Transactions on Machine Intelligence 3(4) (2022) 240-250

ISSN Online: 2821-1693

Transactions on Machine Intelligence

Journal Homepage: https://tmachineintelligence.ir/index.php/TMI

Load Balancing of Servers in Cloud Computing Using the Lion Optimization Algorithm

A. Esmaeili Moshiran[1], Sh. Babaei[2,*]

[1] Department of Computer Engineering, Faculty of Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran
[2] Assistant Professor, Department of Computer Engineering, Faculty of Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran

ARTICLE INFO

Article History:
Received 29 June 2020
Received in revised form 16 September 2020
Accepted 22 December 2020
Available online 24 December 2020

Keywords:
Cloud Computing, Virtualization, Cloud Servers, Load Balancing, Lion Optimization Algorithm (LOA)

ABSTRACT

The growing demand for rapid processing of large and heavy computational tasks, coupled with the advancements in networks and distributed systems, has led to the development of cloud computing systems. In cloud computing, all user needs are provided in a shared environment as services on-demand and as required. One of the significant challenges in cloud-based systems infrastructure is solving the problem of optimal and efficient load distribution. In other words, assigning more virtual machines to a single physical server leads to issues such as non-optimal resource allocation and load imbalance. Hence, the load among physical servers and, consequently, the entire system load must be balanced. It has been proven that the load balancing problem in cloud computing falls under the NP-hard category, and thus, various researchers have used heuristic, metaheuristic, greedy, and other algorithms to solve it. In this paper, the load balancing problem in cloud computing is considered as an optimization problem, and the Lion Optimization Algorithm (LOA) is used to solve it. The proposed method is simulated in MATLAB software and compared with Genetic Algorithm (GA) and Simulated Annealing (SA). The experimental results indicate that the proposed approach significantly improves load balancing in the allocation of virtual machines compared to GA and SA algorithms.

1. INTRODUCTION

With the advancement of information technology, the need for computational tasks to be performed anywhere and anytime has emerged. Additionally, there is a need for individuals to perform heavy computational tasks without having expensive hardware and software, through services. Cloud computing is the latest technological response to these needs. As this technology is still in its infancy, a universally accepted standard definition has not yet been provided, but most experts agree on certain aspects of its definition [1-10]. The National Institute of Standards and Technology (NIST) defines cloud computing as follows [11]: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management

* Corresponding Author: Sh.babaie@iaut.ac.ir
Assistant Professor, Department of Computer Engineering, Faculty of Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran

http://dx.doi.org/10.47176/TMI.2020.240

effort or service provider interaction." Cloud computing users do not own the physical infrastructure; instead, they rent it from a third-party provider, thereby avoiding capital expenditures. Cloud computing technology also has problems and complexities. One of the most critical and widely used issues is load balancing, as dynamic and fair distribution of load among all nodes must be ensured, and customers are always interested in reducing the total job execution time on machines. Customers can specify their required resources such as memory, disk space, processor, and network bandwidth, all of which are encapsulated in templates called virtual machines and mapped to physical resources. Virtualization technologies reallocate virtual machines to physical resources based on changes in their load to dynamically achieve system-wide load balance [12-14].

To improve resource utilization rates, resource allocation must be done acceptably, and load balancing among physical servers must be guaranteed. Accordingly, how resources are allocated to virtual machines to achieve load balancing in the cloud environment and ultimately improve the use of computational resources is crucial. Many algorithms have been introduced to solve the load balancing problem in cloud computing environments. Among these algorithms are some common scheduling algorithms such as Min-Min, Round-Robin, and evolutionary algorithms such as Ant Colony Optimization (ACO) [15-16], Genetic Algorithm (GA) [17], Particle Swarm Optimization (PSO) [12], Bee-inspired Algorithms [18], and others.

One of the metaheuristic algorithms that can be used to solve various optimization problems, including the load balancing problem, is the Lion Optimization Algorithm (LOA). The Lion Optimization Algorithm, inspired by the unique lifestyle and cooperative characteristics of lions, is proposed to solve complex optimization problems [19]. Considering the aforementioned points, this paper utilizes the Lion Optimization Algorithm to enhance load balancing of servers in cloud data centers.

2. FORMAL PROBLEM STATEMENT

First, we assume a virtualized environment consisting of a set of physical machines and virtual machines. Each physical machine or server can host one or several virtual machines, and each virtual machine implements an application. Let $J = \{1, 2, ..., m\}$ denote the set of physical machines and $I = \{1, 2, ..., n\}$ denote the set of virtual machines in the system. For simplicity, we assume that the environment consists of homogeneous physical servers, meaning all virtual machines have the same capacity C [20].

It is also assumed that the total capacity of the physical servers is sufficient to host all virtual machines. With the help of a resource monitoring unit, the utilization of virtual machines can be determined. Each virtual machine Vi has a workload denoted as W_i. Binary variables x_{ij} are used to represent the deployment of virtual machines:

$$x_{ij} = \begin{cases} 1 & If\ the\ i-th\ virtual\ machine\ is\ deployed\ on\ the\ j-th\ physical\ machine \\ 0 & Otherwise \end{cases} \quad (1)$$

The workload of a physical machine can be obtained by summing the workloads of the virtual machines deployed on it. Let P_j denote the workload of physical machine j:

$$P_j = \sum_{i=1}^{m} V_i x_{ij} \quad (2)$$

To prevent violating service level agreements, the workload on physical machines must be balanced. The remaining available resources on each physical machine are used as a criterion for load balancing. Let r_j denote the remaining capacity of physical machine j and $\bar{r}$ denote the average remaining capacity of all physical machines:

$$r_j = C - P_j \quad (3)$$

$$\bar{r} = \frac{1}{m} \sum r_j \quad (4)$$

Let $\sigma(s)$ denote the standard deviation of the remaining capacity across all physical machines:

$$\sigma(s)=\sqrt{\frac{1}{m}\sum_{j=1}^{m}(r_j-\bar{r})^2} \tag{5}$$

where s represents the mapping solutions. The objective is to find the optimal mapping s that achieves the best load balance in the system. Therefore, the model can be described as follows:

$$\min\ C(s)=\sqrt{\frac{1}{m}\sum_{j=1}^{m}(r_j-\bar{r})^2} \tag{6}$$

$$\sum_{j=1}^{m}x_{ij}=1 \tag{7}$$

$$P_j=\sum_{i=1}^{m}V_i\,x_{ij}\le C \tag{8}$$

$$r_j=C-P_j \tag{9}$$

$$\bar{r}=\frac{1}{m}\sum_{j=1}^{m}r_j \tag{10}$$

$$x_{ij}\in\{0,1\}\quad i\in I,\ j\in J \tag{11}$$

In this model, C(s) is the objective or fitness function that needs to be minimized. Constraint (7) states that each virtual machine should only be deployed on one physical machine. Constraint (8) ensures that the workload of a physical machine does not exceed its capacity.

2.1. Load Balancing in Cloud Computing Using the Lion Optimization Algorithm

As previously mentioned, this paper uses the Lion Optimization Algorithm (LOA) for load balancing of servers in cloud computing. The following sections describe the various stages of this algorithm to improve load balancing of servers in cloud data centers.

2.2. Encoding and Initialization

In the Lion Optimization Algorithm, an initial population of lions must be randomly created. In this algorithm, each member of the population (each lion) represents a solution to the problem, where a solution here is a mapping of virtual machines to physical machines. Given the aforementioned considerations, in the proposed approach, each solution is represented by an array of integers of length N, where N is the total number of virtual machines. The index of each element in the array represents the virtual machine number, and the integer value stored in each element represents the physical machine number where the respective virtual machine will be deployed. Figure 1 illustrates the encoding method in the Lion Optimization Algorithm.

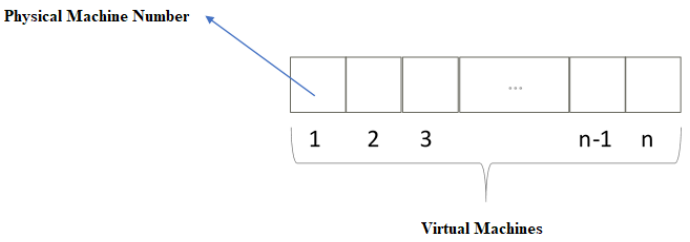

Fig.1. Encoding in the Lion Optimization Algorithm

2.3. Hunting

In each pride, some lionesses hunt in groups to provide food for the pride. These hunters have a specific strategy for surrounding and capturing their prey. Generally, lions follow a somewhat systematic plan during an attack. The roles of lions during ambush are divided into seven different forms, as shown in Figure 2-3. These roles are grouped into left flank, center, and right flank positions. During the hunt, each lioness adjusts her position based on her own location and the positions of other pride members. Consequently, some hunters surround the prey and attack from the front.

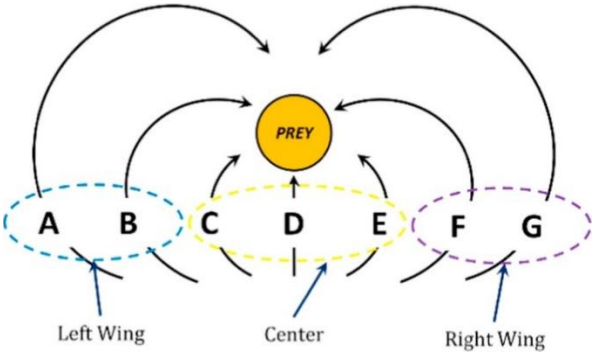

Fig.2. General outline of lion behavior during hunting

Initially, the selected lionesses are randomly divided into three groups: left wing, center, and right wing. The group with the highest fitness value is considered the center, while the other two groups are designated as the wings. Next, a prey is generated at the center of the hunters using equation (12):

$$PREY = \frac{\sum hunters \;\; positions}{number \; of \;\; positions} \tag{12}$$

During the hunt, each hunter approaches the prey according to its group. If a hunter belongs to the center group, the approach is performed using equation (13):

$$Hunter' = \begin{cases} rand(Hunter \;\; , \;\; PREY), & \text{Hunter} < \text{PREY} \\ rand(PREY \;\; , \;\; Hunter), & \text{Hunter} > \text{PREY} \end{cases} \tag{13}$$

Additionally, if the hunter belongs to the left or right wing group, the approach is performed using equation (14):

$$Hunter' = \begin{cases} rand((2 \times PREY - Hunter), PREY), & 2 \times PREY - Hunter < \text{PREY} \\ rand(PREY, (2 \times PREY - Hunter)), & \text{Hunter} > 2 \times PREY - Hunter \end{cases} \tag{14}$$

In the above equations, PREY and Hunter represent the current positions of the prey and hunter, respectively, and Hunter' is the new position of the hunter. During the hunt, if the new position of the hunter is better than its previous position, meaning it has improved its fitness, the prey escapes from the hunter, and the new position of the prey is obtained using equation (15):

$$PREY' = PREY + rand(0,1) \times PI \times (PREY - Hunter) \tag{15}$$

where PI is the percentage improvement in the hunter's fitness [19].

2.4. Moving Towards a Safe Place

In each pride, some lionesses hunt, while the remaining lionesses move towards one of the territory's safe areas. Since each group's territory comprises the best personal positions obtained so far, it helps the Lion Optimization Algorithm (LOA) store the best solutions found during the algorithm's iterations. This information can be valuable and reliable for improving solutions in the LOA. Therefore, the new position of a lioness may be obtained using equation (16):

$$\begin{aligned} &Female\ Lion' = Female\ Lion + 2D \times rand(0,1)\{R1\} + U(-1,1) \times \tan(\theta) \times D \times \{R2\} \\ &\{R1\}.\{R2\} = 0,\ \left\|\{R2\}\right\| = 1 \end{aligned} \tag{16}$$

where Female Lion is the current position of the lioness, D is the distance between the lioness's position and the selected point chosen by the tournament selection operator among the group's territory. {R1} is a vector whose origin is the lioness's previous position, and its direction points towards the selected position. {R2} is perpendicular to {R1} [19].

2.5. Roaming

Each male lion roams within the group's territory for various reasons. To simulate this behavior, R% of the group's territory is randomly selected and visited by the lion. During roaming, if the resident male lion encounters a new position better than its current best position, it updates its best-visited solution. The roaming phase effectively performs a strong local search, helping the LOA explore the vicinity of a solution to improve its fitness. For this purpose, the lion moves x units towards the selected area of the territory, where x is a random number with a uniform distribution:

$$x \sim U(0, 2 \times D) \tag{17}$$

In equation (17), D represents the distance between the lioness's position and the selected area of the territory [19].

2.6. Mating

Mating is a fundamental process that ensures the survival of lions and provides an opportunity for information exchange between members. In each group, Ma% of the lionesses mate with one or more resident male lions. These male lions are randomly selected from the same group as the lionesses to produce offspring. For nomadic lions, the difference is that a nomadic lioness mates with one randomly selected male. The mating operator performs a linear combination of the parents to produce two new offspring. According to the following equations, new cubs are produced after selecting male and female lions for mating [19]:

$$Offspring_j 1 = \beta \times Female\ \ Lion_j + \sum \frac{(1-\beta)}{\sum_{i=1}^{NR} S_i} \times Male\ Lion_j^i \times S_i \tag{18}$$

$$Offspring_j 2 = (1-\beta) \times Female\ \ Lion_j + \sum_{i=1}^{NR} \frac{\beta}{\sum S_i} \times Male\ Lion_j^i \times S_i \quad (19)$$

2.7. Defense

In the pride, when male lions mature, they become aggressive and fight other male lions within the pride. The defeated lions leave their group and become nomadic lions. Conversely, if a nomadic lion is strong enough to try to take over the pride by attacking its resident males, the attacked resident male lion leaves the group and becomes a nomadic lion. The defense operator in the Lion Optimization Algorithm is divided into two main stages:

Defense against newly mature resident male lions

b) Defense against Nomadic Female Lions

2.8. Migration

Inspired by the lifestyle of lions and their migration-related behaviors in nature, where a lion moves from one pride to another or changes its lifestyle, and a resident female becomes nomadic and vice versa, which increases the pride's diversity compared to its previous state. Moreover, the migration of lions and the change in lifestyle facilitate information exchange. In each pride, the maximum number of females is specified as S% of the pride's population. The number of migrating females in each pride equals the excess females of the pride plus 1% of the highest number of females in a pride. When a selected female migrates from the pride and becomes nomadic, new and old nomadic females are classified according to their fitness. Then, the best female among them is randomly selected to fill the vacancy left by the migrating females, maintaining diversity in the population and facilitating information exchange among prides.

2.9. Lion Population Balance

Since there is always a balance in the lion population, at the end of each iteration, the number of living lions is controlled, considering the maximum allowable number of each gender among the nomads. The nomadic lion with the lowest fitness value will die (be removed).

2.10. Convergence

For the stopping condition, as is usually considered in optimization algorithms, the best result is computed when the stopping condition is met. This condition can be defined as processor time, maximum number of algorithm iterations, or number of iterations without improvement.

3. SIMULATION AND EXPERIMENTAL RESULTS

This section presents the simulation and evaluation of the proposed method, which uses the Lion Optimization Algorithm. The following describes how the proposed method is simulated and how various experiments are conducted to evaluate it. All experiments were performed using an Acer computer with a Core i7 2.4 GHz processor and 8 GB of main memory. MATLAB software was used to simulate and evaluate the proposed algorithm. For both the Genetic Algorithm and the Lion Optimization Algorithm, the initial population size was set between 50 to 100. The parameter Ma in the Lion Optimization Algorithm was set at 20%. Additionally, in the Genetic Algorithm, the crossover rate was set at 0.85, the mutation rate at 0.06, the selection operator as roulette wheel, and the crossover operator as two-point crossover. For the Simulated Annealing algorithm, the initial temperature was set at 200,000, the cooling rate at 0.97, and the number of neighbors generated in each iteration was 1.

For the experiments, a data center with 200 to 600 tasks was considered. Each task is assigned to a virtual machine hosted on one of the physical machines. Therefore, the number of virtual machines was also considered between 200 and 600. The computational workload of each task was randomly set between 150 MI (Million Instructions) to 150 MI. Additionally, the capacity of all physical machines was set at 1000 MI. In the first experiment, to examine the impact of the number of virtual machines on the achieved load balance, the proposed approach, along with the Genetic Algorithm and Simulated Annealing Algorithm, was executed with 200 to 600 virtual machines, and the results are shown in Figure 3.

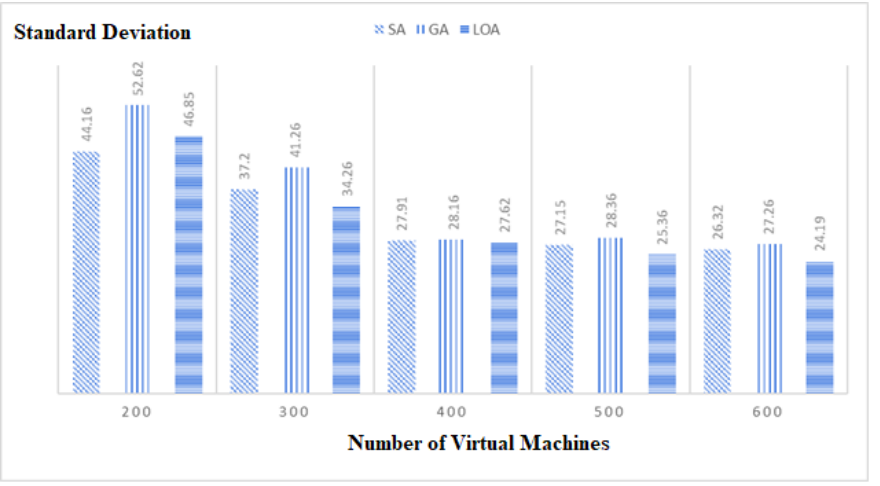

Fig.3. Impact of the number of virtual machines on load balance

Examining the results of this experiment shows that the Lion Optimization Algorithm outperforms both the Genetic Algorithm and the Simulated Annealing Algorithm in four cases, but the Simulated Annealing Algorithm performed better with 200 virtual machines compared to the Genetic Algorithm and the Lion Optimization Algorithm.

In the next experiment, to examine the impact of the number of initial population members on the achieved load balance, the proposed approach was executed with 50 to 100 initial population members, and the results are shown in Figure 4. In this experiment, the number of virtual machines was set to 300.

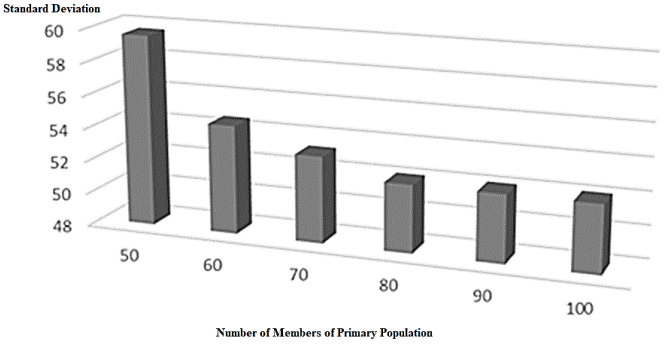

Fig.4. Impact of the number of initial population members on load balance

The results of this experiment indicate that increasing the number of initial population members from 50 improves fitness, or in fact, reduces the standard deviation. However, after 80 initial population members, increasing the number does not affect the standard deviation or the final solution's fitness. Thus, it can be concluded that 80 initial population members are appropriate.

The next experiment is the convergence test. To conduct this, the proposed approach, as well as the Genetic Algorithm and Simulated Annealing Algorithm, were executed for two scenarios with 200 and 300 virtual machines, respectively, and Figures 5 and 6 show the convergence to the final solution.

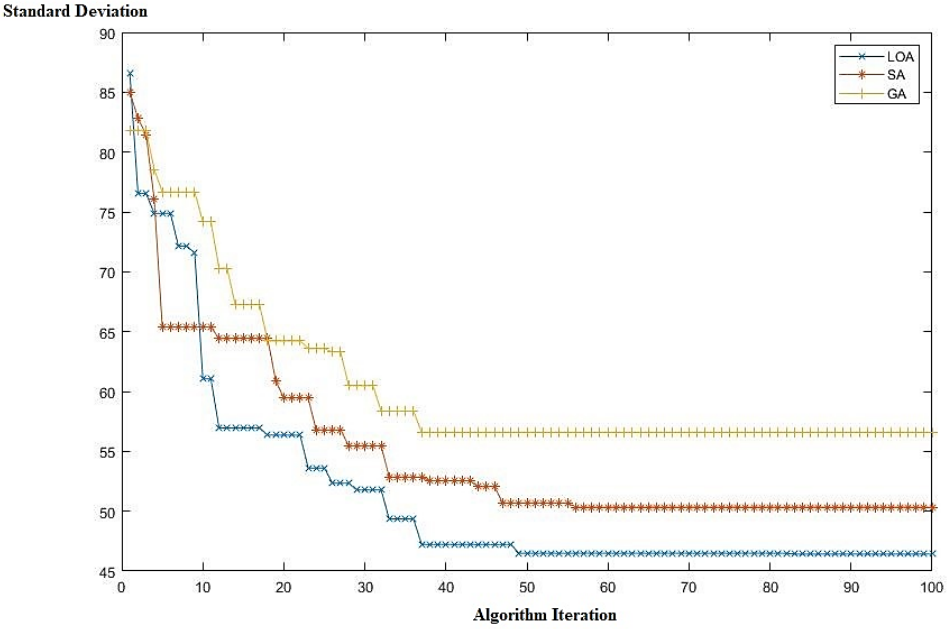

Fig.5. Convergence of the proposed approach compared to GA and SA algorithms (Number of virtual machines: 200)

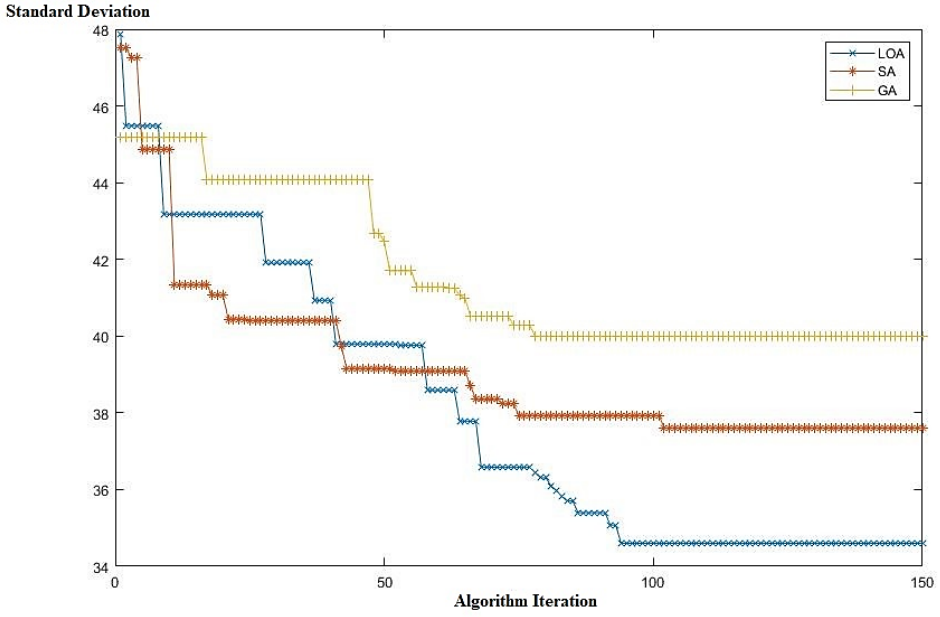

Fig.6. Convergence of the proposed approach compared to GA and SA algorithms (Number of virtual machines: 300)

Examining the results of this experiment reveals that the proposed method for load balancing in cloud computing has a good convergence rate and finds a near-optimal mapping. Additionally, the results show that the load balance achieved by the proposed method is better than that of the Simulated Annealing and Genetic Algorithms.

4. CONCLUSION

Load balancing has always been one of the fundamental challenges in cloud computing and data-centric computing environments. When load balance is not maintained among system nodes, certain nodes may become heavily loaded and experience significant slowdowns, while other nodes have low or no load, leading to decreased system efficiency. Therefore, efficient load balancing algorithms are always needed to improve resource utilization. The choice of load balancing algorithm plays a crucial role in the overall system performance. Hence, a specific algorithm should be used according to the needs. Various methods have been used to maintain load balance in cloud computing environments. In this paper, the Lion Optimization Algorithm was used for load balancing. The goal of the proposed method is to balance the load on physical servers and facilitate resource allocation for virtual machines in cloud computing. Additionally, to provide a comparison, the Genetic Algorithm and Simulated Annealing Algorithm were also simulated. The results obtained from the Lion Optimization Algorithm, Genetic Algorithm, and Simulated Annealing Algorithm indicate that the Lion Optimization Algorithm utilizes resources effectively.

Transparency Statement

The data supporting this study are available upon reasonable request to the corresponding author, subject to ethical and confidentiality considerations.

Acknowledgments

We would like to express our gratitude to all individuals who contributed to this project.

Declaration of Interest

The authors declare that they have no competing interests.

Funding

This research received no specific grant from any funding agency, commercial, or not-for-profit sectors.

REFERENCES

[1] Neupane, D., & Seok, J.-H. (2020). Bearing fault detection and diagnosis using Case Western Reserve University dataset with deep learning approaches: A review. IEEE Access, 8, 93155-93178. https://doi.org/10.1109/ACCESS.2020.2990528

[2] Ghorbanian, S. A., Ahmadi, A., Kakooei, M., Moghimi, A., Mirmazloumi, S. M. I. S., Hamed, S., Moghaddam, A., Mahdavi, S., Ghahremanloo, M., Parsian, S., Wu, Q., Brisco, B., Mirmazloumi, S. M., & Ghahremanloo, M. (2020). Google Earth Engine cloud computing platform for remote sensing big data applications: A comprehensive review. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 13, 5326-5350. https://doi.org/10.1109/JSTARS.2020.3021052

[3] Berg, G., Rybakova, D., Fischer, D., Cernava, T., Vergès, M. C., Charles, T. C., Chen, X., Cocolin, L., Eversole, K., Corral, G., Kazou, M., Kinkel, L., Lange, L., Lima, N., Loy, A., Macklin, J., Maguin, E., Mauchline, T., McClure, R., Mitter, B., Ryan, M., Sarand, I., Smidt, H., Schelkle, B., Roume, H., Kiran, G. S., Selvin, J., de Souza, R. S. C., Overbeek, L. V., Singh, B., Wagner, M., Walsh, A. M., Sessitsch, A., & Schloter, M. (2020). Microbiome definition re-visited: Old concepts and new challenges. Microbiome, 8. https://doi.org/10.1186/s40168-020-00875-0

[4] Boehler, C., Carli, S., Fadiga, L., Stieglitz, T., & Asplund, M. (2020). Tutorial: Guidelines for standardized performance tests for electrodes intended for neural interfaces and bioelectronics. Nature Protocols, 1-22. https://doi.org/10.1038/s41596-020-0389-2

[5] Cao, K., Liu, Y., Meng, G., & Sun, Q. (2020). An overview on edge computing research. IEEE Access, 8, 85714-85728. https://doi.org/10.1109/ACCESS.2020.2991734

[6] Zimmermann, A., Wunderlich, J., Müller, L., Buchner, G., Marxen, A., Michailos, S., Armstrong, K., Naims, H., McCord, S., Styring, P., Sick, V., & Schomäcker, R. (2020). Techno-economic assessment guidelines for CO2 utilization. Frontiers in Energy Research, 8. https://doi.org/10.3389/fenrg.2020.00005

[7] Van Meerbeek, B., Yoshihara, K., Van Landuyt, K. L., Yoshida, Y., & Peumans, M. (2020). From Buonocore's pioneering acid-etch technique to self-adhering restoratives. A status perspective of rapidly advancing dental adhesive technology. The Journal of Adhesive Dentistry, 22(1), 7-34. https://doi.org/10.3290/j.jad.a43994

[8] Kurdi, B., Alshurideh, M., Salloum, S., Obeidat, Z., & Al-Dweeri, R. (2020). An empirical investigation into examination of factors influencing university students' behavior towards e-learning acceptance using SEM approach. International Journal of Interactive Mobile Technologies, 14, 19-41. https://doi.org/10.3991/ijim.v14i02.11115

[9] Ayaz, M., Pasha, M. F., Alzahrani, M., Budiarto, R., & Stiawan, D. (2020). The Fast Health Interoperability Resources (FHIR) standard: Systematic literature review of implementations, applications, challenges and opportunities. JMIR Medical Informatics, 9. https://doi.org/10.2196/21929

[10] Wang, Z., Du, Y., Wei, K., Han, K., Xu, X., Wei, G., Tong, W., Zhu, P., Ma, J., Wang, J., Wang, G., Yan, X., Xiang, J., Huang, H., Li, R., Wang, X., Wang, Y., Sun, S., Suo, S., Gao, Q., & Su, X. (2022). Vision, application scenarios, and key technology trends for 6G mobile communications. Science China Information Sciences, 65. https://doi.org/10.1007/s11432-021-3351-5

[11] Velte, T., Velte, A., & Elsenpeter, R. (2009). Cloud computing: A practical approach. McGraw-Hill, Inc.

[12] Liu, Z., & Wang, X. (2012). A PSO-based algorithm for load balancing in virtual machines of cloud computing environment. In International Conference in Swarm Intelligence (pp. 142-147). https://doi.org/10.1007/978-3-642-30976-2_17

[13] Acharya, J., Mehta, M., & Saini, B. (2016). Particle swarm optimization based load balancing in cloud computing. In 2016 International Conference on Communication and Electronics Systems (ICCES) (pp. 1-4). https://doi.org/10.1109/CESYS.2016.7889943

[14] Al Nuaimi, K., Mohamed, N., Al Nuaimi, M., & Al-Jaroodi, J. (2012). A survey of load balancing in cloud computing: Challenges and algorithms. In Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on (pp. 137-142). https://doi.org/10.1109/NCCA.2012.29

[15] Nishant, K., Sharma, P., Gupta, V., Singh, K., Nitin, & Rastogi, R. (2012). Load Balancing of Nodes in Cloud Using Ant Colony Optimization. In 2012 UKSim 14th International Conference on Computer Modelling and Simulation (pp. 3-8). https://doi.org/10.1109/UKSim.2012.11

[16] Zhang, Z., & Zhang, X. (2010). A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation. In Industrial Mechatronics and Automation (ICIMA), 2010 2nd International Conference on (Vol. 2, pp. 240-243). https://doi.org/10.1109/ICINDMA.2010.5538385

[17] Makasarwala, H. A., & Hazari, P. (2016). Using genetic algorithm for load balancing in cloud computing. In 2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI) (pp. 1-6). https://doi.org/10.1109/ECAI.2016.7861166

[18] L. D. B., & Krishna, P. V. (2013). Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied Soft Computing, 13(5), 2292-2303. https://doi.org/10.1016/j.asoc.2013.01.025

[19] Yazdani, M., & Jolai, F. (2016). Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm. Journal of Computational Design and Engineering, 3(1), 24-36. https://doi.org/10.1016/j.jcde.2015.06.003

[20] Ramezani, F., Lu, J., & Hussain, F. K. (2014). Task-based system load balancing in cloud computing using particle swarm optimization. International Journal of Parallel Programming, 42(5), 739-754. https://doi.org/10.1007/s10766-013-0275-4