Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: performance evaluation

Article *in* The Journal of Supercomputing · January 2022

DOI: 10.1007/s11227-021-03810-8

3 authors, including:

Seyed Salar Sefati
Polytechnic University of Bucharest
28 PUBLICATIONS **383** CITATIONS

SEE PROFILE

Maryamsadat Mousavinasab
Politecnico di Torino
1 PUBLICATION **86** CITATIONS

SEE PROFILE

The Journal of Supercomputing
https://doi.org/10.1007/s11227-021-03810-8

Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: performance evaluation

SeyedSalar Sefati[1] · **Maryamsadat Mousavinasab**[2] · **Roya Zareh Farkhady**[3]

Accepted: 11 April 2021

Abstract
The introduction of cloud computing has brought about significant developments in information technology. Users can benefit from the multitude of cloud technology services only by connecting to the internet. In cloud computing, load balancing is the fundamental issue that has challenged experts in this research area. Load balancing helps increase user satisfaction and enhance systems' productivity through efficient and fair work assignments between computing resources. Besides, maintaining a load balancing among resources would be difficult because the resources are usually distributed in a heterogeneous way. Many load-balancing methods try to solve this problem by the metaheuristics algorithm, and each of them attempted to enhance the operation and efficiency of systems. In this paper, Grey wolf optimization (GWO) algorithm has been used based on the resource reliability capability to maintain proper load balancing. In this method, first, the GWO algorithm tries to find the unemployed or busy nodes and, after discovering this node, try to calculate each node's threshold and fitness function. The results of simulation in CloudSim showed that the costs and response time in the proposed method are less than the other methods, and the obtained solutions are ideal.

Keywords Cloud computing · Load balancing · Grey wolf optimization algorithm · Reliability

1 Introduction

Cloud computing is a model based on giant computer networks such as the internet, which provides a new model to present, consume, and deliver information technology (IT) services and other shared resources using the network [1]. With the aid

✉ SeyedSalar Sefati
Stu.salarsefati@iaut.ac.ir

Extended author information available on the last page of the article

Published online: 19 May 2021

of cloud computing and services, users can have access to applications from all over the world [2]. In such a method, users access the service based on their needs, regardless of the cloud's and the user's location. The cloud model is commercially supported by major companies such as Google, Amazon, and Microsoft. This support is in the form of using computer resources consisting of hardware and software that are offered as a service on the network [3]. Current cloud computing systems offer a wide range of virtual services on demand [4]. Cloud computing is meant by the internet-based provision of hardware and software services, and users pay for their usage [5]. In fact, instead of installing their applications on their PCs, users can receive them from the internet in the form of applications and installed on a set of network servers. In other words, only by access to a mobile device, computer, etc., they can be connected quickly and get access to the cloud services [6].

Load balancing is the efficient and fair assignment of work to computing resources to maintain the load satisfaction (i.e., processor load, the used memory, delays, or the network load) of users and increase the rate of resource productivity [7]. When some virtual machines (VMs) have higher load volumes than others, the system's efficiency is reduced [8]. Thus, the cloud would be restricted, and the time of completing different tasks would be affected. In addition, when the cloud system is faced with a more significant number of demands, the system has to distribute them among its resources. If the cloud system can provide users with efficient and accessible resources to complete such tasks, the system performance will enhance [9]. In a cloud system, various computing resources exist to facilitate responding to users' demands. Thus, the proper selection of resources by considering the characteristics of tasks will enhance a system's efficiency; therefore, a mechanism is required to select appropriate resources in responding to users' demands [10].

In the current study, a method of selecting the best services has been proposed to obtain optimal quality of service (QoS), overcoming limitations of connection and setting the QoS criteria [11]. QoS is among the most important topics in cloud computing problems. The proposed method has used the hybrid form of the Grey wolf optimization (GWO) algorithm and QoS. GWO algorithm has been inspired by the behavior of grey wolves in nature [12] where they start to search for prey in their environment, and load balancing has used this concept. In the current study, first, the nodes should select the cluster head (CH). For this purpose, the node with the highest number of neighboring is chosen as the CH node. Then, the threshold of nodes is calculated to check the virtual machine's load; if the load of the virtual machine is more than the threshold, the virtual machine has an overload, but if the amount of load is less than the threshold, the virtual machine is low load. After finding the suitable node and evaluating the node, the alpha and beta wolves attack the prey and select it as the appropriate node. Using this method brought about more favorable results compared to other methods, the results have been summarized below:

- Better hybridization of QoS to respond to users' needs.
- Reducing response time and the costs of selecting VMs and consequently enhancing load balancing.
- Conducting a series of investigations to evaluate the performance of the method through various ways of maintaining load balancing.

The rest of this article has been organized in the following manner: Section 2 deals with the related literature, while the proposed algorithm has been presented in Sect. 3. The set of experimental data has been shown in Sect. 4, and Sect. 5 has given the results of the analyses. Finally, Sect. 6 has been devoted to discussion and future works.

2 Related work

Kruekaew and Kimpan [13] proposed the non-concentrated artificial bee colony (ABC) algorithm load-balancing technique. It was inspired by nature and stated that maintaining load balancing in cloud is increasing or decreasing demand. The assigned servers dynamically regulate users' demands. These servers have been classified as virtual servers, and each virtual server has its virtual service queues. Like the movements that honeybees show in their dances, each server calculates the required benefits by processing a request and demand from the queue. A measure of such benefits is the time spent by the processor to process a request. Here, the dance part of honeybees is similar to the advertisement board. After processing a request, the server can select benefits for the plausible advertisement boards. In addition, it can investigate the advertisements (watching the dance) and provide services (hence, the scout behavior). If such benefits are calculated and high compared to the overall benefits of a colony, the server remains in the same virtual server and becomes a plausible advertisement. Otherwise, the servers take the scout or explorer roles. The assessment outcomes indicated that the mentioned method decreases makespan and response time. Nevertheless, the mentioned technique has low reliability and high cost.

Devaraj et al. [14] proposed the improved particle swarm optimization algorithm (PSO). In this algorithm, it could provide resources appropriate to users' tasks in an efficient manner. To assist the PSO algorithm, a simulation algorithm was implemented, as well. Using the latter algorithm, it was not only helpful in preventing the algorithm from being trapped in local optima, but it also increased the convergence rate of PSO. The PSO algorithm works by selecting the global best (gbest) particle with a small distance of point to a line. With the application of minimum distance from a point to a line, the best particle candidates could be selected. This method has a good performance in throughput and reliability and running time. However, it has limited functionality in energy consumption and service monitoring.

Lilhore et al. [15] proposed an exploratory scheduling method based on the hybrid particle swarm optimization (HPSO) algorithm to balance the load within a system. This method attempted not only to maintain load balancing, but also to reduce the longest time of task completion among all processors within the cloud. There, it was assumed that independent tasks and the cloud environment have heterogeneous resources with different processing capabilities. Thus, the time taken to process tasks may be changed according to the scheduling of tasks on different resources. The results showed that this method was able to reduce running time and increase the rate of exploiting resources. However, it has limited functionality in reliability and service monitoring.

Kokilavani and Amalarethinam [16] proposed an algorithm that started with a set of unassigned tasks. First, the minimum finish time was found for each job. Then, the smallest one that was the shortest time among all tasks on each accessible resource was selected from among such minimum periods. Then, the task was scheduled on the related machine according to the minimum time. After that, the running time for all the other tasks of the machine was updated by adding the running time of the assigned task, and the assigned task was eliminated from the list of tasks assigned to the machine. This procedure was repeated until all the tasks were assigned to resources. In this paper, the authors found low energy consumption and overload. However, this algorithm does not pay attention to the reliability and response time.

Goyal and Singh [17] proposed a load-balancing mechanism based on the theory of complex networks and the ant colony optimization (ACO) algorithm in cloud computing. The method was aimed at overcoming the complexity and issues of dynamic load balancing and used the open scale and small world features of complex networks to maintain better load balancing. This method improved many dimensions of the related ACO algorithms and has been proposed to maintain load balancing in distributed systems; in addition, the method is able to overcome heterogeneity. This mechanism is tuned to dynamic environments, performs greatly in error tolerance, and has acceptable scalability since it assists system efficiency. Nevertheless, it has a high response time and cost.

Makasarwala and Hazari [18] give a genetic algorithm (GA)-based approach for load balancing in the cloud. For population initialization, priority of a request is considered based on their time. Population initialization is done based on priority. The priority of a request is being calculated based on the time, and it is calculated from the length of the job. In this paper, permutation encoding is used as chromosome representation. That is decimal numbers are used for gene representation. Fitness function is the base part for the evolution of algorithm. This function is decision making in the algorithm for selecting the algorithm. The chromosomes having good fitness are selected for generating child. This method has high efficiency in response time but does not have good performance in reliability and availability.

3 The proposed model

In the environment of cloud computing, a large scale of data centers and users has been distributed all over the world [19]. When a large number of user requests simultaneously reach the cloud, the cloud has to organize them efficiently and provide them with proper services [20]. In fact, the cloud must distribute the load among resources fairly. Load balancing is a major challenge in cloud computing and requires the working load to be distributed equally across all nodes to maintain user satisfaction [21]. Numerous load-balancing algorithms have been proposed for distributed, grid, and cloud environments. The current study is aimed at maintaining load balancing based on QoS. Here, a new method has been proposed for load balancing based on QoS to decrease the response time and cost spent finding proper resources. The proposed method is capable of maintaining load balancing based on QoS in a cloud-computing environment through using the GWO algorithm.

3.1 Statement of the problem

Consider cloud c, which includes n physical machines or any physical machine comprised of M VMs:

$$C = \{\mathrm{PM}_1,\ \mathrm{PM}_2,\ \mathrm{PM}_n\} \tag{1}$$

In addition, any physical machine is comprised of several VMs in the following manner:

$$\mathrm{PM}_n = \left\{\mathrm{VM}_1,\ \mathrm{VM}_2, \ldots, \mathrm{VM}_m\right\} \tag{2}$$

In the first, VM_1 indicates the first virtual machine, while VM_m is the last (VM). Similarly, if users comprise the cloud and the user function can be indicated in the following manner:

$$U_i = \left\{T_1,\ T_2 \ldots T_j\right\} \tag{3}$$

The main goal of this procedure is to minimize the costs and energy and maximize the exploitation of resources and QoS, as well as to maintain a balanced load across all VMs within a cloud environment. To attain a proper planning procedure, loads should be balanced. Without it, the system will spend the maximum time and energy on executing its functions. To overcome this problem, an efficient and balancing multi-purpose procedure has been proposed in this study based on the GWO algorithm. Figure 1 illustrates the proposed load-balancing system.

3.2 Virtualization in the cloud

The most secure way to combine service providers is using virtualization technology. Virtualization makes it possible to operate several VMs on any physical hardware. Each one of those VMs can have their own operating system and execution

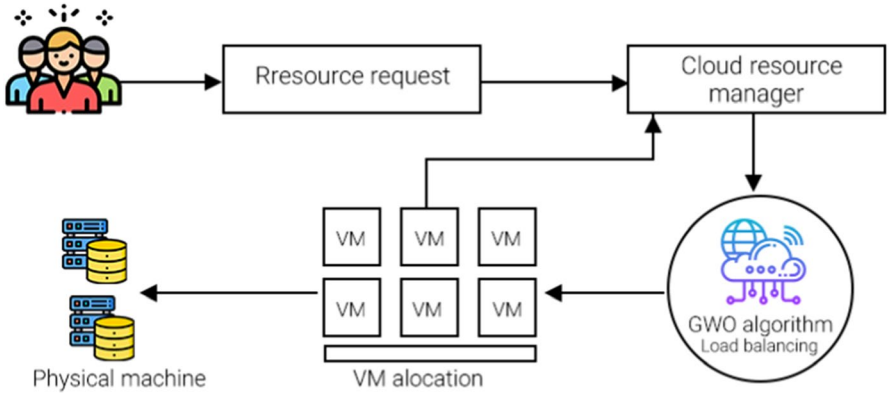

Fig. 1 The proposed load-balancing system

[22]. In this manner, using virtualization makes it possible to apply several incongruous operating systems to execute one application simultaneously. Virtualization is creating a virtual copy of anything like an operating system, calculating server device, storage devices, or network devices [23]. Many people equate virtualization with cloud computing, though it is completely wrong. Virtualization is usually considered as one of the basic technologies in cloud computing. The significant difference between the cloud and virtualization is that the cloud aims to gain control of resources [24]. This depends on the type of infrastructure, the context, and software as a service. On the other hand, virtualization aims to optimize the use of resources and can have numerous meanings for different people according to its type of usage. Such uses can include the following items:

- Server virtualization.
- Storage virtualization.
- Network virtualization.
- Service virtualization.

Figure 2 shows that server, network, storage, and service forms of virtualization can simultaneously be present in a data center, and this is possible through applying virtualization management. Other types of virtualization are possible, but the use of virtualization technology in data centers is still in its infancy.

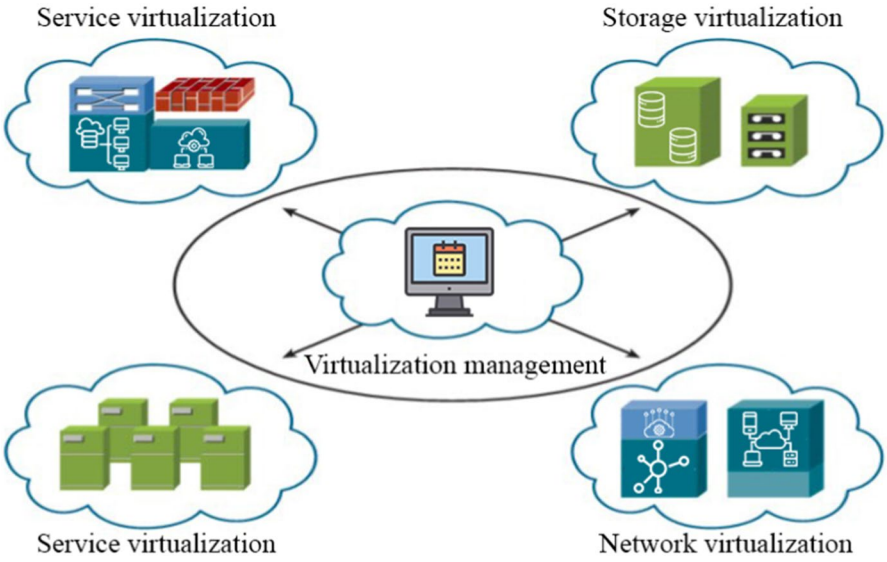

Fig. 2 Types of virtualization

3.3 The immigration of virtualization machines

A technique used to optimize the management of resources in cloud data centers is the immigration of VMs [25]. Such immigration increases the flexibility and scalability of cloud data centers. Immigration in this sense means transferring the state of memory discs and processors from a physical machine to another one. This is performed with different goals such as managing energy consumption, load balancing and distribution, durability against failure, reducing response time, increasing service quality, and repairing and maintaining servers. In resource management, immigration is conducted with two major goals of load balancing and server combination.

Load balancing The goal is to distribute processing load equally across the physical servers of a data center and prevent high levels of difference in using resources in physical servers.

Server combination In the process of establishing VMs, optimization algorithms are usually applied to decide whether a VM should be used on a particular physical host or not. There, the goal is to minimize the overall number of physical hosts and consequently reduce energy consumption. In dynamic methods of resource allocation, decisions regarding load balancing or combining service providers are conducted through monitoring the performance of VM and the level of using resources (Fig. 3).

3.4 The parameters of service quality

Makespan and resource utilization This is the comparison between the average time taken to complete the processing and the time taken to complete tasks. The parameter attempts to minimize service quality in cloud computing and is the opposite of utilization that seeks to maximize service quality [26]. According to Eq. 4, the utilization of a VM is defined as a part of the time taken to complete tasks (i.e., makespan) where the VM is busy. To formulate the problem, the tasks were determined as follows: $\{T_1, T_2, T_3, \ldots, T_n\}$ set where members are

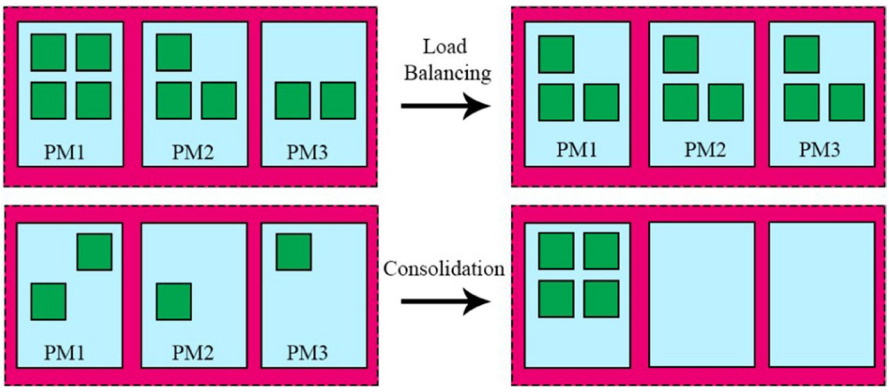

Fig. 3 Migration with the aim of load balancing and combining servers

not independent, and the VMs in the form of the $\{VM_1, VM_2, VM_3, \ldots, VM_m\}$ set that is independent. If the time taken to process task T_i in VM_j is considered as PT_{ij} and the finishing time VM_j is shown as CT_j, the following 5 equations can be used to define the overall utilization rate, where M indicates the number of VMs. The goal is to maximize the time span for utilizing resources.

$$\text{Makespan} = \max\{CT_j | J = 1, 2, \ldots, M\} \tag{4}$$

$$\text{Utilization } VM_i = \frac{\sum_{(i=1)}^{n} PT_{ij}}{\text{makespan}} \tag{5}$$

$$\text{Resourceutilization} = \frac{\sum_{i=1}^{n} \text{Utilization}_{VM_j}}{m} \tag{6}$$

Costs It refers to the money that a user pays per request to a VM and is calculated according to the memory, processes, VM, and bandwidth that have been used. This rate can be obtained using Eq. 7 [27]:

$$\text{Cost} = \sum_{i=1}^{K} C_i * T_i \tag{7}$$

where K is the number of VMs assigned to the user requests, C_i is the costs of a VM, and T_i is the time when a user can make use of a VM.

Response time It refers to the capacity of a service to provide the required service under any circumstances for the specified period and is evaluated using the following equation [10]:

$$RA_{RK} = \frac{RES_K}{REC_K} \tag{8}$$

RES_k shows the work sent to R_k that is redistributed in a specific time and is calculated in terms of milliseconds. In addition, REC_k shows the number of collected requests.

3.5 Standardizing the values of QoS parameters

Different parameters of QoS in any particular service are measured using different units. On the other hand, all parameters have to be measured using a single scale to measure the objective function. Thus, the values of all parameters related to the QoS have to be normalized on a single scale. In fact, normalizing the QoS parameters makes it possible to obtain homogeneous measures of them. The general approach for this purpose is normalizing all the values of all parameters in the range of 0–1. The mentioned parameters can be classified into two groups: maximization and minimization. Equations 9 and 10 indicate the normalization

relationships for the maximization and minimization parameters, respectively [10].

$$N_{CS.Q}i = \begin{cases} \frac{Q^i_{\max} - CS.Q^i}{Q^i_{\max} - Q^i_{\min}} & Q^i_{\max} \neq Q^i_{\min} \\ 1 \end{cases} \tag{9}$$

$$N_{CS.Q}i = \begin{cases} \frac{CS.Q^i - Q^i_{\min}}{Q^i_{\max} - Q^i_{\min}} & Q^i_{\max} \neq Q^i_{\min} \\ 1 \end{cases} \tag{10}$$

In the above equations, $CS.Q^i$ shows the value of the ith parameter in the candidate service CS, and $Ncs.Q^i$ is its normalized value. In addition, $Q^i{}_{\max}$ and $Q^i{}_{\min}$ are the maximum and minimum values of the ith parameter among all services.

3.6 Fitness function

The important goals involved in the problem of load balancing based on the QoS are dealing with the limitations set by users, finding free nodes and assigning tasks to them, and optimizing a fitness function. The fitness function has to optimize the values of the QoS parameters for the created combined services. The positive and negative parameters' inclinations and their impact on the evaluation function are opposite. To overcome this problem, all parameters of the QoS have to be normalized in a proper mode (using the function of the previous section) so that their positive and negative impacts can be converged in one direction. The fitness function for a solution can be defined using Eq. 11.

$$\text{Fitness} = \sum_{i=1}^{4} W_i * Q_i \tag{11}$$

where W is positive with that indicates the importance of each parameter involved in the QoS and is determined by users.

3.7 The Grey wolf optimization algorithm

The purpose of this article is to present a new version of the Grey wolf meta-heuristic algorithm to optimize the search capability of wolf hunting [28]. This algorithm has been inspired by the social behavior of grey wolves, which are members of the coyote family and live in groups of 5–12 [29]. Each grey wolf starts from the alpha (α), which can be either male or female. The alpha wolf that is known as the leader of the pack orders other wolves and takes the major decisions of the pack, such as sleeping and waking up. The alpha wolves obtain natural skills to organize their pack and monitor their pack in a proper manner (Table 1).

After the alpha wolf, there is the beta (β) grey wolf. The beta wolves are either male or female and are ranked second in the hierarchy of Grey wolves. The alpha wolves

Table 1 Stages from discovering until exploiting the game

First stage	Detecting, following, and approaching the game
Second stage	Proceeding, surrounding, and harassing the game
Third stage	Attacking the game

make decisions, while the beta wolves help them to transfer the instructions to the lower levels of the wolves. In addition, the alpha ones to get feedback use the beta wolves. If any alpha wolf dies, one of the beta ones replaces it. The third rank is occupied by the delta (δ) wolves, which are classified as spies, guards, hunters, and supporters. The delta wolves help to protect the whole pack and keep on the boundaries to make suitable reactions when the pack is faced with dangerous situations. The delta hunters provide others with food, while the delta guards look after the old, weak, or sick wolves in the pack. In cases of the death of beta wolves, the senior delta wolf gets promoted to its status. Alphas are considered the best solution in GWO. Beta is considered the second, while the delta and omega rank next. α, β, and δ are used as guides for the target.

During the hunting process, grey wolves circle around their game and surround it. The mathematical model of this behavior has been presented below.

$$D = \left|C.X_P(t) - X(t)\right| \tag{12}$$

$$X(t+1) = X_P(t) - A.D \tag{13}$$

where T indicates the current iteration, A and D are coefficient vectors, X_p is the location of the hunt, and X is the vector for the location of the grey wolf.

$$A = 2a.r_1 - a \tag{14}$$

$$C = 2.r_2 \tag{15}$$

The alpha is the most senior or strongest wolf among the packs intended for hunting. Sometimes, such a hunt is conducted by betas and deltas. In mathematics, the best three solutions are saved, and they are transferred to update locations based on the remaining omega wolves. Such tasks are conducted according to the following equations:

$$D_a = \left|C_1 * X_a - X\right| D_{\hat{\omega}} = \left|C_2 * X_{\hat{\omega}} - X\right| \tag{16}$$

$$D_\delta = \left|C_3 * X_\delta - X\right| \tag{17}$$

$$X_1 = X_a - A1 * \left(D_a\right) * X_2 = X_{\hat{\omega}} - A2 * \left(D_{\hat{\omega}}\right) \tag{18}$$

$$X_3 = X_\delta - A3 * \left(D_\delta\right) \tag{19}$$

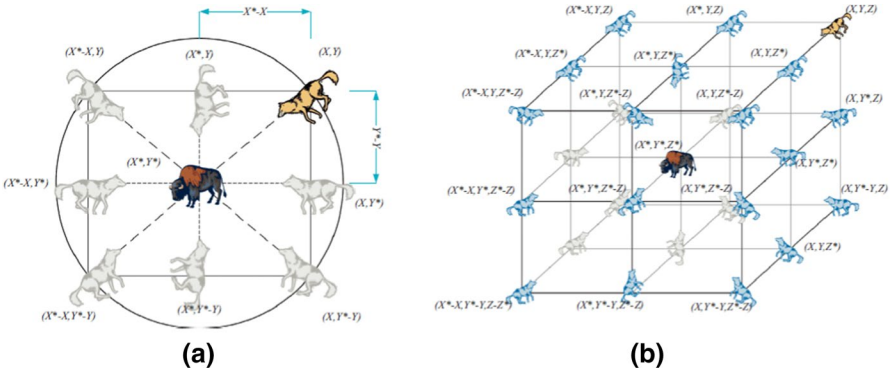

Fig. 4 The location and placement vector of D3 and D4

$$X(t+1) = \frac{X_{1+X_{2+X_3}}}{3} \tag{20}$$

A concept similar to the one illustrated in Fig. 4 can be extended to an N-dimensional search space where grey wolves move in cubes to obtain the best solution. The pseudo-code 1 shows the GWO algorithm.

The pseudo-code of the GWO algorithm 1

1 Set the initial values of the population size n, parameter a, coefficient vectors A and C, and the maximum number of iterations Maxiter
2 Set t = 0
3 for (i = 1: n) do
4 Generate an initial population Xi (t) randomly
5 Evaluate the fitness function of each search agent (solution) f (Xi)
6 End for
7 Assign the values of the first, second, and the third best solution Xα, Xβ, and Xδ, respectively
8 Repeat
9 for (i = 1: n) do
10 Update each search agent in the population as shown in Eq. (20)
11 Decrease the parameter a from 2 to 0
12 Update the coefficients A, C as shown in Eq. (14) and (15), respectively
13 Evaluate the fitness function of each search agent (vector) f(Xi)
14 End for
15 Update the vector Xa,Xb and Xδ
16 Set t = t + 1
17 Until (t ≥ Maxiter). {Termination criteria are satisfied}
18 Reduce the best solution Xα

3.8 The proposed method

Obtaining the application In the proposed method, the goal is to get a real-time application. In such applications, the time for running tasks is essential for the user. The user sets a time limit for the task to be performed and desires that the task be completed within that period. Thus, the time limit is obtained as input from the user.

Selection of the CH First, one node has to be selected as the CH. For this purpose, the nodes with the highest number of neighboring nodes are chosen as the CH. Wolves are originated from this virtual node machine and are randomly distributed in the neighboring VMs.

The start of the GWO GWO algorithm starts with the alpha wolves from the CH node. The wolves' movement toward the hunt: First, the alpha wolves, which are the leader and dominate the pact, and the beta wolves that help in making decisions start to move around other nodes.

Finding nodes After finding several appropriate nodes based on the estimated loads, the nodes are recommended to the server and are saved as candidate nodes.

Calculating the threshold A threshold level has been defined to investigate the load of each VM. If the load is higher than the threshold, the VM has overload; on the other hand, if the load is below the threshold, VM is under load.

If $(L_{vm\,j} \leq \Psi)$

(VM) is under loaded

Else if $(Lvmj \geq \Psi)$

(VM) is over loaded

End

The load threshold of each VM can be calculated as follows:

$$\psi = AL + \sigma \tag{21}$$

where AL indicates the mean of VMs of a data center and σ indicates the standard deviation of the load mean of the VM in a data center; thus, if the load of each machine is higher than the threshold, that VM is among the nodes with overload machines, and no task is assigned to it. Otherwise, the machine is under load. Now, since the lowest loaded machine is selected from the underload VMs, a fitness factor of load balancing is calculated for each one of the underload VMs:

$$LB = 1 - L_{vmj} \tag{22}$$

Thus, the higher the LB of a VM, the more will be the possibility of that machine to perform the particular application.

Estimating the time for performing the tasks After the algorithm was applied together with the user's time limit, the algorithm checks whether the time that will take the application to perform the task is within the user's specified time or not. For this purpose, the time taken to run the applications must be estimated, and the time for running an application on a VM can be calculated as the weight (w) of the application (i) divided by the speed of the processor (j) working on the VM. The following equation is used to calculate the time taken for an application to work on a VM:

$$T(I\,,\,j) = \frac{w_i}{SP_j} \tag{23}$$

where W_i is the weight of the application and SP_j is the speed of the processor.

$\mathrm{Min}_{rr} < T < \mathrm{Min}_{TT}$ After the time for running the application by each VMs is estimated, it has to be checked whether the resulting time is within the user's time limit or not. In cases that the estimated time is within the set limit, the QoS of the application is calculated for these resources; otherwise, the QoS is considered 0. In other words, no task is assigned to that VM.

If $(Min_{tt} \leq T\,(I,j) \leq Max_{tt})$

Reliability assessment

Else

Reliability $= 0$

End

The mechanism for evaluating the QoS The QoS of a system refers to the possibility of working properly and free of error for a specified time according to the predetermined conditions. The QoS of a program can also be defined as the possibility of all resources remaining efficient until all the tasks assigned to them are completed. The QoS of any (VM) can be calculated using Eq. (24):

$$R = \mathrm{e}^{-\lambda\left[\frac{w_i}{w_{vm}}\right]} \tag{24}$$

where λ is the number of failures in a specified time, W_i is the weight of the ith task, and W_{vm} is the weight of the current tasks in a VM.

Calculating the load of nodes.

The whole tasks assigned to a VM are called "Load," and the load for each VM can be calculated using the following formula:

$$L_{vmj} = \frac{\mathrm{num}_{\mathrm{task}}}{\mathrm{rate} - s_{vnj}} \tag{25}$$

where L_{vmj} is the load of the VM (j), Num_{task} is the number of tasks; Rate S_{vmj} is the rate of service in the jth (VM). Now, it has to be determined whether this VM is among the overloaded or underloaded machines. If the former is the case, no task is assigned to the machine. In addition, the average load of VMs in a data center is calculated using the following formula:

$$\text{AL} = \frac{1}{m}\sum_{i=1}^{m} L_{vmj} \tag{26}$$

M number of VMs.

Thus, the standard deviation of the load of VMs in a data center can be calculated as follows:

$$\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(L_j - AL)^2} \tag{27}$$

where σ is the standard deviation of the load in the *VMs* of a data center, L is the average load of the VMs in a data center, L_j is the load of the jth VM, and M is the number of VMs.

Surrounding by the wolves and evaluating the possibility of the selection of a node: In the proposed method, k Wolves select an under loaded VM with proper QoS for the purpose of assigning tasks to it using the following formula:

$$p_t^k(t) = \begin{cases} 0 \\ \dfrac{\left[X_p(t)\right]^{\alpha}\left[R_j\right]^{\beta}\left[LB_j\right]^{\gamma}}{\sum\left[X_p(t)\right]^{\alpha}\left[R_k\right]^{\beta}\left[LB_k\right]^{\gamma}} \end{cases} \tag{28}$$

where X_P (t) shows the values of the location vector for jth virtual at the moment T, R_J is the QoS for the jth VM, LB_J is the fitness factor of the load of the jth VM.

Selecting a suitable node and attacking the game: After evaluating and confirming a node, the alpha and beta wolves attack the game and select that as the suitable node. Figure 5 depicts the proposed method flowchart.

4 The simulation environment

The CloudSim was applied to simulate and evaluate the proposed algorithm. It is an open-source and free toolkit to simulate cloud computing scenarios. It has been designed in the CLOUDS lab of the department of computer and software engineering, the University of Melbourne [30]. The CloudSim toolkit provides major classes to define data centers, VMs, applications, users, calculation resources, and policies to manage various system sections (e.g., scheduling). Users can combine these components to evaluate new strategies in applying clouds. In addition, CloudSim can be used to evaluate the effectiveness of strategies from different points of view—from cost–benefit to accelerate the time taken to run the application [31]. There is no limitation in using CloudSim; classes can be expanded or replaced, and

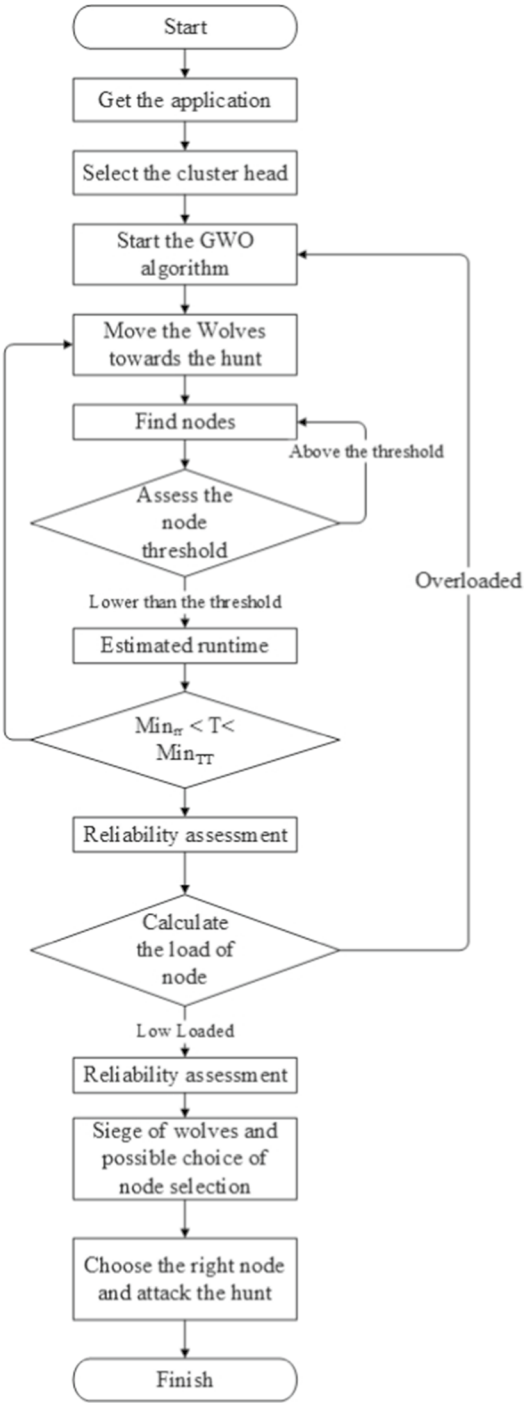

Fig. 5 Flowchart of proposed method

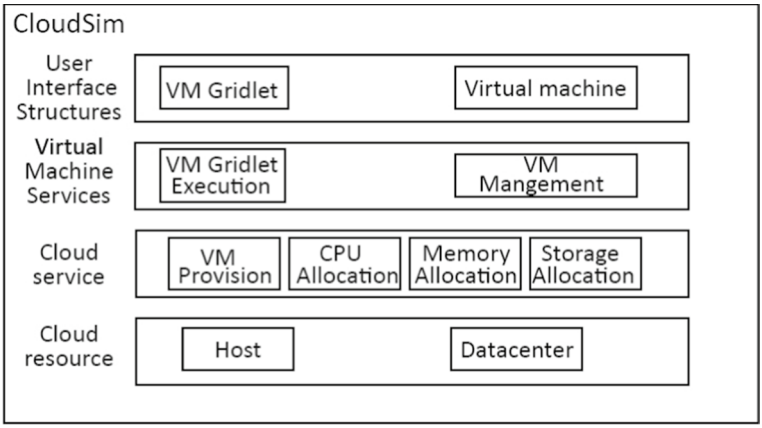

Fig. 6 The architecture of CloudSim

new policies can be added and written. CloudSim is like building blocks by the use of which you can make your simulation environment. Thus, Cloudsim is not a ready-made solution, which you can set the parameters and then collect the results to use in your project. Figure 6 illustrates different layers of CloudSim. In the lowest layer, there are cloud resources such as hosts and data centers that are executed during the simulation period. Over that layer are the cloud services including the assignment of processors, memory, bandwidth, etc. The two upper layers include VM services and the user interface structures. To conduct all the experiments, a Lenovo PC with a 4.2 GHz Core i7 CPU and 16 gigabytes of the main memory was used.

4.1 Simulation data and parameters

For the purpose of the current study, CloudSim was applied for the SaaS-level simulation of cloud computing. This simulator makes it possible to simulate a virtual environment and supports the provision of the required resources. As mentioned before, the proposed method is aimed at presenting an efficient GWO-based scheduling algorithm for cloud computing environment to reduce the makespan and QoS. Several experiments were conducted to analyze the proposed method, which has been described later in this section. In the first experiment, a cloud-based data center comprised of three hosts was defined. Each of these hosts was able to support virtualization and sharing their resources among several VMs. The software details of such hosts are presented in Table 2, and the simulation parameters of the GWO algorithms are presented in Table 3. Since the proposed GWO algorithm is based on metaheuristics algorithm, it was compared with the ABC [13], PSO [14], and GA [18].

Table 2 The technical characteristics of the host

Host ID	Processing cores	Processing speed, Mips	RAM,MB	Hard, MB	Bandwidth, Mbps
1	4	5000	204,800	1,048,576	102,400
2	2	2500	102,400	1,048,576	102,400
3	1	1000	51,200	1,048,576	102,400

Table 3 The parameters of the algorithm

Parameters	Value
Population size (no. of solutions)	10
Maximum iterations	100
C1, C2	1.49445
R1, R2	Random numbers between 0 and 1

4.2 The obtained results

The results of makespan indicated the significant efficiency of the proposed method in comparison with other methods. Since, in the GWO algorithm first, several wolves attempt to look for some suitable nodes for the purpose of load transmission, it takes less time to complete processing in comparison with other algorithms and is more acceptable. The proposed method has used the hybrid form of GWO algorithm and QoS. GWO algorithm has been inspired by the behavior of Grey wolves in nature, where they start to search for prey in their environment, and load balancing has used this concept. In the current study, first, the alpha wolf searches the loads that are not involved in any task or service and then orders other wolves to attack them. In addition, the ABC algorithm showed better performance because it can find the most suitable loads in searching for loads. Because the honeybee-mating algorithm uses the round-robin (RR) scheduling in the initial stages, its performance depends on the initial decisions of the RR. As can be seen in the picture, the ABC balancing method is more efficient compared to the improved GA algorithm. That is because load balancing in the ABC method is able to detect the overloaded and underloaded VMs, while the GA algorithm is not equipped with such a feature. This has been illustrated more clearly in Fig. 7.

The below diagram has illustrated the response time in the three investigated algorithms. The number of tasks has been set at 20, 30, 40, 50, 60, 70, and 80. The response time is among the parameters that have to be minimized; for this purpose, to obtain the overall time taken to solve aggregate functions, each service's response time is summed. The above diagram indicates the favorable performance of the proposed algorithm with regard to the response time parameter. Since first the worker bee starts the search for nectar in the bee algorithm and informs the other bees of that location through performing a waggle dance, the diagram of response time in

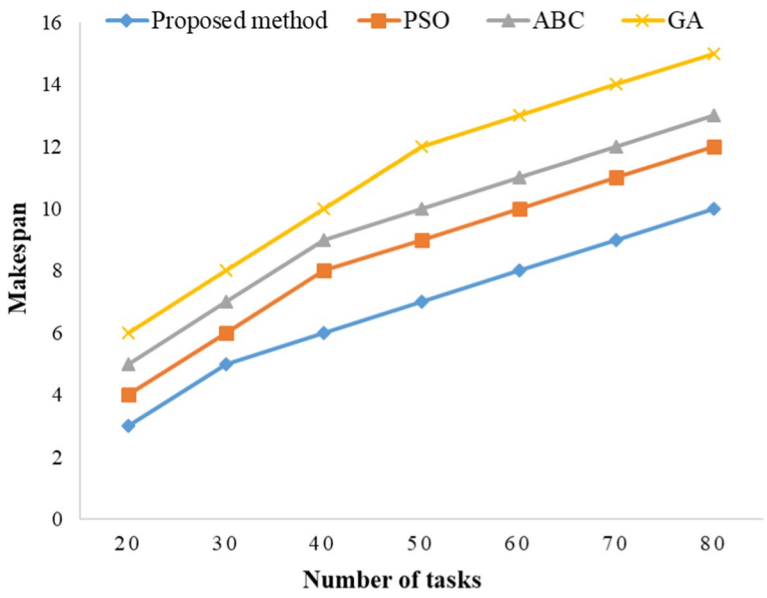

Fig. 7 The comparison of makespan

the artificial bee algorithm is almost similar to that of the proposed method. This has been illustrated more clearly in Fig. 8.

It can be observed in the proposed method that using resources can bring about faster response time and more satisfaction for users. Besides, the method can lead to the equal distribution of workload among VM, which will increase the simultaneous use of resources and decrease the time. It can also be observed the GWO algorithm leads to better results compared to other algorithms and can access resources in a shorter time. The results of this method can clearly be observed in Fig. 9.

Figure 10 shows that though the simulation results of the GWO algorithm are almost the same as the results of the other three algorithms, it has performed optimally concerning cost reduction and has been able to decrease costs to a more extent compared to the different algorithms. The cost evaluation index refers to the money that a user pays for each VM, and it is calculated according to the memory used, processes, VM, and the utilized bandwidth. Since the volume of work is almost the same for all algorithms, their conditions, number, and hardware features are similar, and the running time for all algorithms is the same in this scenario, the algorithm that uses VM to a lesser extent is considered a good one; that is because it can perform its tasks with the minimum of equipment.

The obtained results indicate that the proposed algorithm has less fitness in comparison with the GA, PSO, and ABC algorithms. To test the convergence of the proposed algorithm, it was compared with the GA, PSO, and ABC algorithms with 100 iterations. Figure 11 shows the convergence test for 80 tasks with 200 candidate services for each task in 100 iterations.

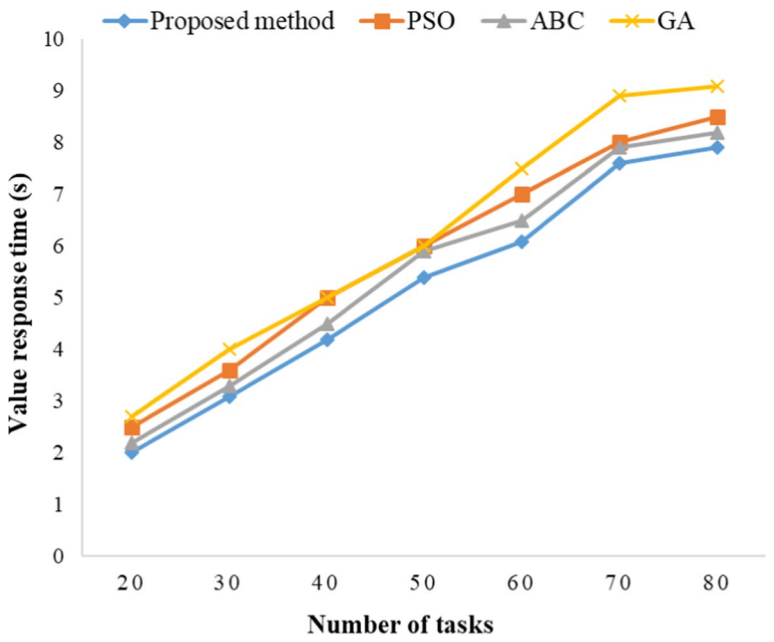

Fig. 8 The comparative diagram of response time

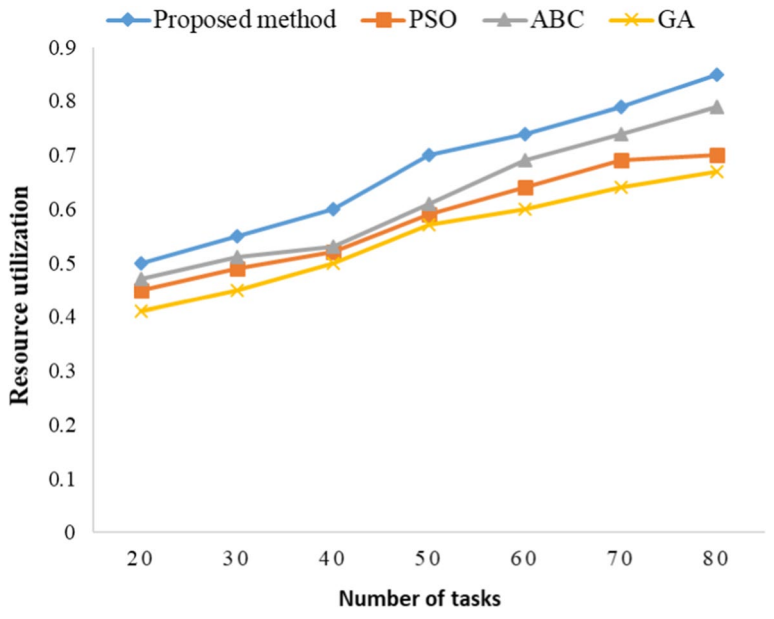

Fig. 9 The diagram of resource utilization

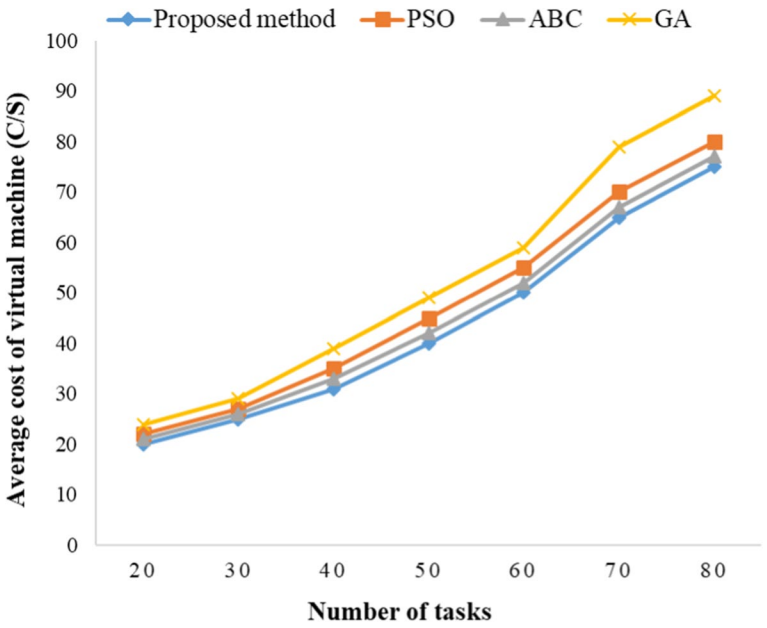

Fig. 10 The average costs of running a VM according to the number of tasks

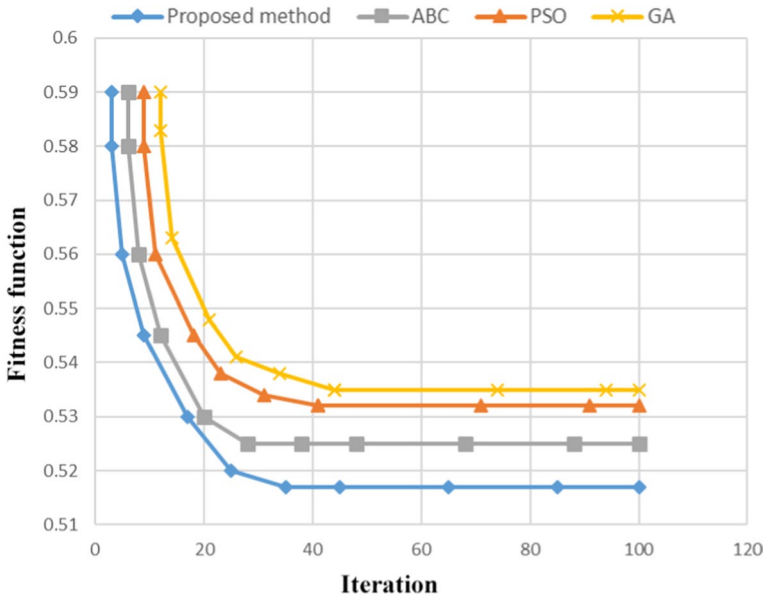

Fig. 11 The convergence diagram for 80 tasks in 100 iterations

5 Analysis of results

Many load-balancing algorithms for distributed, grid, and cloud environments have been proposed. However, each of them has particular advantages and disadvantages. In the current study, a method of load balancing uses the GWO algorithm based on the QoS. The study's goal was to distribute the load between the resources appropriately so that none of the resources remain overload or underload. For this purpose, the QoS and a load of each resource were calculated, and load balancing was maintained using the GWO algorithm. The application was handed over to a resource with high QoS and lower load compared to other resources. Based on the current study's innovation concerning maintaining load balancing based on the QoS, the cloud system's productivity and performance were enhanced. In addition, the simulation results indicated that in the proposed method, the time is taken to complete tasks and the degree of imbalance decreases compared to other algorithms, and the system is more stable. The proposed method was analyzed and compared with the previous methods, and the results showed it to be an ideal algorithm. By conducting a search and calculating each node's weight, the GWO algorithm obtained better results compared to ABC, GA, and PSO algorithms in all parameters. However, the difference between the current study and the other works is selecting an underloaded VM, and a machine has the highest degree of compatibility with the users' requests. Improving the load balancing algorithm in the proposed method led to the more proper distribution of workload in the VMs. Consequently, this resulted in the increased utilization of resources. The heuristic algorithms' proper use and the proposed method reduced the time and costs compared to similar methods, as shown in Fig. 12.

Makespan was another parameter that was investigated, and it was shown that the proposed method is significantly superior to other methods in that regard. In other words, first, the alpha wolves search the machines that have lower load compared to others. After a suitable destination was found, they order the CH to transmit the load. The average results are illustrated in Fig. 13.

One of the other parameters investigated was the response time, and it was shown that the algorithm could lead to more favorable results compared to other methods. Though the ABC algorithm resulted in an immediate and more competitive solution relative to the one proposed in the current study, the results of the GWO algorithm in 80 tasks were better than the other methods. Figure 14 illustrates the average results in this regard.

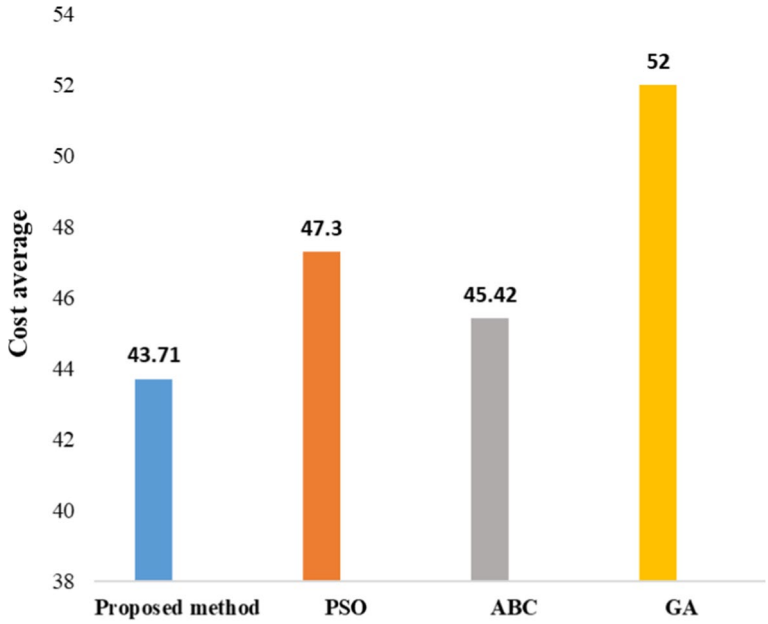

Fig. 12 The average costs for different in 80 tasks

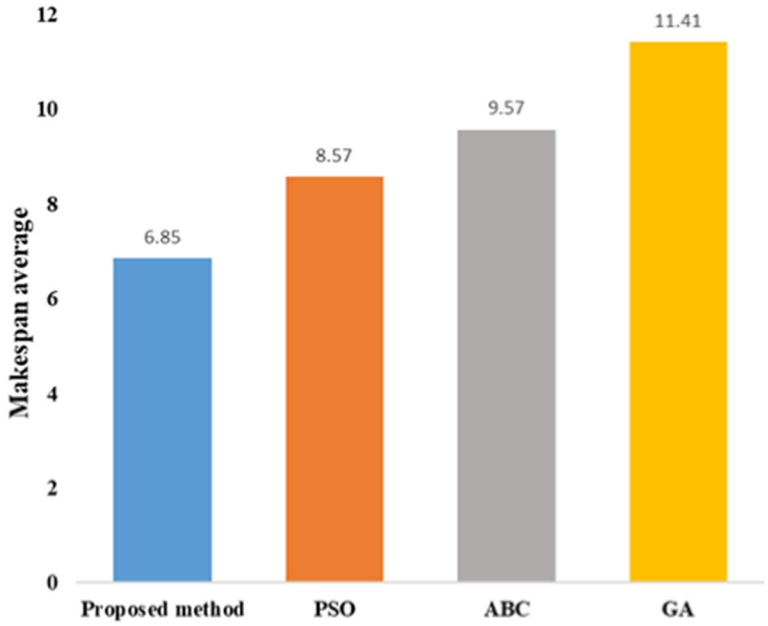

Fig. 13 The average results of makespan in 80 tasks

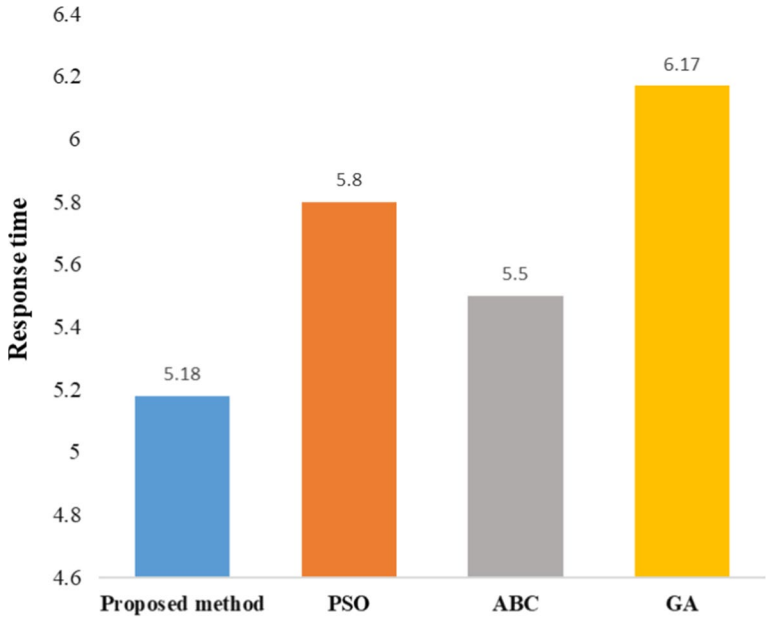

Fig. 14 The average results of response time in 80 tasks

6 Conclusion and future work

Creativity and innovation mean the ability to produce diverse, new, and appropriate solutions to solve problems and issues. In this article, we have tried to find nodes that service values are less than the threshold, where they start to search for prey in their environment, and load balancing has used this concept. First, the alpha wolf explores the services that are not involved in any task or jobs and then order other wolves to attack them. Using this method brought about more favorable results compared to other methods. The proposed algorithm also improves the makespan utilization during load balancing and reduces the response time of the tasks effectively as compared to separated algorithms. In general, if researchers want to extend and improve the results obtained in the current study, they can take the following into consideration. Therefore, presenting a metaheuristic based on machine learning and multi-criteria decision-making (MCDM) method is a suitable direction for the upcoming studies. In addition, more data centers and VMs can be used to improve the proposed method. Furthermore, they can involve other factors such as accessibility, security, and scalability in addition to reliability. In the future, the load balancing will be carried out among the dependent tasks dynamically.

References

1. Haji LM, Ahmad OM, Zeebaree SR, Dino HI, Zebari RR, Shukur HM (2020) Impact of cloud computing and internet of things on the future internet. Technol Rep Kansai Univ 62(5):2179–2190

2. Kumar J, Rani A, Dhurandher SK (2020) Convergence of user and service provider perspectives in mobile cloud computing environment: taxonomy and challenges. Int J Commun Syst 33(18):e4636
3. Goldberg DW, Bowlick FJ, Stine PE (2021) Virtualization in CyberGIS instruction: lessons learned constructing a private cloud to support development and delivery of a WebGIS course. J Geogr High Educ 45(1):128–154
4. Sefati S, Abdi M, Ghaffari A (2021) Cluster-based data transmission scheme in wireless sensor networks using black hole and ant colony algorithms. Int J Commun Syst. https://doi.org/10.1002/dac.4768
5. Eswari S, Manikandan S Competent data transmission function in cloud computing with high probability aesthetic
6. Hayyolalam V, Pourghebleh B, Kazem AAP, Ghaffari A (2019) Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques. Int J Adv Manuf Technol 105(1):471–498
7. Golchi MM, Saraeian S, Heydari M (2019) A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: performance evaluation. Comput Netw 162:106860
8. Alicherry M, Lakshman T (2013) Optimizing data access latencies in cloud systems by intelligent virtual machine placement. In: 2013 Proceedings IEEE INFOCOM, 2013. IEEE, pp 647–655
9. Nurmi D et al. (2009) The eucalyptus open-source cloud-computing system. In: 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009. IEEE, pp 124–131
10. Zanbouri K, Jafari Navimipour N (2020) A cloud service composition method using a trust-based clustering algorithm and honeybee mating optimization algorithm. Int J Commun Syst 33(5):e4259
11. Sefati S, Navimipour NJ (2021) A QoS-aware service composition mechanism in the Internet of things using a hidden Markov model-based optimization algorithm. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2021.3074499
12. Tikhamarine Y, Souag-Gamane D, Ahmed AN, Kisi O, El-Shafie A (2020) Improving artificial intelligence models accuracy for monthly streamflow forecasting using grey Wolf optimization (GWO) algorithm. J Hydrol 582:124435
13. Kruekaew B, Kimpan W (2020) Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing. Int J Comput Intell Syst 13(1):496–510
14. Devaraj AFS, Elhoseny M, Dhanasekaran S, Lydia EL, Shankar K (2020) Hybridization of firefly and Improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments. J Parallel Distrib Comput 142:36–45
15. Lilhore UK, Simaiya S, Maheshwari S, Manhar A, Kumar S Cloud performance evaluation: hybrid load balancing model based on modified particle swarm optimization and improved metaheuristic firefly algorithms
16. Kokilavani T, Amalarethinam DG (2011) Load balanced min-min algorithm for static meta-task scheduling in grid computing. Int J Comput Appl 20(2):43–49
17. Goyal SK, Singh M (2012) Adaptive and dynamic load balancing in grid using ant colony optimization. Int J Eng Technol 4(4):167–174
18. Makasarwala HA, Hazari P (2016) Using genetic algorithm for load balancing in cloud computing. In: 2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 2016. IEEE, pp 1–6
19. Garg SK, Yeo CS, Anandasivam A, Buyya R (2011) Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers. J Parallel Distrib Comput 71(6):732–749
20. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Futur Gener Comput Syst 25(6):599–616
21. Kashyap D, Viradiya J (2014) A survey of various load balancing algorithms in cloud computing. Int J Sci Technol Res 3(11):115–119
22. Smimite O, Afdel K (2020) Containers placement and migration on cloud system. arXiv:2007.08695
23. Hao F, Lakshman T, Mukherjee S, Song H (2009) Enhancing dynamic cloud-based services using network virtualization. In: Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures, 2009, pp 37–44
24. Dillon T, Wu C, Chang E (2010) Cloud computing: issues and challenges. In: 2010 24th IEEE International Conference on Advanced Information Networking and Applications, 2010. IEEE, pp 27–33

25. Bari MF, Zhani MF, Zhang Q, Ahmed R, Boutaba R (2014) CQNCR: optimal VM migration planning in cloud data centers. In: 2014 IFIP Networking Conference, 2014. IEEE, pp 1–9
26. Ashouraei M, Khezr SN, Benlamri R, Navimipour NJ (2018) A new SLA-aware load balancing method in the cloud using an improved parallel task scheduling algorithm. In: 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), 2018. IEEE, pp 71–76
27. Ghobaei-Arani M, Rahmanian AA, Souri A, Rahmani AM (2018) A moth-flame optimization algorithm for web service composition in cloud computing: simulation and verification. Softw Pract Exp 48(10):1865–1892
28. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
29. Betka A, Terki N, Toumi A, Dahmani H (2020) Grey wolf optimizer-based learning automata for solving block matching problem. SIViP 14(2):285–293
30. Mishra SK, Sahoo B, Parida PP (2020) Load balancing in cloud computing: a big picture. J King Saud Univ Comput Inf Sci 32(2):149–158
31. Siddiqi MH, Alruwaili M, Ali A, Haider SF, Ali F, Iqbal M (2020) Dynamic priority-based efficient resource allocation and computing framework for vehicular multimedia cloud computing. IEEE Access 8:81080–81089

Authors and Affiliations

SeyedSalar Sefati[1] · Maryamsadat Mousavinasab[2] · Roya Zareh Farkhady[3]

Maryamsadat Mousavinasab
maryamsadat.mousavinasab@studenti.polito.it

Roya Zareh Farkhady
Roya.farkhady@roshdiyeh.ac.ir

[1] Department of Computer Engineering, Tabriz branch, Islamic Azad University, Tabriz, Iran

[2] Department of Management and Production Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Turin, Italy

[3] Department of Computer Engineering, Institute of Higher Education Roshdiyeh, Tabriz, Iran