International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

Multi-Objective of Load Balancing in Cloud Computing using Cuckoo Search Optimization based Simulation Annealing

Manjula Hulagappa Nebagiri[1], Latha Pillappa Hnumanthappa[2]

Submitted: 11/10/2023 **Revised**: 12/12/2023 **Accepted**: 19/12/2023

Abstract: Load balancing (LB) in Cloud computing (CC) is the most challenging and helpful research for distributing tasks between Virtual Machines (VMs) at Data Centers (DC). In the CC environment, the tasks are allocated between VMs and have various frame lengths, initial times as well and execution times. The LB is one of the most significant problems in CC and solving these problems leads to reducing the response time, energy consumption, and cost. In this study, a hybrid method of Cuckoo Search Optimization (CSO) and Simulation Annealing (SA) algorithm called CSSA is proposed for efficiently balancing the load in VMs. This approach updates the search space of SA by using the CSO approach by considering the multi-objectives of cost, Resource Utilization (RU), response time and Degree of Imbalance (DoI). The experimental outcomes show that the proposed CSSA delivers the performance metrics such as makespan, Degree of Imbalance, Resource utilization and Response time and achieved values of about 150.09, 36.62, 0.45 and 2467 by using the no. of tasks of 2000, which ensures better results compared with the existing methods named BSASSO, QMPSO and MMHHO.

Keywords: *Cloud computing, Data center, Improved cuckoo search optimization, Load balancing, Simulation annealing, Virtual machine*

1. Introduction

The Load Balancing (LB) is significant for enhanced utilization of cloud resources and obtaining a greater VM performance. The resources are allocated by VMs that are hosted on Physical Machines (PM) [1][2]. The LB is a procedure to provide requests among various systems by task scheduling so that the jobs are executed with minimal time as well as monitors the VM's performance. The LB is the effective and impartial task to determine the load achievement of users and to enhance a resource development rate. The major goal of the LB approach is to minimize the makespan while enhancing resource usage [3][4]. The gain of performing these LB approaches in the cloud can obtain enhanced performance, security, durability, cost reduction as well high throughput. An LB approach enhances the performance and also enhances the quality of service (QoS) parameters like makespan, resource utilization, response time and Degree of Imbalance (DoI) [5][6].

Recently, Cloud Computing (CC) has played an important role in the internet-based transmission of knowledge. The CC has inspired the establishment of number of researchers due to the development of communication technology as well as the enhancement of many Internet users [7][8]. LB in CC is challenging and helpful research for providing the tasks between VM at the DC. The CC is an effective technology that contains three major parts Service providers, cloud environments as well and cloud users [9][10]. A focal point of CC is task scheduling for VMs and the LB of VM. To minimize computational cost in cloud computing, the server association and DC virtualization are significant [11][12]. In this regard, the existing researchers utilize different approaches to allocate resources for VM to control the energy consumption in DC. Identifying the best placement of VMs to PMs is one of the important challenges in cloud management systems [13][14][15]. Even, though virtualization plays a significant role in CC, problems still frequently arise such as improper load balancing and scheduling to VMs. To solve this problem, the scheduling and load balancing between the nodes in CC is proposed to optimize the resource allocation using a hybrid method of CSO and SA. The primary contributions of this research are described as follows;

- The hybrid method of Cuckoo Search Optimization (CSO) and Simulation Annealing (SA) algorithm called CSSA is proposed for balancing the load in the VM. This method utilized the CSO process to get exploration space whereas SA is assigned to find an improved response.
- The proposed method significantly achieves the multi-objective constraints such as Resource Utilization (RU), Response Time (RT), Degree of Imbalance (DoI), and cost. The proposed method is evaluated by using the number of tasks and virtual machines.

The rest of the paper is arranged as follows: Section 2 discusses the recent research on load balancing problems. Section 3 provides the proposed work of this paper. The results and discussion are illustrated in Section 4 and the

[1] *Department of Computer Science, Sambhram Institute of Technology, Bengaluru, India*
[2]*Department of Information Science and Engineering, Sambhram Institute of Technology, Bengaluru, India*
** Corresponding Author Email: n.manjula123@gmail.com*

conclusion is provided in Section 5.

2. Literature Survey

Parida [16] implemented a binary variant of the self-adaptive Salp Swarm optimization (BSASSO) approach for LB. The BSASSO approach maintained not only the trade-off between VM as well as mapped the tasks onto the suitable VM. A single-objective fitness function according to the LB issue was examined to estimate task fitness. A self-adaptive method automatically analyzed a population size for the provided issue. The BSASSO minimized the computational cost and makespan as well as enhanced the throughput but, this approach was considered only for the dependent tasks.

Pradhan [17] presented Deep Reinforcement Learning with Parallel Particle Swarm Optimization (DRLPPSO) for solving a problem of LB. The DRL was used to train a neural network to obtain the best reward and PPSO was used to reduce the entire performance time of all the income load. This approach was aimed to maximize the reward function through the reduction of makespan time as well as consumption of energy while achieving maximum accuracy. The DRLPPSO approach efficiently minimized the overall processing time. The suggested approach was easy to fall into local optimum in high-dimensional space.

Kruekaew and Kimpan [18] introduced the multi-objective task scheduling method with Artificial Bee Colony (ABC) with a Q-learning (MOABCQ) approach for an independent task scheduling approach in CC. The Reinforcement Learning (RL) approach was utilized with the MOABCQ algorithm to the faster performance of the approach. This approach was also aimed to optimize the scheduling as well as resource utilization, cost, makespan, and high throughput. The suggested method achieved the minimum makespan due to MOABCQ-LJF could allocate a task to suitable resources. The MOABCQ-LIF is not optimal as well and system performance cannot be optimized in each dataset.

Sefati [19] developed a Grey Wolf Optimization (GWO) according to the capability of resource reliability to preserve the suitable LB. Initially, the GWO tried to identify the idle or busy nodes and then, the identified node was estimated for every threshold node as well as the fitness function. After identifying and evaluating a node, alpha and beta wolves attacked a prey and chose it as a suitable node. The suggested approach improved the usage of makespan while load balancing as well as effectively reduced the response time. However, the suggested approach has limited functionality in reliability as well as service monitoring.

Kaur and Kaur [20] presented a hybrid approach-based resource provisioning and load-balancing architecture for the implementation of workflow to optimize VM utilization. The two hybrid methods were developed for HDD-PLB framework to Predict the Earliest Finish Time (PEFT) Heuristic with the Ant Colony Optimization (ACO) metaheuristic (HPA). The two developed methods for load balancing had been determined which approach was preferable for HDD-PLB. The suggested method minimized computational time, cost, and effective usage of resources. However, the suggested approach was executed by taking dynamic VMs hosted on a single physical machine.

Jena [21] implemented a new method of Modified Particle Swarm Optimization (MPSO) and an Improved Q-learning algorithm called QMPSO for dynamic load balancing between Virtual Machines (VMs). This approach was taken out to modify the MPSO velocity by pbest and gbest based on the best activity developed by improved Q-learning. This approach was developed to enhance a machine's performance through load balancing between VMs, improve the VM throughput as well and control the balance among task priorities through optimizing a task’s waiting time. The QMPSO achieved the efficient LB by energy utilization comparison. However, the server could design only the VM if it had sufficient memory, due to all the requests being approved by the servers and equivalent VMs comparison by the central server.

Haris and Zubair [22] developed a hybrid optimization approach of Manta Ray Modified Multi-objective Harris Hawk Optimization (MMHHO) for dynamic load balancing. A hybrid process revised a search space of HHO through the Manta Ray Forging Optimization (MRFO) approach by establishing cost, utilization of resources as well and response time. The hybrid process in the developed approach efficiently enhanced the system performance by improving the throughput, and LB among VM, balancing the task based on superiority. The suggested method enhanced the operation efficiency as well as efficiently balanced the VM load. However, the HHO and MRFO have a low convergence rate due to the rapid development of user dependence on the cloud.

Mirmohseni [23] developed a hybrid Fuzzy PSO and Genetic Algorithm (FPSO-GA) for the optimal task scheduling in DC management operations with the energy method. The developed approach was completed through a VM management-aware distribution approach for VM control. This approach was also focused on enhancing the quality of the service as well as minimizing the violation number while the method performance. This approach enhanced resource utilization and efficient load balance performance. However, this approach has a low convergence rate due to the rapid development of user dependence on the cloud.

3. Proposed Methodology

In this research, a hybrid method of Cuckoo Search Optimization (CSO) and Simulation Annealing (SA) algorithm called (CSSA) is proposed to solve the LB problem in CC. The LB technique is achieved by combining the CSO into SA. This approach leads to discovering the better-underloaded container which contains the multi-objective function called the cost, response time, DoI as well as RU to minimize the LB algorithm's fitness function. The process of CSSA discovers the load of every VM as well as balances the load through the fitness function. Fig. 1 depicts the block diagram of the CSSA-based Load Balancing.

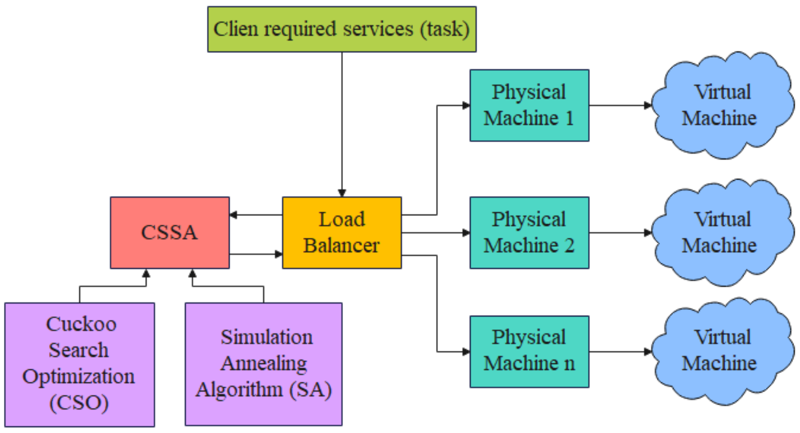

Fig. 1. Block diagram of the proposed method

3.1. Load Balancing in Cloud Computing

The CC provides information on the cloud provisions to cloud consumers where every procedure is implemented in an environment. The cloud contains a large amount of DC called PM and each DC contains the specific computing resources to implement a consumer task [24]. The cloud consumers contain the number of jobs to the execution of the VM and the LB approach is to assign the number of jobs of the user to the VM, which continually establishes VM load in the CC background. A VM load based on the execution time of each task fluctuated as the execution time of one task differed from the alternative task.

The cloud services collect various sets of user requests, demanding the growth of a dynamic environment for task execution. The LB approaches are computed when the balancer obtains user requests and then selects an essential VM. Eventually, the jobs are routed to the load balancer, which utilizes LB to allocate a job to a suitable VM. The VMs obtains the request from the users and the request should be distributed to the VM for processing. The resource allocation CC is challenging when only VMs are overwhelmed or there are some jobs to execute. As a result, the users may be disappointed with their service as well as move to various cloud providers. Therefore, a stable LB approach is generated for an enhanced system performance. The load-balancing process utilized the input as the output of the optimization algorithm. Fig. 2 depicts the general diagram of Load Balancing in Cloud Computing.

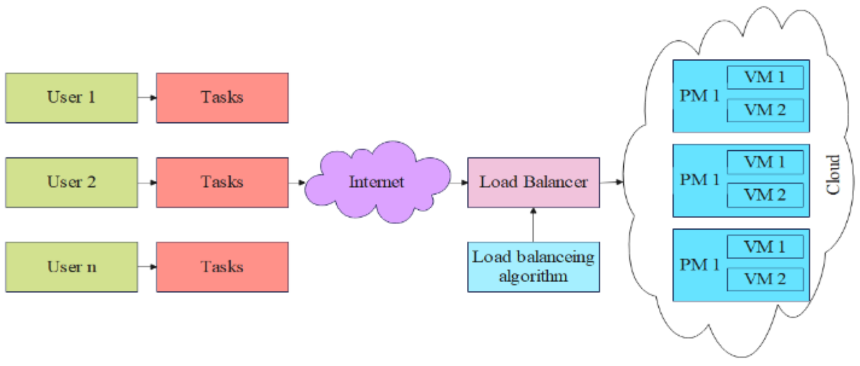

Fig. 2. General diagram of the Load Balancing in Cloud Computing

3.2. Multi-objective Constraints

In this study, the multi-objectives are examined and LB aims to decrease cost, improve RU as well and minimize the RT. By utilizing these objective functions, the system performance can be improved.

3.2.1. Makespan

It is a helpful factor for multi-objective scheduling approaches, which can minimize the execution time as well as permit tasks to be completed in immature. The makespan can be calculated by the (1) as follows;

$$Makespan = Max(ExtTime(VM_i)) \qquad (1)$$

3.2.2. Resource Utilization (RU)

It calculates the number of VM resources. In DC, the utilization of resources is difficult for minimizing the consumption of energy. CC utilizes the number of schemes to generate suitable resource use. Using QoS, the proposed method aims to enhance the utilization of the resource. The RU can be calculated by the (2) as;

$$RU = \frac{\sum_{i=1}^{N}\left(T_{VM_i}\right)}{MS \times N} \quad (2)$$

3.2.3. Response Time (RT)

The LB response time is described as the time to respond to a user request by assigning VMs by minimum load limits. It is inversely correlated to the efficiency of the system. The RT equals the value of the best makespan. The RT can be calculated by the (3) as;

$$RT = Fin_t - Arr_t + TDelay \quad (3)$$

3.2.4. Cost (C)

It varies based on the tasks and is calculated by the following (4) as;

$$Cost = \sum_{i=1}^{N}\left(VM_i^{time} \times VM_i^{cost}\right) \quad (4)$$

3.2.5. Degree of Imbalance (DoI)

It is utilized to compute load imbalance between VMs, which is decreased to make the system balanced. The DoI can be calculated by the (5) as follows;

$$DoI = \frac{T_{max} - T_{min}}{T_{avg}} \quad (5)$$

Where, N – number of tasks or resources; VM^{time} – time of VM has executed a certain task; VM^{cost} – cost of VM for a task execution; T_{VM_i} – time consumed by VM_j to completed all the tasks; TDelay – delay in transmission; Fin_t and Arr_t – end time and arrival time of user demand; T_{max}, T_{min} and T_{avg} – Maximum, minimum and Average execution time.

3.2.6. Fitness Function (F)

The proposed method's fitness function is mathematically evaluated by (6) as follows;

$$Fitness\ Function\ (F) = we^{(RU)} + we^{(RT)} + we^{(C)} \quad (6)$$

Where, w and e – weight factor and exponential function of each parameter; RU, RT and C are the objective functions. Therefore, the proposed CSSA based LB process efficiently balances a load in the cloud according to the previously discussed fitness function.

3.3. Cuckoo Search Optimization

The CSA is a bio-inspired metaheuristic optimization algorithm that is modeled based on mimicking characteristics of cuckoo birds. This bird makes fantastic sounds and has an excellent reproduction strategy. To establish reproduction, the cuckoo laid their eggs in others' nests by draining others' eggs from the nest. The CSA [25] has three types of brood parasitism, which are collaborative breeding, nest acquisition as well as intraspecific brood parasitism. The CSA basic rules are explained as follows. Every cuckoo bird brings one egg while selecting the nest as well and the best nest is used for the next generation. The nest has good eggs that are called the local solution which is used for reproduction. Levy an air tips which is named as flights and is processed for find the result to good globally.

The dimension of a matter to be optimized is decided through the matter nature as N. The number of birds in the nest is m; the current number of iterations is t; the maximum number of iterations is T. A position vector X_i of the nest $(1 \leq i \leq m)$ is defined a $X_i = \{X_{i1}, X_{i2}, \ldots X_{iN}\}$. In CSO, a bird often looks at its path in m nest in N-dimensional space. The optimizing procedure is to efficiently replace the previous worst solution with a new solution as well as depend upon the stochastic walks and Levy flight for seek. As a result, the nesting path and location update of the CS is expressed in (7) as follows;

$$X_i^{t+1} = X_i^t + \alpha \oplus Levy(\lambda) \quad (7)$$

Where, X_i^t and X_i^{t+1} represents the nest of the bird's location vectors i at t and $t-1$ iterations; X_i represents the present status and α represents the step or transition size. ⊕ represents point-point multiplication; Levy(λ) represents the stochastic-looking path. The relationship with time t follows the Levy dispensation is expressed in (8) as follows;

$$Levy(\lambda) \sim \mu = t^{-\lambda}(1 -< \lambda \leq 3) \quad (8)$$

The step size data from the current optimal solution is used and which is computed the step size factor is expressed in (9) as follows;

$$\alpha = \alpha_0(X_i^t - X_{best}) \quad (9)$$

Where, α_0 – stable and held as 0.01; X_{best} – current optimal solution. The stochastic numbers are estimated for the appropriate computation, which is expressed in (10) as follows;

$$Levy(\lambda) \sim \frac{\varphi\mu}{|v|^{\frac{1}{\beta}}} \quad (10)$$

Where, μ and v – standard normal distribution $\beta \epsilon (0,2)$. By integrating the (7) to (10), the CSO utilizes the Lagrange range multiplier function to make a new solution while Levy flight, which is expressed in (11) as follows;

$$X_i^{t+1} = X_i^t + \alpha_0 \frac{\varphi\mu}{|v|^{\frac{1}{\beta}}}(X_i^t - X_{best}) \quad (11)$$

After updating the position, the stochastic number rand ϵ [0,1] is contrasted by identifying the probability p_a. If rand $> p_a$, then X_i^{t+1} is modified, or else it remains unmodified. The desired stochastic walks are utilized to modify the X_i^{t+1} to develop a similar number of new solutions. The arithmetic formula for the stochastic walk desire is expressed in (12) as follows;

$$X_i^{t+1} = X_i^t + r\left(X_j^t - X_k^t\right) \quad (12)$$

Where, r – homogeneous scattered stochastic value in the range between [0,1] interval and X_j^t and X_k^t – two stochastic solutions of the t-th iteration. The output of the CSO is utilized as input for the SA approach to effectively solve the local optimal problem.

3.4. Simulation Annealing (SA)

The Simulated Annealing (SA) is a heuristic stochastic search approach inspired by the annealing phenomenon in nature. It integrates the Metropolis approach as well as distinctly simulates an annealing procedure. In real world, the firmness of molecules and atoms is nearly related to energy level. If a solid is not in a minimum energy state in an annealing process, the solid requires to be provided to heating operation, which is followed by the cooling operation. As the temperature drops steadily, atoms' energy in the solid is slightly minimized, after a systematic crystal is produced, that deviates from the global optimum solution in an approach. Therefore, the speed of cooling can be reduced by energy enhancement. Moreover, the Metropolis approach was developed, which permits the new state to be obtained by definite probability, even if the new solution is poorer than the previous one. According to the Metropolis approach, the acceptance probability P is described with respect to (13) as follows;

$$P\begin{cases}1\\ exp\left(\frac{f(X)-f(X_{new})}{T}\right)\end{cases} \quad (13)$$

Where, X and X_{new} represents current and new solution acquired while next iteration; T represents interrelated temperature which is expressed in (14) as follows;

$$T = \alpha \times T \quad (14)$$

Where, α – number less than 1. To ensure greater search space, a value nearest to 1 is carried and α is assumed as 0.99. With respect to enhancing the SA search ability, this local search approach according to a chaotic map is generated into SA to support easily falling into local optimum. Hence, the solution of the candidate X'_c of SA after the chaotic local search can be acquired by (15) as follows;

$$X'_c = (1 - s) \times T + s \times C'_i \quad (15)$$

Where, T – maximum number of iterations; i = 1, 2, 3, …, n. Moreover, a random walk behavior of levy flight also be useful for an approach to enhance the capability of the search as well as ignore getting captured in a local optimum. Hence, levy flight is generated as an approach to explore the new solutions and which is expressed in (16) as follows;

$$X'_L = X_i \times (1 + Levy(\beta)) \quad (16)$$

Where, X'_L – new candidate solution; Levy(β) – random number developed by the distribution of Levy. The best performance between X'_c and X'_L is obtained as a solution of the candidate by specific probability P.

3.5. Hybrid Cuckoo Search and Simulated Annealing (CSSA)

The SA simulates an annealing procedure of metal which needs the balancing among the heating and cooling procedures of metal to design the metal as applicable shape. The SA is utilized to proximate the global optimum solution of an algorithm by iteratively developing the closest solution over the current solution and after choosing to retain the current solution or exchange it with the closest solution. This meant that SA provided the probability for the poor solution to be enhanced, which is adapted feasibility utilizing a minimizing parameter known as temperature (T). Initially, SA randomly develops a candidate solution (A) and after randomly develops another solution (B). Subsequently, it estimates the fitness value of A and B. If the fitness value of B (f(B)), then SA accepts B. Or else, SA accepts B based on minimizing probability (p(B)), which is expressed in (17) and (18) as follows;

$$\Delta E = f(B) - f(A) \quad (17)$$

$$p(B) = exp^{\Delta E/T} \quad (18)$$

Where, ΔE – difference among fitness values of A and B; T represents the current temperature value as well and exp is the exponential value.

The CS algorithm does not perform efficiently well as endures from an immature convergence issue when it is applied to resolve the optimization problems with the greater dimensions. This is due to its evolutionary operators such as Levy flight, abandoned approach as well and random selection approach, these may not be capable of efficiently exploring the search space of large optimization problems. Moreover, the SA had effective evolutionary operators that balanced among exploration and exploitation of the solutions. The CS aimed to enhance the search performance by applying the SA's evolutionary operators. The SA is utilized in CS to enhance the likelihood of identifying a global optimum solution as well as eliminate being trapped within a sub-optimal value. The optimized solution and client-required tasks are provided to the load balancer and then the load balancer can process the input functions and provide the output to the VMs.

4. Experimental Results

The CSSA-based LB performance has been implemented based on the simulation results. Cloud Sim is a simulation software for testing the proposed method in the cloud environment. It provides efficient modeling as well as simulation over the CC services as well as application testing. The proposed method is simulated on Cloud Sim4.0 software and system specification with Windows 10 OS, Intel core i5 processor, 3.30 GHz CPU and 4GB RAM.

4.1. Performance Analysis

This research estimated the performance of a proposed CSSA based LB process compared with existing methods. Initially, the proposed method performance is determined in terms of performance metrics named makespan, resource utilization, response time, degree of imbalance and cost.

Table 1. Performance Analysis in terms of Makespan using No. of Virtual Machines

Methods	*No. of Virtual Machines*			
	100	**200**	**300**	**400**
PSO	1590	1430	1450	1390
GWO	1550	1400	1330	1300
HHO	1420	1370	1260	1210
CSO	1400	1250	1200	1180
CSSA	1350	1200	1180	1150

Table 1 and Fig. 3 illustrate the performance of the makespan during the no. of VMs from 100 to 400. The proposed CSSA achieved the minimum makespan when compared to the existing methods such as PSO, GWO, HHO and CSO. The proposed method achieves the makespan of 1350, 1200, 1180 and 1150 with the no. of VM of 100, 200, 300 and 400 respectively.

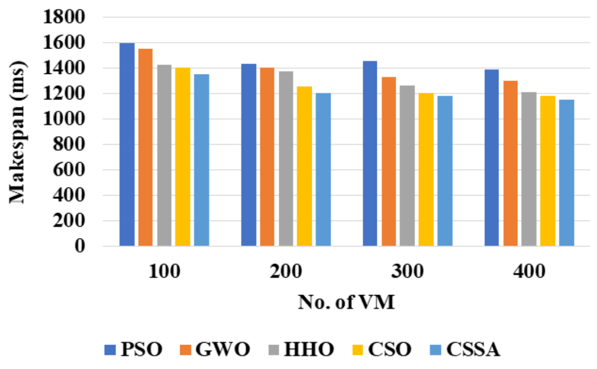

Fig. 3. Graphical representation of Makespan vs. No. of VMs

Table 2 and Fig. 4 illustrate the performance of the DoI during the no. of tasks using 100, 500, 1000 and 2000. The proposed CSSA achieved the minimum DoI when compared to the existing methods such as PSO, GWO, HHO and CSO. The proposed method achieves the DoI of 12.89, 17.89, 25.28, and 36.62 with the no. of tasks of 100, 500, 1000 and 2000 respectively.

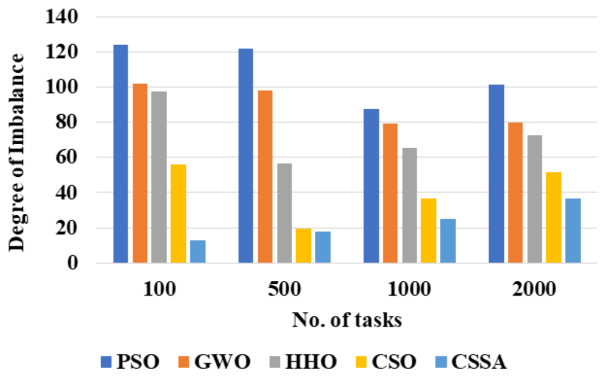

Fig. 4. Graphical representation of Degree of Imbalance vs. No. of tasks

Table 2. Performance Analysis in terms of Degree of Imbalance using No. of Tasks

Methods	*No. of tasks*			
	100	**500**	**1000**	**2000**
PSO	124.26	121.67	87.44	101.27
GWO	101.78	97.89	79.01	79.98
HHO	97.28	56.29	65.20	72.78
CSO	56.23	19.37	36.41	51.37
CSSA	12.89	17.89	25.28	36.62

Table 3. Performance Analysis in terms of Resource utilization using No. of Tasks

Methods	*No. of tasks*			
	100	**500**	**1000**	**2000**
PSO	0.29	0.32	0.33	0.52
GWO	0.25	0.29	0.31	0.51
HHO	0.18	0.21	0.28	0.49
CSO	0.11	0.19	0.25	0.46
CSSA	0.09	0.12	0.23	0.45

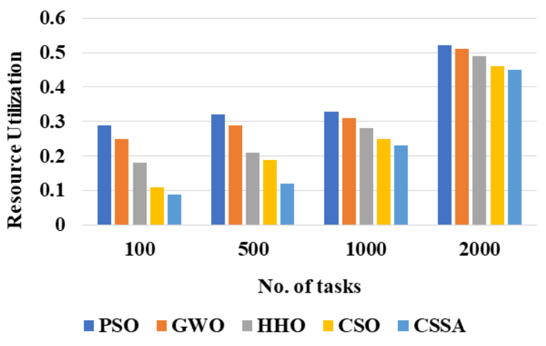

Fig. 5. Graphical representation of Resource utilization vs. No. of tasks

Table 3 and Fig. 5 illustrate the performance of the resource utilization during the no. of tasks using 100, 500, 1000 and 2000. The proposed CSSA achieved the minimum resource utilization when compared to the existing methods such as PSO, GWO, HHO and CSO. The proposed method achieves resource utilization of 0.09, 0.12, 0.23 and 0.45 with the no. of tasks of 100, 500, 1000 and 2000 respectively.

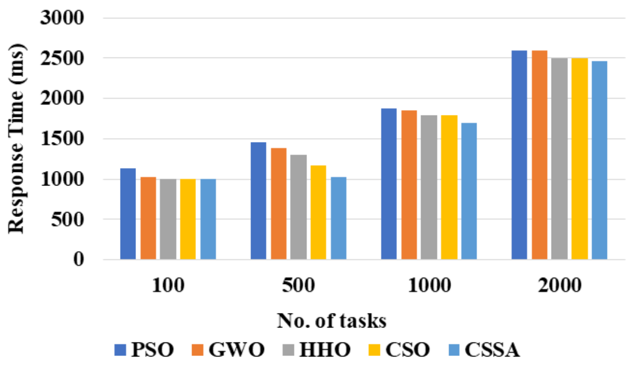

Fig. 6. Graphical representation of Response Time vs. No. of tasks

Table 4. Performance Analysis in terms of Response Time (ms) using No. of Tasks

Methods	*No. of tasks*			
	100	**500**	**1000**	**2000**
PSO	1132	1456	1878	2598
GWO	1029	1389	1856	2590
HHO	1002	1298	1793	2501
CSO	998	1167	1789	2498
CSSA	997	1028	1693	2467

Table 4 and Fig. 6 illustrate the performance of the response time during the no. of tasks using 100, 500, 1000 and 2000. The proposed CSSA achieved the minimum resource utilization when compared to the existing methods such as PSO, GWO, HHO and CSO. The proposed method achieves resource utilization of 997ms, 1028ms, 1693ms and 2467ms with the no. of tasks of 100, 500, 1000 and 2000 respectively.

4.2. Comparative Analysis

This section shows the comparative analysis of the proposed combination of the Cuckoo Search Optimization (CSO) with the Simulation Annealing (SA) Algorithm in terms of number of tasks, makespan, Degree of Imbalance, resource utilization, and response time in Table 5. Table 5. represents the comparative analysis of the proposed method with existing methods using no. of tasks. Table 6. represents the comparative analysis of proposed method with existing methods using no. of VMs.

Table 5. Comparison of proposed method with existing methods using No. of tasks

Method	*No. of tasks*	*Makespan (ms)*	*Degree of Imbalance*	*Resource Utilization*	*Response time*
BSASSO [16]	2000	151.02	153.4	N/A	328971
QMPSO [21]	2000	2389.00	120	N/A	N/A
MMHHO [22]	100	N/A	N/A	0.38	1000
	2000	6600	38.02	N/A	N/A
Proposed CSSA	100	121.28	12.89	0.09	997
	2000	150.09	36.62	0.45	2467

Table 6. Comparison of proposed method with existing methods using No. of VM

Methods	*No. of Virtual Machines*	*Makespan*
BSASSO [16]	200	1500
QMPSO [21]		3500
MMHHO [22]		2700
Proposed CSSA		1200

4.3. Discussion

In this section, the advantages of the proposed method and the limitations of existing methods are discussed. The existing method has some limitations such as the BSASSO [16] and considered only the dependent tasks during balancing the load. The QMPSO [21] server could design only the VN if it had sufficient memory. The MMHHO [22] the HHO and MRFO have a low convergence rate in exploration and exploitation. The proposed CSSA-based load balancing model outperforms these existing model limitations. The CSO has the advantage of efficient random walks and SA is efficient for global search. By combining PSO with GWO achieves a better balance between exploration and exploitation, enhancing convergence speed. The proposed method utilizes up to 200 VMs and number of tasks up to 2000 to evaluate the performance. The experimental results show that a proposed CSSA delivers performance metrics like makespan, Degree of Imbalance, Resource utilization and Response time and achieved values of about 150.09, 36.62, 0.45 and 2467 by using the no. of tasks of 2000 respectively, which ensures the better results compared with the existing methods such as BSASSO, QMPSO and MMHHO.

5. Conclusion

In this study, a hybrid optimization using Cuckoo Search

Optimization (CSO) and Simulation Annealing (SA) algorithm called CSSA is proposed for load balancing in the cloud. The proposed method updates the SA's search space by using the CSO by considering the multi-objective functions of cost, Resource Utilization (RU), response time, and Degree of Imbalance (DoI). The proposed CSSA improves the system performance by balancing the load among VM. The experimental results show that a proposed CSSA delivers performance metrics like makespan, Degree of Imbalance, Resource utilization and Response time and achieves values of about 150.09, 36.62, 0.45, and 2467 by using the no. of tasks of 2000 respectively. In the future, the proposed method will extend to perform various performance metrics such as bandwidth and dependent tasks.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

For this research work all authors' have equally contributed in Conceptualization, methodology, validation, resources, writing—original draft preparation, writing—review and editing.

References

[1] S. Negi, M. M. S. Rauthan, K. S. Vaisla, and N. Panwar, "CMODLB: an efficient load balancing approach in cloud computing environment," *The Journal of Supercomputing*, vol. 77, no. 8, pp. 8787–8839, 2021.

[2] O. Y. Abdulhammed, "Load balancing of IoT tasks in the cloud computing by using sparrow search algorithm," *The Journal of Supercomputing*, vol. 78, no. 3, pp. 3266–3287, 2022.

[3] J. Prassanna and N. Venkataraman, "Adaptive regressive holt–winters workload prediction and firefly optimized lottery scheduling for load balancing in cloud," *Wireless Networks*, vol. 27, no. 8, pp. 5597–5615, 2021.

[4] D. A. Shafiq, N. Z. Jhanjhi, A. Abdullah, and M. A. Alzain, "A load balancing algorithm for the data centres to optimize cloud computing applications," *IEEE Access*, vol. 9, pp. 41731–41744, 2021.

[5] L. Yan, H. Chen, Y. Tu, and X. Zhou, "A task offloading algorithm with cloud edge jointly load balance optimization based on deep reinforcement learning for unmanned surface vehicles," *IEEE Access*, vol. 10, pp. 16566–16576, 2022.

[6] G. A. P. Princess and A. S. Radhamani, "A hybrid meta-heuristic for optimal load balancing in cloud computing," *Journal of Grid Computing*, vol. 19, no. 2, p. 21, 2021.

[7] J. Nazir, M. W. Iqbal, T. Alyas, M. Hamid, M. Saleem, S. Malik, and N. Tabassum, "Load balancing framework for cross-region tasks in cloud computing," *Computers, Materials & Continua*, vol. 70, no. 1, pp. 1479–1490, 2022.

[8] F. M. Talaat, H. A. Ali, M. S. Saraya, and A. I. Saleh, "Effective scheduling algorithm for load balancing in fog environment using CNN and MPSO," *Knowledge and Information Systems*, vol. 64, no. 3, pp. 773–797, 2022.

[9] T. P. Latchoumi and L. Parthiban, "Quasi oppositional dragonfly algorithm for load balancing in cloud computing environment," *Wireless Personal Communications*, vol. 122, no. 3, pp. 2639–2656, 2022.

[10] A. Pradhan and S.K. Bisoy, "A novel load balancing technique for cloud computing platform based on PSO," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 7, pp. 3988–3995, 2022.

[11] S. Nabi and M. Ahmed, "PSO-RDAL: Particle swarm optimization-based resource-and deadline-aware dynamic load balancer for deadline constrained cloud tasks," *The Journal of Supercomputing*, vol. 78, no. 4, pp. 4624–4654, 2022.

[12] S. Nabi, M. Ahmad, M. Ibrahim, and H. Hamam, "AdPSO: adaptive PSO-based task scheduling approach for cloud computing," *Sensors*, vol. 22, no. 3, p. 920, 2022.

[13] 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*, vol. 11, no. 3, pp. 2477–2489, 2021.

[14] M. S. Al Reshan, D. Syed, N. Islam, A. Shaikh, M. Hamdi, M. A. Elmagzoub, G. Muhammad, and K. H. Talpur, "A Fast Converging and Globally Optimized Approach for Load Balancing in Cloud Computing," *IEEE Access*, vol. 11, pp. 11390–11404, 2023.

[15] A. Javadpour, A. M. H. Abadi, S. Rezaei, M. Zomorodian, and A. S. Rostami, "Improving load balancing for data-duplication in big data cloud computing networks," *Cluster Computing*, vol. 25, no. 4, pp. 2613–2631, 2022.

[16] B. R. Parida, A. K. Rath, and H. Mohapatra, "Binary self-adaptive salp swarm optimization-based dynamic load balancing in cloud computing," *International Journal of Information Technology and Web Engineering (IJITWE)*, vol. 17, no. 1, pp. 1–25, 2022.

[17] A. Pradhan, S. K. Bisoy, S. Kautish, M. B. Jasser, and A. W. Mohamed, "Intelligent decision-making of load balancing using deep reinforcement learning and parallel PSO in cloud environment," *IEEE Access*, vol. 10, pp. 76939–76952, 2022.

[18] B. Kruekaew and W. Kimpan, "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee

colony algorithm with reinforcement learning," *IEEE Access*, vol. 10, pp. 17803–17818, 2022.

[19] S. Sefati, M. Mousavinasab, and R. Z. Farkhady, "Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: performance evaluation," *The Journal of Supercomputing*, vol. 78, no. 1, pp. 18–42, 2022.

[20] A. Kaur and B. Kaur, "Load balancing optimization based on hybrid Heuristic-Metaheuristic techniques in cloud environment," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 3, pp. 813–824, 2022.

[21] U. K. Jena, P. K. Das, and M. R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment," *Journal of King Saud University-Computer and Information Sciences*, vol. 34. no. 6B, pp. 2332–2342, 2022.

[22] M. Haris and S. Zubair, "Mantaray modified multi-objective Harris hawk optimization algorithm expedites optimal load balancing in cloud computing," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 10B, pp. 9696–9709, 2022.

[23] S. M. Mirmohseni, C. Tang, and A. Javadpour, "FPSO-GA: a fuzzy metaheuristic load balancing algorithm to reduce energy consumption in cloud networks," *Wireless Personal Communications*, vol. 127, no. 4, pp. 2799–2821, 2022.

[24] S. Dhahbi, M. Berrima, and F. A. M. Al-Yarimi, "Load balancing in cloud computing using worst-fit bin-stretching," *Cluster Computing*, vol. 24, no. 4, pp. 2867–2881, 2021.

[25] C. X. Zhang, K. Q. Zhou, S. Q. Ye, and A. M. Zain, "An improved cuckoo search algorithm utilizing nonlinear inertia weight and differential evolution for function optimization problem," *IEEE Access*, vol. 9, pp. 161352–161373, 2021.