algorithm.

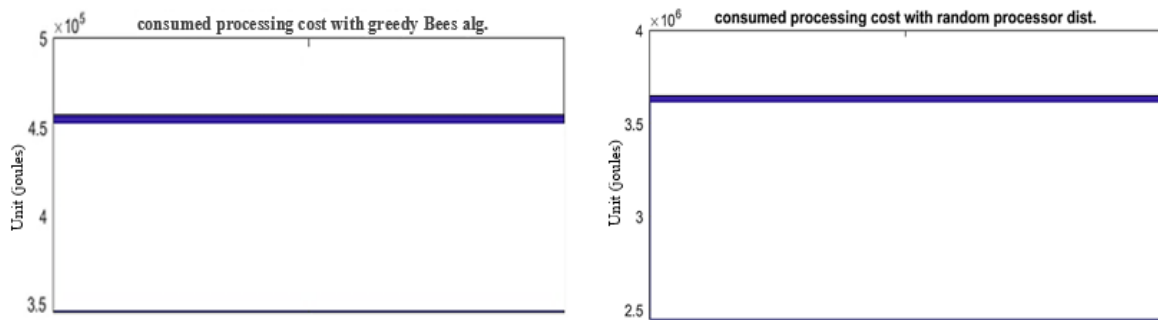


Figure 3. Energy consumption criterion

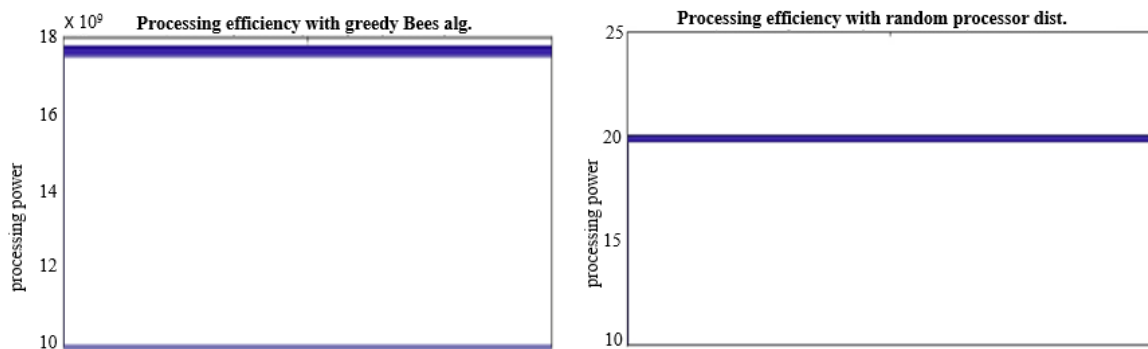


Figure 4. The efficiency criterion

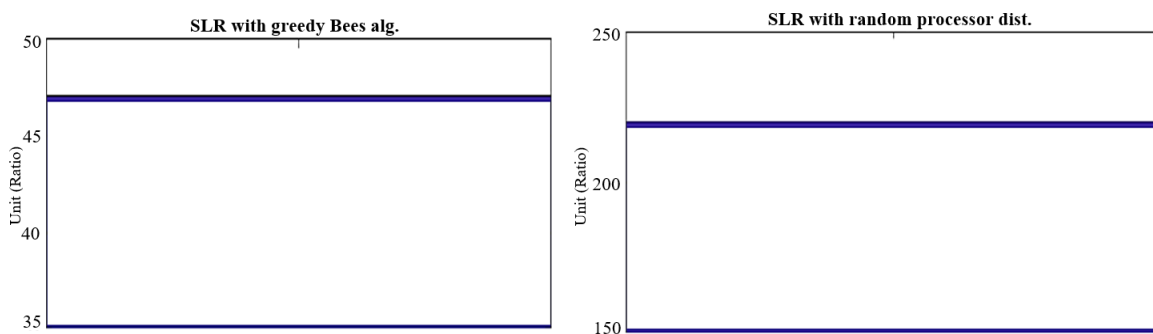


Figure 5. Comparison of SLR parameter

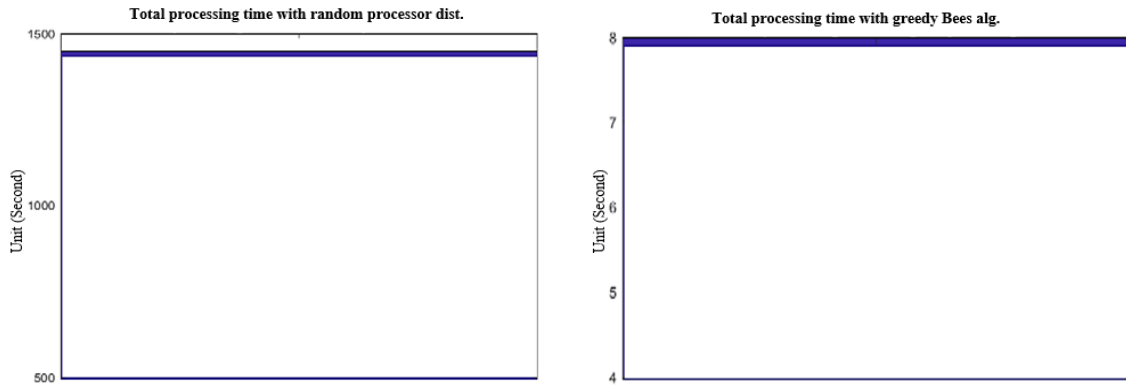


Figure 6. Total processing time with random processor dist.

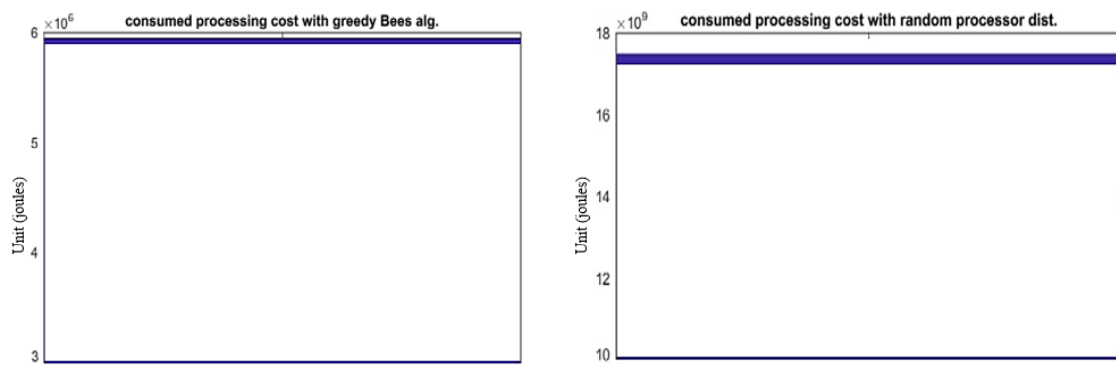


Figure 7. The cost of processing in the proposed method was nearly two thousand times less than the random allocation method

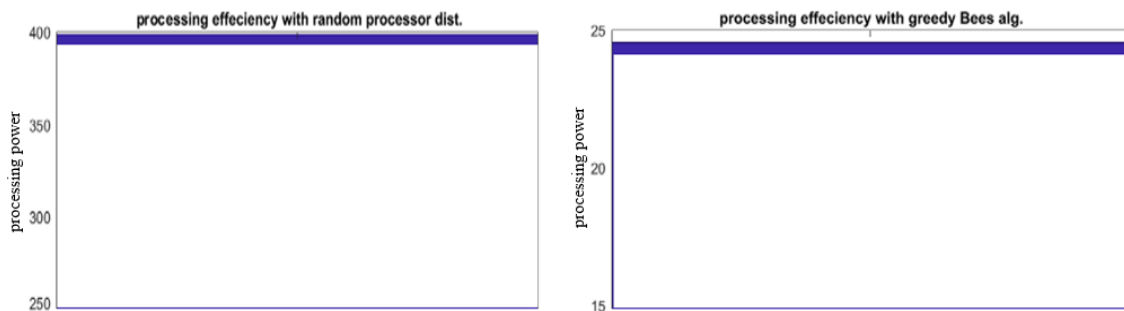


Figure 8. Processing efficiency with greedy Bees alg.

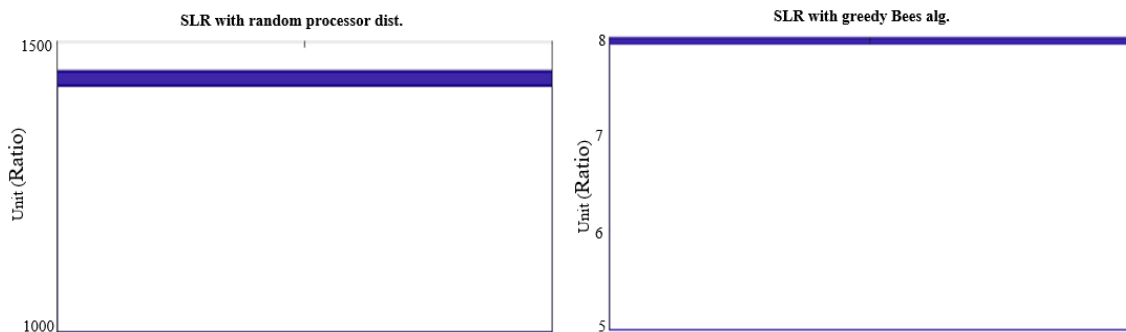


Figure 9. SLR with greedy Bees alg.

4. CONCLUSION AND FUTURE WORKS

In the two simulations above, it is assumed that the power of the preceding available hardware resources was less than the amount of processing volume required for each task. To finish the discussion, we run the simulation in a different way. This time, 65 tasks with a maximum processing volume of 5,000 units are distributed among a hundred cloud processors with a maximum processing power of 7,000 units. The processing cost is 10,000 units. The results showed that the current method still has had superiority to the random allocation of resources, but this level of superiority was not as significant as in the previous cases. The proposed algorithm for actively distributing processing resources between the tasks presented seems to be more effective when the available hardware processing power is less than the processing power required by the tasks. The remaining point is the time allowed for processing, which according to the speed limit and the number of processors, it cannot consider this time constraint. While from the users' point of view, the length of time is undesirable, but at the same time, the user cannot expect any task that expects the cloud system to be done by the user. However, the allowed processing time of a parameter is independent, and any value can be assumed, and it cannot expect the process distribution system to do a very large task using a small processor small in a short time.

There are several suggestions for future research: i) applying input tasks in specific cases, infinitely different states can be considered for sequencing and processing tasks, and system performance is examined in this case; ii) considering two parallel user tasks for engaging processors that become idle before completing all tasks; iii) comparison of techniques other than a random distribution of tasks for comparison with the proposed method; iv) considering unforeseen factors such as the non-availability of some processing resources. The current method has many advantages compared to the random distribution method, but to complement other discussions, researchers can provide other tactics of distributing processing resources at the same time as the method proposed in this study and compare their performance with the implementation of both simulations.




REFERENCES

- [1] R. Ding and Z. X. Guo, "Microstructural modelling of dynamic recrystallisation using an extended cellular automaton approach," *Computational Materials Science*, vol. 23, no. 1–4, pp. 209–218, Apr. 2002, doi: 10.1016/S0927-0256(01)00211-7.
- [2] H. Tabrizchi and M. Kuchaki Rafsanjani, "A survey on security challenges in cloud computing: issues, threats, and solutions," *The Journal of Supercomputing*, vol. 76, no. 12, pp. 9493–9532, Dec. 2020, doi: 10.1007/s11227-020-03213-1.
- [3] F. Mahan, S. M. Rozehkhani, and W. Pedrycz, "A novel resource productivity based on granular neural network in cloud computing," *Complexity*, vol. 2021, pp. 1–15, May 2021, doi: 10.1155/2021/5556378.
- [4] P. Christen and O. Del Fabbro, "Automatic programming of cellular automata and artificial neural networks guided by philosophy," in *New Trends in Business Information Systems and Technology*, Springer, 2021, pp. 131–146. doi: 10.1007/978-3-030-48332-6_9.
- [5] L. Abualigah and A. Diabat, "A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments," *Cluster Computing*, vol. 24, no. 1, pp. 205–223, Mar. 2021, doi: 10.1007/s10586-020-03075-5.
- [6] R. Kumar and A. Chaturvedi, "Improved cuckoo search with artificial bee colony for efficient load balancing in cloud computing environment," in *Smart Innovations in Communication and Computational Sciences*, 2021, pp. 123–131. doi: 10.1007/978-981-15-5345-5_11.
- [7] B. Wang, F. Liu, and W. Lin, "Energy-efficient VM scheduling based on deep reinforcement learning," *Future Generation Computer Systems*, vol. 125, pp. 616–628, Dec. 2021, doi: 10.1016/j.future.2021.07.023.
- [8] L. Heng, G. Yin, and X. Zhao, "Energy aware cloud-edge service placement approaches in the Internet of Things communications," *International Journal of Communication Systems*, vol. 35, no. 1, Jan. 2022, doi: 10.1002/dac.4899.
- [9] G. E. Weiqing and C. Yanru, "Task-scheduling algorithm based on improved genetic algorithm in cloud computing environment," *Recent Advances in Electrical and Electronic Engineering (Formerly Recent Patents on Electrical and Electronic Engineering)*, vol. 14, no. 1, pp. 13–19, Jan. 2021, doi: 10.2174/2352096513999200424075719.
- [10] A. Zandvakili, N. Mansouri, and M. M. Javidi, "Swarm-based algorithms using chaos for task scheduling in cloud," in *2021 7th International Conference on Web Research (ICWR)*, May 2021, pp. 211–215. doi: 10.1109/ICWR51868.2021.9443157.
- [11] M. N. Abdulredha, B. A. Attea, and A. J. Jabir, "An evolutionary algorithm for task scheduling problem in the cloud-fog environment," *Journal of Physics: Conference Series*, vol. 1963, no. 1, Jul. 2021, doi: 10.1088/1742-6596/1963/1/012044.
- [12] K. Karmakar, R. K. Das, and S. Khatua, "Resource scheduling for tasks of a workflow in cloud environment," in *International Conference on Distributed Computing and Internet Technology*, 2020, pp. 214–226. doi: 10.1007/978-3-030-36987-3_13.
- [13] Vijindra and S. Shenai, "Survey on scheduling issues in cloud computing," *Procedia Engineering*, vol. 38, pp. 2881–2888, 2012, doi: 10.1016/j.proeng.2012.06.337.
- [14] A. Mubeen, M. Ibrahim, N. Bibi, M. Baz, H. Hamam, and O. Cheikhrouhou, "Alts: an adaptive load balanced task scheduling approach for cloud computing," *Processes*, vol. 9, no. 9, Aug. 2021, doi: 10.3390/pr9091514.
- [15] A. S. Abdalkafor and K. M. Ali Alheeti, "A hybrid approach for scheduling applications in cloud computing environment," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 2, pp. 1387–1397, Apr. 2020, doi: 10.11591/ijece.v10i2.pp1387-1397.
- [16] X. Chen et al., "A WOA-based optimization approach for task scheduling in cloud computing systems," *IEEE Systems Journal*, vol. 14, no. 3, pp. 3117–3128, Sep. 2020, doi: 10.1109/JSYST.2019.2960088.
- [17] M. Abd Elaziz and I. Attiya, "An improved Henry gas solubility optimization algorithm for task scheduling in cloud computing," *Artificial Intelligence Review*, vol. 54, no. 5, pp. 3599–3637, Jun. 2021, doi: 10.1007/s10462-020-09933-3.
- [18] W. Saber, W. Moussa, A. M. Ghuniem, and R. Rizk, "Hybrid load balance based on genetic algorithm in cloud environment," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 3, pp. 2477–2489, Jun. 2021, doi: 10.11591/ijece.v11i3.pp2477-2489.




- [19] C. Fancy and M. Pushpalatha, "Traffic-aware adaptive server load balancing for software defined networks," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 3, pp. 2211–2218, Jun. 2021, doi: 10.11591/ijece.v11i3.pp2211-2218.
- [20] S. Velliangiri, P. Karthikeyan, V. M. Arul Xavier, and D. Baswaraj, "Hybrid electro search with genetic algorithm for task scheduling in cloud computing," *Ain Shams Engineering Journal*, vol. 12, no. 1, pp. 631–639, Mar. 2021, doi: 10.1016/j.asej.2020.07.003.
- [21] P. Krishnadoss, "CCSA: hybrid cuckoo crow search algorithm for task scheduling in cloud computing," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 4, pp. 241–250, Aug. 2021, doi: 10.22266/ijies2021.0831.22.
- [22] Q. Li and Yike Guo, "Optimization of resource scheduling in cloud computing," in *2010 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, Sep. 2010, pp. 315–320. doi: 10.1109/SYNASC.2010.8.
- [23] J. Li and Y. Han, "A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system," *Cluster Computing*, vol. 23, no. 4, pp. 2483–2499, Dec. 2020, doi: 10.1007/s10586-019-03022-z.
- [24] X. Guo, "Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm," *Alexandria Engineering Journal*, vol. 60, no. 6, pp. 5603–5609, Dec. 2021, doi: 10.1016/j.aej.2021.04.051.
- [25] X. Zhang *et al.*, "Runtime model-based management of diverse cloud resources," in *Proc. of the 16th International Conference on Model Driven Engineering Languages and Systems*, 2013, pp. 572–588. doi: 10.1007/978-3-642-41533-3_35.
- [26] C. A. Lee, "A perspective on scientific cloud computing," Oct. 2010. doi: 10.1145/1851476.1851542.

BIOGRAPHIES OF AUTHORS






Mehdi Salehi Babadi    received his M.Sc. degree in software Engineering from Department of Computer Engineering, Borujerd Branch, Islamic Azad University, Borujerd, Iran, in 2012. Currently, He is a Ph.D. student in the Department of Computer Engineering at Borujerd Branch, Islamic Azad University, Borujerd, Iran since 2015. His Ph.D. advisor is Professor Mohammad Ebrahim Shiri. His research interests include social network, cellular automata, IoT, cloud computing, cryptography and security. He can be contacted at email: msalehib@hotmail.com.






Mohammad Ebrahim Shiri    received his Ph.D. degree in Computer Science from the Department of Computer Science, university of Montreal, Montreal, Canada in 2000. Currently, he is an Assistant Professor in the Department of Computer Science at Amirkabir University of Technology in Tehran, Iran. His research interests include machine learning, e-learning, database, network security and cloud computing. He can be contacted email: shiri@aut.ac.ir.



Mohammad Reza Moazami Goudarzi    received his Ph.D. degree in the Department of Mathematics, Islamic Azad University, Tehran, Iran in 2011. Currently, he is an assistant professor in the Department of Mathematics at Borujerd Branch, Islamic Azad University, Borujerd, Iran. His research areas include operation research and data envelopment analysis. He can be contacted at email: mrmoazamig@gmail.com.



Hamid Haj Seyyed Javadi    received the M.Sc. and Ph.D. degrees in Amirkabir University of Technology, Tehran, Iran in 1996 and 2003 respectively. He has been working as a fulltime faculty member and Professor in the Department of Mathematics and Computer Science at Shahed University, Tehran, Iran. His research interests are IoT, computer algebra, cryptography and security. He can be contacted at email: h.s.javadi@shahed.ac.ir.