Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2022.026020
Article

Tech Science Press

Hybrid Invasive Weed Improved Grasshopper Optimization Algorithm for Cloud Load Balancing

K. Naveen Durai*, R. Subha and Anandakumar Haldorai

Sri Eshwar College of Engineering, Coimbatore, 641202, India
*Corresponding Author: K. Naveen Durai. Email: naveendurai.k@sece.ac.in
Received: 13 December 2021; Accepted: 14 January 2022

Abstract: In cloud computing, the processes of load balancing and task scheduling are major concerns as they are the primary mechanisms responsible for executing tasks by allocating and utilizing the resources of Virtual Machines (VMs) in a more optimal way. This problem of balancing loads and scheduling tasks in the cloud computing scenario can be categorized as an NP-hard problem. This problem of load balancing needs to be efficiently allocated tasks to VMs and sustain the trade-off among the complete set of VMs. It also needs to maintain equilibrium among VMs with the objective of maximizing throughput with a minimized time span. In this paper, a Hybrid Invasive Weed Improved Grasshopper Optimization Algorithm-based-efficient Load Balancing (HIWIGOA-LB) technique is proposed by adopting the merits of the Invasive Weed Optimization Algorithm (IWOA) into the Grasshopper Optimization Algorithm (GOA) for determining the near-optimal solution that facilitates optimal load balancing. In particular, the random walk strategy is adopted to prevent the local point of optimality problem. It also utilized the strategy of grouping to modify the exploitation coefficient associated with the traditional GOA for balancing the rate of exploration and exploitation. The simulation investigations of the proposed HIWIGOA-LB scheme confirmed its better performance in minimizing the make span and response time by 13.21% and 16.71%, with a maximized throughput of 19.28%, better than the baseline approaches considered for investigation.

Keywords: Cloud computing; load balancing; invasive weed optimization algorithm; grasshopper optimization algorithm; makespan; response time

1 Introduction

In general, cloud computing plays an anchor role as a pervasive and game-changer in most of the operations that involve resource-intensive applications, operating models, collaborative abilities, end-user services, and service provisioning [1]. The primary objective of cloud services is to offer end-users with easy and rapid access to virtual machines [2]. It also focuses on providing a diversified number of distributed services with maximum efficacy [3]. At this juncture, the activity of load balancing the tasks between the VMs is a highly important aspect due to the increasing number of tasks submitted to them in the public cloud [4]. Load balancing also focuses on improving performance with minimized cost and

energy consumption [5]. Moreover, the performance of the cloud environment is comparatively reduced when the number of overloaded VMs on the network is high [6]. The overloaded VMs also result in outages and downtime of tasks, which leads to a subsequent drop in the degree of system utilization in the public cloud [7].

Approximately 60% of the energy inherent in the cloud environment is consumed by the data centers that possess idle servers. Furthermore, 80% of the energy is utilized due to impotent usage of computational resources existing in the data centers [8]. Factors like response time, throughput, migration time, degree of resource utilization, makespan, and overhead are considered essential in the load balancing activity of public clouds [9]. This problem of load balancing is considered the NP problem in real time [10]. The significance of the metaheuritic load balancing algorithm in cloud computing includes: (i) It is multi-objective in nature and it focuses on distributing the tasks or jobs among the hosts or VMs such that the degree of imbalance is relatively minimized; (ii) It focuses on maximizing the degree of resource utilization by minimizing the cost and makespan; (iii) It aids in maintaining the degree of exploitation and exploration involved in the search process employed over the population space (set of VMs or hosts to which tasks need to be allocated); (iv) It plays a dominant role in the formulation of a multi-dimensional fitness function that aids in exploring the possibilities of assigning jobs or tasks to suitable hosts or VMs in the cloud environment, (v) It is in charge of assessing the potential of various factors, such as level of imbalance, makespan, and throughput, which are critical in the process of load balancing in clouds. (vi) It also acts as a primary actor in facilitating the estimation of probability related to the tasks that need to be allocated to the hosts or VMs depending on their degree of availability [11]. Hence, meta-heuristic approaches to load balancing are considered to be phenomenal in sharing the load among the available VMs in the cloud [12].

In this paper, the HIWIGOA-LB scheme is proposed for attaining potential load balancing of tasks between virtual machines with the benefits of Improved GOA (IGOA) and Invasive Weed Optimization Algorithm (IWOA) for balancing exploitation and exploration incurred during the mapping process [13]. This proposed HIWIGOA-LB scheme loaded VMs adopted the strategy of random and grouping to handle the issues of local optimality and improve the exploration by incorporating a modified movement coefficient used by the classical GOA [14]. The simulation experiments of the proposed HIWIGOA-LB and baseline schemes were conducted using makespan, resource utilization rate, throughput, and response time with different numbers of cloudlets. The simulation investigations of the proposed HIWIGOA-LB and baseline schemes are also achieved using execution time, makespan, and degree of imbalance based on their impacts before and after the load balancing process [15]. The subsequent sections of the paper are as follows: Section 2 details the review conducted on the existing swarm intelligence-based load balancing approaches that have contributed to the literature over recent years. Section 3 presents the whole view of the propounded HIWIGOA-LB scheme with the primitive algorithms of GOA and IWOA, with this integration and its employment to the problem of load balancing that maps tasks to suitable VMs depending on the degree of utilization. Section 4 illustrates the simulation experimental setup and results investigation of the HIWIGOA-LB scheme and the benchmarked scheme with the justification behind its predominant performance. Section 5 concludes the paper with its main contributions and future scope of improvement.

2 Related Works

The recently proposed meta-heuristic optimization algorithms-based load balancing schemes are discussed as follows.

Binary Particle Swarm Optimization and Gravitational Search Algorithms (PSOGSA) are used for load balancing and task scheduling in the cloud environment. Binary Load Balancing-Hybrid Grasshopper search

agent Swarm Optimization and Gravitational Search Algorithm (Bin-LB-PSOGSA) is a bio-inspired technique that effectively allows scheduling jobs to enhance the balance level of loads on VMs [16]. The method identifies the best task for VM association, which is persuaded by the measurement of allocated workload and VM execution speed. The searching process of Bin-LB-PSOGSA allows the tasks to submit requests to VMs dynamically. Rescheduling of tasks that have been allotted is re-applied. The search space is described as a hypercube. Every mass travel over hypercube nodes by turning over one or several bits of the mass position matrix. Consecutively, the position matrix is binary-coded. In [17], we have developed a new balanced Grasshopper search agent Swarm Optimization based algorithm (BPSO) for load scheduling in the cloud. The proposed BPSO technique has enhanced balancing in the load scheduling process. The method reduced the total transfer time and total stabilization cost. [18] presented a load balancing scheme using the method of simulated annealing (SA). This SA scheme was initially included to solve the issue of hard-combinatorial optimization issues based on the principle of controlled randomization. SA further plays a vital role in the minimization process that is attributed to temperature replication, which plays a dominant role in the field of thermodynamics. This method was proposed for estimating optimal solutions based on the phenomenon of random derivatives.

The main properties of this method concentrate on determining the worst deviation that has the probability of being accepted as a principal solution. Thus, this SA-scheme was optimal over the other searching methodologies since it inherited the potential to prevent an optimal solution from being trapped in the local point of optimality. A Hybrid Pigeon and Harris Hawks Optimization Algorithm-based load balancing approach (HPHHOA-LBS) was proposed for guaranteeing optimal utilization of resources with minimized task response time [19]. This HPHHOA-LBS approach significantly handled the load among different available VMs in a short time compared to the existing algorithms. It was identified to improve the efficiency of the load balancing process to a maximum level of 97.84%, on par with the baseline schemes. Then, a Binary Bird Swarm Optimization Algorithm-based Load Balancing (BBSWOA-LB) technique was proposed for assigning tasks to suitable VMs in the cloud environment [20]. In this BBSWOA-LB technique, the tasks are considered non-preemptive and independent in nature. The experimental investigations of this approach were conducted using the GOCI dataset logged by Goggle during the execution of cloudlets under real-time workloads. It concentrates on improving the overall performance of the system by balancing the entire system with a minimized response time. It derived the merits of the binary Bird Swarm Optimization Algorithm (BSWOA) for determining the underload and overload conditions of VMs with maximized optimality.

A new Artificial Bee Colony Monarchy Butterfly Optimization Algorithm-based Load Balancing (ABCMBOA-LB) scheme was proposed for effective resource utilization with a minimized makespan and maximized throughput [21]. It was proposed with the exploitation capability of MBOA and the exploration potentialities of ABC for potential mapping of tasks into ideal VMs in the cloud computing environment. The simulation experiments of this ABCMBOA-LB scheme confirmed maximized DOI and minimized makespan independent of the number of cloudlets considered for evaluation. An integrated Discrete ABC and Parteo-based load balancing scheme was proposed by [22] for handling the issue of flexible task scheduling through the enforcement of multiple objectives. This integrated Discrete Artificial Bee Colony (ABC) and Parteo-based load balancing approach consists of two components related to scheduling and routing for facilitating effective solutions in the domain with greater efficacy. It uses discrete values for both the utilized scheduling and routing components. A crossover operator was used in this discrete ABC and Pareto scheme to incorporate more potential into the employee bee phase for the purpose of discovering potent information from the scheduling and routing components. It included an exterior Pareto archive group for recording the previously estimated non-dominated solutions and a rapid Pareto set updating procedure.

A Honeybee behavior-based load balancing algorithm was proposed by [23] for sharing incoming requests in the cloud computing environment in an effective way. It focused on maintaining potential balance among the virtual machines with the objective of improving the level of throughput. It can balance the task's priority on the virtual machines such that the cumulative time involved in waiting is predominantly reduced to the maximum level. It also exhibited a notable improvement in the mean execution time, with a minimized waiting time for tasks in the queue. Then, an enhanced ABC-based load balancing mechanism was propounded by [24] to ensure a high degree of load balancing and scheduling process in the clouds. This enhanced ABC-based load balancing scheme concentrated on reducing the make-span of tasks with a minimized number of VM migrations. It categorized the underloaded tasks as the food sources and the number of tasks isolated from the overloaded VMs as the honeybees in the implemented algorithm [25]. The foraging activities of the honeybees are included in the load balancing process to effectively balance the load among the available virtual machines in the cloud environment.

3 Proposed Hybrid Invasive Weed Improved Grasshopper Optimization Algorithm-based Load Balancing (HIWIGOA-LB) Scheme

The proposed HIWIGOA-LB scheme is propounded as a reliable attempt to facilitate predominant load balancing between virtual machines in clouds based on the benefits of Improved GOA (IGOA) and Invasive Weed Optimization Algorithm (IWOA) for balancing exploitation and exploration. This proposed HIWIGOA-LB scheme concentrates on enhancing the population initialization and exploitation of the search space to enable predominant load balancing between virtual machines in clouds. It included the weighted task scheduling process based on the optimization problem formulated using the parameters of energy makespan, response time, datacenter cost, and degree of imbalance. The proposed HIWIGOA-LB scheme focuses on three vital potentialities that correspond to: (i) the classification of VMs into under-loaded and over-loaded groups during the process of load balancing; (ii) the energy minimization of the datacenter for the objective of minimizing the overall incurred cost; and (iii) the identification of the complete set of VMs in the datacenter that are under-utilized or over-utilized for attaining load balancing in an effective manner. It depicts feasible dimensions that might be used for the formulation of upper and lower thresholds that could form an indicator to identify the over-utilization and under-utilization of VMs by the number of tasks entering into the cloud environment.

3.1 HIWIGOA-Based Load-Balancing Algorithm

In this section, the detailed view of the HIWIGOA algorithm is initially presented, which is then followed by the proposed load balancing algorithm with an eagle insight into the proposed mechanism.

3.1.1 Standard Grasshopper Optimization Algorithm (GOA)

Eq. (1) is used in the standard GOA to determine how to update the position ($SA_{G(i)}$) of the grasshoppers (search agents) to map tasks into suitable VMs.

$$SA_{G(i)} = S_{I(i)} + G_{F(i)} + W_{A(i)} \tag{1}$$

where $S_{I(i)}$ and $G_{F(i)}$ represent the social interaction and gravity force factors that drive the search agent's position change. Moreover, it depicts the wind advection factor that influences the degree of exploitation or exploration that needs to be performed by the search agent. At this juncture, the factor of social interaction is computed based on Eqs. (2) and (3).

$$S_{I(i)} = \sum_{j=1.\, j\neq i}^{S_N} s(d_{ij}) \frac{(x_j - x_i)}{d_{ij}} \tag{2}$$

$$s(r) = ae^{-\frac{r}{b}} - e^{-r} \tag{3}$$

where "a" and "b" represent the adjusting factors that play an anchor role in attaining flexibility of the social parameters. Moreover, the value of d_{ij} representing the euclidean distance with respect to i^{th} and j^{th} search agenst is calculated based on Eq. (4).

$$d_{ij} = |x_j - x_i| \tag{4}$$

Furthermore, the parameters of social interaction and gravity force with respect to the search agent are calculated based on Eqs. (5) and (6).

$$G_{F(i)} = -g_{const}\widehat{v(e_g)} \tag{5}$$

$$W_{A(i)} = u_{const}\widehat{v(e_W)} \tag{6}$$

where, g_{const} and u_{const} represent the gravity and wind power constants, which play an anchor role in impacting the search process of the search agents. Furthermore, $\widehat{v(e_g)}$ and $\widehat{v(e_W)}$ highlight the vector of gravity and wind force, respectively, emphasizing the significance of the search agent movement updating process.

Furthermore, the search agent updating process presented in Eq. (1) can be improved using Eq. (7) by including the parameters of $S_{I(i)}$, $G_{F(i)}$ and $W_{A(i)}$. demonstrated through Eqs. (2), (5), and (6).

$$SA_{G(i)} = \sum_{j=1.\, j\neq i}^{S_N} s(d_{ij}) \frac{(x_j - x_i)}{d_{ij}} - g_{const}\widehat{v(e_g)} + u_{const}\widehat{v(e_W)} \tag{7}$$

where N represents the number of grasshoppers. To apply the GOA to solve the optimization problems, a modified mathematical model can be presented as follows:

$$SA_{G(i)} = \sum_{j=1.\, j\neq i}^{S_N} s(d_{ij}) \frac{(x_j - x_i)}{d_{ij}} - g_{const}\widehat{v(e_g)} + u_{const}\widehat{v(e_W)} \tag{8}$$

where, S_N highlights the number of search agents (grasshoppers). In this context, the mathematical model is modified for use in this type of optimization, such as the mapping of tasks to appropriate VMs specified in Eq. (9).

$$SA_{G(i)} = c\left(\sum_{j=1.\, j\neq i}^{S_N} c\frac{U_T^d - L_T^d}{2} s\left(x_j^d - x_i^d\right)\frac{(x_j - x_i)}{d_{ij}}\right) + \widehat{P_{Best}} \tag{9}$$

where, U_T^d *and* L_T^d depict the upper and lower thresholds of the dimensions "d" considered for exploration, with "$\widehat{P_{Best}}$." being the best position of the search agent during the process of optimization. In addition, the value of is determined based on Eq. (10).

$$c = c_{Max} - Iter_{Curr}\left(\frac{c_{Max} - c_{Min}}{Iter_{Max}}\right) \tag{10}$$

where c_{Max} and c_{Min} represent the minimum and maximum values of the adjusting constant, respectively, and $Iter_{Curr}$ and $Iter_{Max}$ denote the maximum number of iterations and the current implementation iteration.

3.1.2 Inclusion of IWOA for Improving GOA

In the GOA algorithm, the objective function value is completely ignored during the movement of each search agent. The search agent (grasshopper) with a better value of objective function necessitates a bigger step to determine the best solutions that can be feasibly identified in the search space. Hence, the capability of exploitation offered by GOA needs significant improvement. Moreover, GOA suffers from the problem of falling into the local point of optimality. To address these two issues, IWOA is included in the GOA algorithm for attaining a better allocation of resources to VMs and balancing the load systematically. This HIWIGOA-LB scheme is proposed for achieving global optimality during the processing of load balancing and task scheduling. This hybridization of IWOA into GOA was attained using a random walk strategy to improve its exploitation potential. This inclusion of IWOA accelerates the convergence rate of GOA and aids in controlling the step movement of the search agents towards the optimal solution. Moreover, the types and steps used for position updating by the HIWIGOA are achieved through the values of objective functions and iterative numbers. In particular, the random way and IWOA algorithms are hybridized first to improve the local search potential. Secondly, the strategy of grouping is adopted for balancing the tradeoff between exploration and exploitation. In addition, Fig. 1 presents the process of hybridizing IWOA with GOA in the HIWIGOA algorithm.

In this context, the search agent (plant) in the IWOA algorithm has the capability of generating more optimal solutions when they possess better objective functions. The possible number of solutions that could be generated by the IWOA algorithm is presented in Eq. (11).

$$Ps = \frac{Ob_{Fn(i)} - Obc_{Fn(warsti)}}{Ob_{Fn(Best)} - Obc_{Fn(warsti)}} \left(Ps_{(Max)} - Ps_{(Min)}\right) + Ps_{(Min)} \tag{11}$$

where, $P_{S(Max)}$ *and* $P_{S(Min)}$ represent the number of maximum and minimum candidate solutions (seeds) that can be possibly generated by the IWOA algorithm. In specific, $Ob_{Fn(Best)}$ and $Ob_{Fn(Worsti)}$ highlight the best and worst objective function values determined by the algorithm. The generated candidate solutions are determined to be normally distributed based on Eq. (12).

$$\sigma_{Iter} = \left(\frac{(Iter_{Max--Iter_{Curr}})^n}{(Iter_{Max})^n}\right)(\sigma_{Init} - \sigma_{Final}) + \sigma_{Final} \tag{12}$$

where, $\sigma_{Initial}$ and σ_{Final} represents the initial and final standard deviation values associated with the generated candidate solution.

3.1.3 Strategy of Random Walk

In this proposed HIWIGOA algorithm, the strategy of random walk is adopted for enhancing the potentiality of exploitation during the process of load balancing tasks over available VMs. In this strategy, new solutions ($SA_{New\_G(i)}$) are determined in a more random manner from the best primary ($SA_{FBest\_G(i)}$) and secondary best solutions ($SA_{SBest\_G(i)}$) as specified in Eq. (13)

$$SA_{New\_G(i)} = Rand * SA_{FBest\_G(i)} + Rand * SA_{SBest\_G(i)} \tag{13}$$

3.1.4 Strategy of Grouping

In general, different inertial weights are useful for enhancing the potential of optimization algorithms depending on the contexts in which they are employed. This strategy of grouping is utilized for modifying the coefficient c pertaining to the traditional GOA for balancing the rate of exploration and exploitation. This coefficient c is identified to decrease with an increase in the number of iterations in a

more linear manner. It is determined to improve the possibility of improvement. Then, the modified coefficient value is determined based on Eq. (14).

$$c_{mod} = c_{Min}\left(\frac{c_{Max}}{c_{Min}}\right)^{\frac{1}{1+A\,\mathrm{const}\left(\frac{Iter_{Curr}}{Iter_{Max}}\right)}} \tag{14}$$

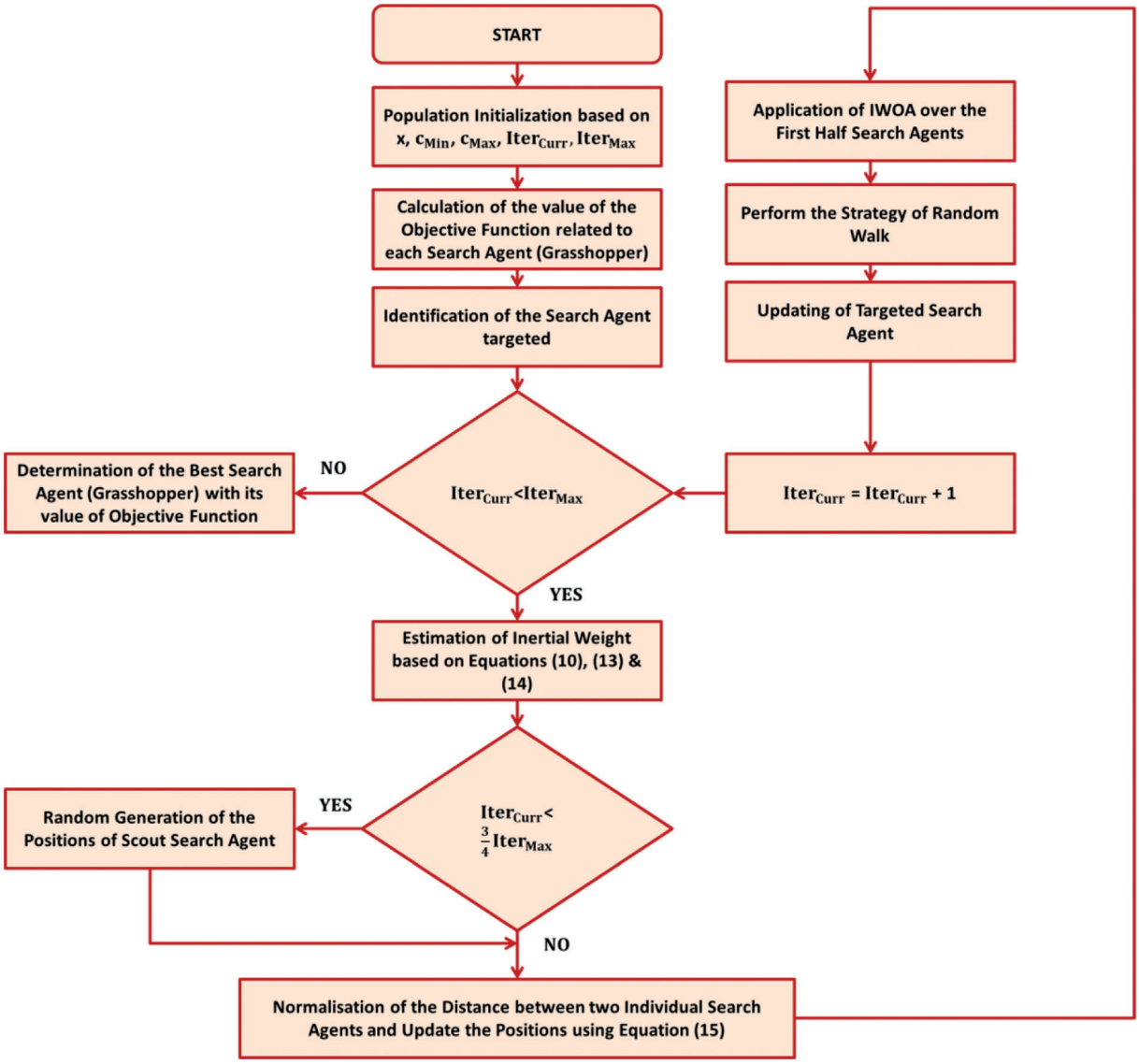

Figure 1: Process of Hybridizing IWOA with GOA in the HIWIGOA Algorithm

The above-mentioned coefficient, c_{mod}, is utilized for search process optimization. In this strategy of grouping, the complete population of search agents is partitioned into three groups, such as elite, onlooker, and scout agents, depending on the objective function values possessed by them in Eq. (15).

$$c_{mod} = c_{Max} - Iter_{Curr}\left(\frac{c_{Max} - c_{Min}}{Iter_{Max}}\right) \tag{15}$$

At this juncture, the elite search agents are determined to inherit better objective function, and thereby exhibit small step movement. The onlooker search agent, on the other hand, uses moderate values of objective function and targets in updating its position, similar to the GOA algorithm. In particular, scout search agents play an important role in preventing the problem of local optimality, which is included in the traditional GOA algorithm. In this strategy, the positions of the scout search agents that possess the worst objective function are generated randomly during the first three quarters of the iterations. During the remaining quarter of iterations, the scout search agent plays an indispensable role in improving the capability of exploration. Thus, the coefficient (c_{mod}) associated with the scout search agent is updated based on Eq. (16).

$$c_{mod} = c_{Max} - (c_{Max} - c_{Min})\left(\frac{Iter_{Curr}}{Iter_{Max}}\right)^4 \tag{16}$$

Finally, this modified (c_{mod}) is included into Eq. (9) to make it ideal for mapping the incoming tasks to suitable VMs as represented in Eq. (17).

$$SA_{G(i)} = c_{mod}\left(\sum\nolimits_{j=1.\,j\neq i}^{S_N} (c_{mod})\frac{U_T^d - L_T^d}{2} s\left(x_j^d - x_i^d\right)\frac{(x_j - x_i)}{d_{ij}}\right) + \widehat{P_{Best}} \tag{17}$$

3.2 System Model and Assumptions Used in the Proposed HIWIGOA-LB Scheme

The system model used for implementing the proposed HIWIGOA-LB approach comprises of 'k' hosts or cloud data centers represented by the set $C =: \{c_1, c_2, \ldots\ldots c_k\}$ with 'm' number of virtual machines depicted through $V_M = \{V_{M(1)}, V_{M(2)}, \ldots\ldots V_{M(r)}\}$ and 'm' number of tasks portrayed through $T_S = \{T_{S(1)}, T_{S(2)}, \ldots\ldots, T_{S(m)}\}$. The tasks are submitted to the cloud computing environment by the users with the help of available cloud brokers. These tasks submitted to the clouds are highlighted through the set of factors, $t_{s(i)} = \{tft_i, tte_i.tl_i, ta_i\}$ that present task finishing time, threshold time necessary for task execution, task length, and time of arrival in the cloud environment. The complete set of factors included in the tasks is transformed onto the virtual machine, $V_{M(i)}$, with the help of the cloud broker available in the environment. The proposed HIWIGOA-LB scheme mainly focuses on the under-utilization and over-utilization of VMs, tasks' makespan, datacenter costs, and energy consumption. In this context, '$PT_{Task(i)}$' highlights the processing time of tasks based on Eq. (18).

$$PT_{Task(m)} = \sum\nolimits_{i=1}^{m} S_{ij} \text{ Where } 1 \leq j \leq n \tag{18}$$

Then, the virtual machine capacity $VM_{CAP(j)}$ that depends on the bandwidth ($BW_{PE(j)}$), million instructions per second ($MIPS_{PE(j)}$) and process count ($PC_{PE(j)}$) associated with the processing elements of clouds is determined based on Eq. (19)

$$CAP_{VM(j)} = BW_{PE(j)} * MIPS_{PE(j)} * PC_{PE(j)} \tag{19}$$

Furthermore, the load that has the possibility of being assigned to each virtual machine $LOAD_{VM}$ is determined based on the total number of tasks $TN_{Tasks}(T, t)$ and the service rate $SR(V_{M(i)}, t)$ of virtual machines at time 't' as represented in Eq. (20)

$$LOAD_{VM} = \frac{Tasks_{TN}(T,t)}{SR\left(V_{M(i)},t\right)} \tag{20}$$

Then, the sum of the task loads assigned to the complete set of virtual machines existing in the cloud environment is presented based on Eq. (21).

$$LOAD_{C(VM)} = \sum_{j=1}^{n} LOAD_{VM(j)} \tag{21}$$

Eq. (22) shows the time incurred for processing the tasks submitted to the total number of VMs present in the cloud environment.

$$TP_{TOTAL\_VM)} = \frac{LOAD_{C(VM(j))}}{CAP_{VM(j)}} \tag{22}$$

Furthermore, the task execution time and processing time of each task assigned to the individual VMs present in the clouds are presented through Eqs. (23) and (24) respectively.

$$Time_{Exec} = \frac{Tl_{(i)}}{CPU_{Fract(i)}} \tag{23}$$

$$TP_{(I-VM)} = \frac{LOAD_{VM(i)}}{CAP_{VM(j)}} \tag{24}$$

Thus, the time of finishing an individual task assigned to a specific VM is calculated based on the execution and start time of the tasks incoming to the clouds based on Eq. (25).

$$TF_{TS(n)} = STime_{TS(n)} + Time_{Exec} \tag{25}$$

At this juncture, the decision variable $DS_{VAR(ij)}$ considered for assigning the incoming tasks to the corresponding VMs is determined using the processing time of tasks as presented in Eq. (26).

$$DS_{VAR(ij)} = \begin{cases} 1, & \text{if } tft_i < tte_i \\ 0, & \text{if } tft_i > tte_i \end{cases} \tag{26}$$

In this context, "makespan" refers to the total time incurred for task completion based on the efficient allocation of VMs. This factor of makespan needs to be potentially minimized. Thus, the objective fitness function concentrates on minimizing the makespan of the tasks into the cloud computing environment as specified in Eq. (27).

$$f_1(Y) = Min\left(\underset{s\in T_S, j\in VM_j}{Max} ft_{ij}\right) \tag{27}$$

where, ft_{ij} is the finishing time incurred by a task $T_{S(s)}$ over a virtual machine $VM_{(j)}$.

Furthermore, if the energy consumption incurred by the execution of the task $T_{S(s)}$ over a virtual machine $VM_{(j)}$ is represented as $EC_{Task(s)}$, then the rate of energy consumptions ($Rate\_EC_{Task(s)}$) incurred by the VM based on the time of execution of the task ($Exec_{Time}$) in the corresponding VM is estimated based on (27).

$$EC_{Task(s)} = Rate\_EC_{Task(s)} * Exec_{Time}$$

Moreover, the cumulative energy consumption of all the VMs for processing the tasks is estimated based on Eq. (28).

$$\mathrm{TEC}\left(\mathrm{VM}_{(j)}\right) = \sum_{i=1}^{k} \sum_{j=1}^{s} \mathrm{EC}_{\mathrm{Task(s\_ij)}} \tag{28}$$

Thus, the objective function concentrating on energy consumptions is presented in Eq. (29).

$$f_2(Y) = \mathrm{Min}(\mathrm{TEC}\left(\mathrm{VM}_{(j)}\right) \tag{29}$$

In addition, the datacenter cost is the other parameter considered for task scheduling of the VMs depending on their availability as determined by Eq. (30).

$$\mathrm{DC}_{\mathrm{Cost}}\left(\mathrm{VM}_{(j)}\right) = \mathrm{VM}_{(j)} * \mathrm{Cost}_{\mathrm{PER-UNTI}} \tag{30}$$

where, $\mathrm{Cost}_{\mathrm{PER-UNIT}}$ is the cost incurred for utilizing 1 KW of energy by the data center under operation in the cloud environment.

Hence, the objective function concentrates on minimizing datacenter cost in Eq. (31).

$$f_3(Y) = \mathrm{Min}(\mathrm{DC}_{\mathrm{Cost}}\left(\mathrm{VM}_{(j)}\right) \tag{31}$$

Finally, the aforementioned objective functions formulated based on makespan, energy consumption, and data center cost are subjected to the constraints presented in Eqs. (32)–(34)

$$\sum_{i=1}^{s} \psi_{ij} = 1, \ \left(t_{S(s)} \in T, \mathrm{VM}_{(j)} \in V\right. \tag{32}$$

$$\sum_{i=1}^{k} T_{S(i0} \leq \mathrm{etd}_i, \ \left(t_{S(s)} \in T, \mathrm{VM}_{(j)} \in V\right. \tag{33}$$

$$\sqrt{\frac{1}{k} \sum_{i=1}^{k} \left(\mathrm{PT}_{\mathrm{Task(s\_i)}} - \mathrm{PT}_{\mathrm{Task(i)}}\right)^2} \leq \mathrm{UP}_{\mathrm{Th}} \tag{34}$$

The aforementioned constraints emphasize that only a single task needs to be allocated to each individual VM, the time of executing the task must be lower than the deadline for completing that particular task by the VM, and the calculated standard deviation of load should be lower than the upper value of the threshold in VM allocation. In addition, the process of load balancing also depends on the degree of imbalance as presented in Eq. (35).

$$\mathrm{Imb\_Degree} = \frac{\mathrm{Max}_{\mathrm{Task(s)}} - \mathrm{Min}_{\mathrm{Task(s)}}}{\mathrm{Mean}_{\mathrm{Task(s)}}} \tag{35}$$

where, $\mathrm{Max}_{\mathrm{Task(s)}}$, $\mathrm{Min}_{\mathrm{Task(s)}}$ and $\mathrm{Mean}_{\mathrm{Task(s)}}$ represents the maximum, minimum and mean number of tasks present in the cloud computing environment.

3.3 Implementation Steps of the Proposed HIWIGOA-LB Scheme

The HIWIGOA-LB scheme uses a multi-objective function for deciding about the allocation or re-allocation of new tasks or old tasks to an appropriate virtual machine or hosts. This allocation and reallocation depend on the primitive constraints that emphasize that the load of the VMs should be greater than the upper limit value after the task has been assigned to them. Then, the constraint of deadline is considered when there is a huge amount of availability in the VMs. Moreover, the migration of tasks from a heavily loaded VM to a lightly loaded VM is imperative depending on the required deadline or completion time of the tasks. In this context, the VM with the minimum value of the greater deadline task is selected when the completion time of the incoming task or re-allocating task is high. In contrast, VMs with higher and moderate deadline tasks are selected when the completion time of the incoming task or re-allocating task is moderate. Further, the grouping of virtual machines is completely

based on the existing load of the virtual machine. In this proposed scheme, two categories of groups are formed and designated as under-loaded and over-loaded VM groups based on the estimation of objective function. The VMs that are present in the overloaded VM groups are made to remove the tasks and wait until they are capable of identifying a potential VM for allocation in the next iteration. The VMs existing in the under-loaded groups are allocated to waiting tasks or tasks that need to be reallocated. This process of removing the tasks from the overloaded VM groups is continued unless the number of underloaded VMs is null. In this context, the solutions (hosts or VMs to be allocated) represent the grasshopper search agents, which are generated based on the principle of randomness. This proposed HIWIGOA-LB scheme utilizes the benefits of the Pareto ranking scheme for handling the issues of multi-objective optimization problems involved in allocating tasks to the VMs based on their availability quantified in terms of under-loaded and over-loaded conditions. It also stores the non-dominating solutions generated previously based on storing the best solutions history determined by the grasshopper search agent. Thus, the aforementioned assisted in the potential allocation of tasks into the VMs based on their over-allocation and under-allocation constraints that play a vital role in the load balancing process.

4 Simulation Results and Discussion

In this section, the experimental setup used for conducting the experimental investigation of the proposed HIWIGOA-LB scheme and the benchmarked BBSWOA-LB (Binary Bird Swarm Optimization Algorithm-based Load Balancing), Adaptive Grasshopper search agent Swarm Optimization Algorithm-based Load Balancing (APSOA-LB), Honey Bee Optimization Algorithm-based Load Balancing (HBOA-LB) and Artificial Bee Colony Monarchy Butterfly Optimization Algorithm-based Load Balancing (ABCMBOA-LB) are presented. Then, the different tests of validation conducted for quantifying the potential of the proposed HIWIGOA-LB scheme over the benchmarked schemes are also demonstrated.

4.1 Experimental Setup

The implementation of the proposed HIWIGOA-LB scheme is achieved using the CloudSim toolkit classes, which are extended for modeling and simulating the environment of the cloud. The CloudSim simulator aided in constructing a virtualized environment that facilitates on-demand provisioning. It is used for modeling, simulating, and experimenting with cloud applications and services. Ten feasible solutions were defined for the algorithm during the experimental process, with the maximum number of iterations assigned to 100 iterations. In addition, the parameters considered during the implementation of the proposed HIWIGOA-LB scheme are depicted in "Tab. 1".

Table 1: Simulation parameters used to implement HIWIGOA-LB scheme

Simulation parameter	Value
Total number of cloudlets	500
Operating system	Windows
Name of the virtual machine (VM)	Xen
RAM size of VM	2048 MB
Number of VMs	10
Simulation tool	CloudSim
CPU detail of VMs	Corei5 Extreme Edition
Number of cores in VM	1
Speed (MIPS) of VMs	1500
Bandwith (MIPS) of VMs	1024

Experiment 1:

Investigation of the Proposed HIWIGOA-LB Scheme with Respect to Makespan, DOI and Execution time before and after Load Balancing under Different Number of Cloudlets.

In this experimental investigation, the proposed HIWIGOA-LB scheme and the benchmarked schemes were evaluated with respect to makespan, degree of balance, DOI, and execution time realized before and after the load balancing process under the number of cloudlets set to 500. Fig. 2 presents the makespan achieved by the proposed HIWIGOA-LB and the baseline BBSWOA-LB, APSOA-LB, HBOA-LB, and ABCMBOA-LB approaches.

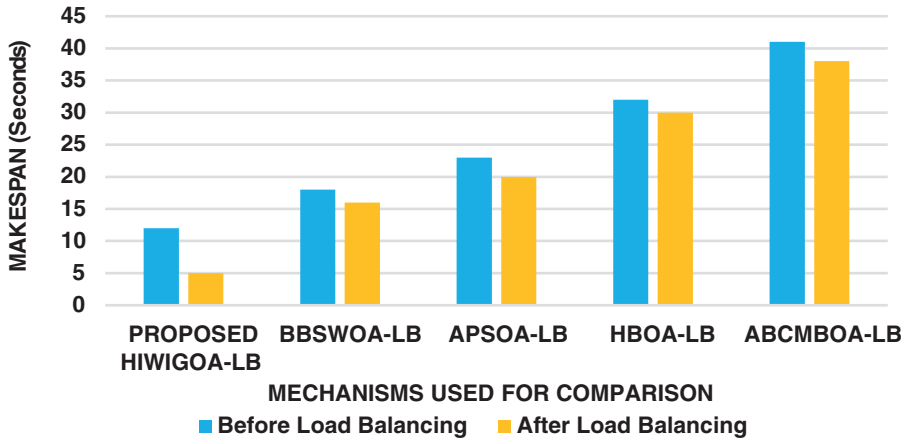

Figure 2: Proposed HIWIGOA-LB-Makespan attained before and after load balancing with number of cloudlets set to 500

The results proved that the makespan guaranteed by the proposed HIWIGOA-LB scheme after load balancing (12 s) is comparatively better than the makespan (5 s) ensured by it before load balancing. The deviation in the time realized during the implementation of the HIWIGOA-LB scheme is 7 s. But, the baseline BBSWOA-LB, APSOA-LB, HBOA-LB, and ABCMBOA-LB approaches facilitated a gap of only 4 s, 3 s, and 3 s, respectively. This potential minimization in the makespan confirmed by the proposed HIWIGOA-LB scheme is mainly due to the utilization of the random strategy of IWOA into GOA that is attributed to a better balance between the exploitation and exploration rate. Thus, the proposed HIWIGOA-LB scheme minimized the makespan before and after load balancing by 19.56%, which is comparatively better than the minimized makespan of 13.21%, 11.98%, 10.52%, and 9.64%, facilitated by the baseline BBSWOA-LB, APSOA-LB, HBOA-LB, and ABCMBOA-LB approaches.

Furthermore, Fig. 3 depicts the degree of imbalance (DOI) achieved by the proposed HIWIGOA-LB and the baseline BBSWOA-LB, APSOA-LB, HBOA-LB, and ABCMBOA-LB approaches. The DOI attained by the proposed HIWIGOA-LB scheme after load balancing is identified to be 15%, which is an improvement over the DOI of 24 ensured by it before load balancing. Hence, the deviation in DOI visualized during the implementation of the HIWIGOA-LB scheme is 9%. However, the benchmarked BBSWOA-LB, APSOA-LB, HBOA-LB, and ABCMBOA-LB approaches only guaranteed a deviation of 4%, 5%, 5%, and 3%, respectively. This predominant minimization in DOI is attained by the proposed HIWIGOA-LB scheme mainly due to the constraints that are enforced during the process of identifying the status of under and over-utilization of VMs, such that the incoming tasks may be optimally allocated to suitable VMs. Thus, the proposed HIWIGOA-LB scheme minimized the DOI before and after load balancing by 22.18%, which is comparatively better than the minimized DOI of 15.41%, 12.38%, 11.94%, and 10.72%, facilitated by the baseline BBSWOA-LB, APSOA-LB, HBOA-LB, and ABCMBOA-LB approaches.

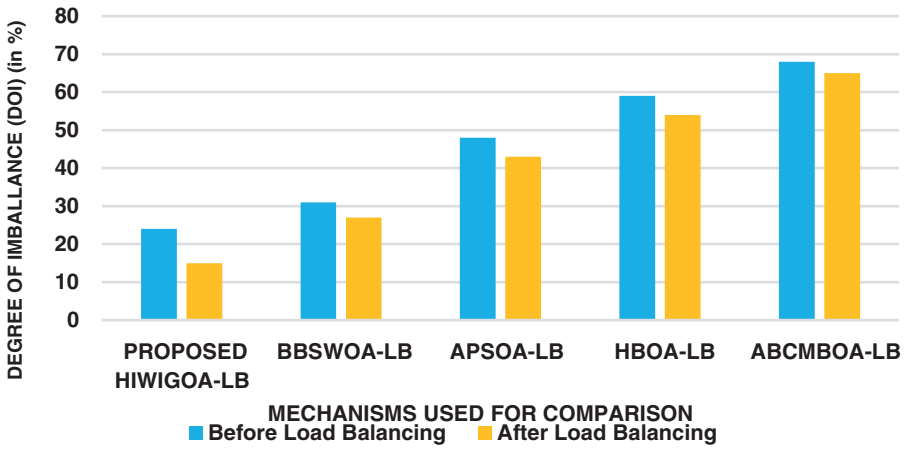

Figure 3: Proposed HIWIGOA-LB-degree of imbalance (DOI) achieved before and after load balancing with number of cloudlets set to 500

Moreover, Fig. 4 demonstrates the execution time incurred by the proposed HIWIGOA-LB scheme and the baseline BBSWOA-LB, APSOA-LB, HBOA-LB, and ABCMBOA-LB approaches with respect to makespan and resource utilization with the number of cloudlets set to 500. The execution time incurred for makespan and resource utilization by the proposed HIWIGOA-LB scheme is highly minimized and on par with the benchmarked load balancing approaches. This minimization in execution time by the proposed HIWIGOA-LB scheme is mainly due to the adoption of a random walk of IWOA into GOA with a sustained balance between exploration and exploration. It also handled the problem of local point of optimality and thereby improved the execution rate compared to the existing schemes used for investigation. With respect to makespan, the execution time incurred by the proposed HIWIGOA-LB scheme is minimized by 14.21%, which is comparatively better than the execution time minimization of 10.86%, 9.42%, 8.23%, and 7.92%, ensured by the baseline BBSWOA-LB, APSOA-LB, HBOA-LB, and ABCMBOA-LB approaches. On the other hand, the proposed HIWIGOA-LB scheme with respect to resource utilization minimized the execution time by 17.32%, which is superior on par with 13.28%, 12.42%, 11.06%, and 10.64%, guaranteed by the benchmarked approaches.

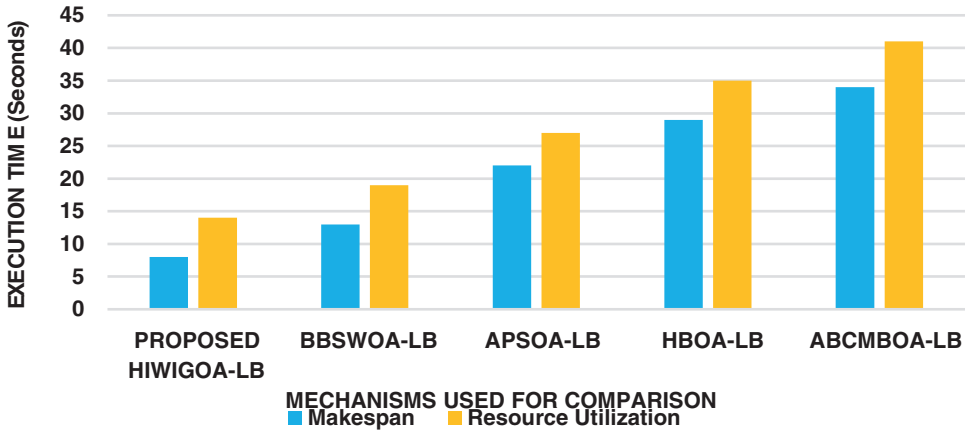

Figure 4: Proposed HIWIGOA-LB-execution time incurred for makespan and resource utilization during load balancing with number of cloudlets set to 500

Experiment 2:

Investigation of the Proposed HIWIGOA-LB Scheme with Respect to Throughput, Response time, Resource Utilization time and DOI under Different Number of Cloudlets.

In this experimental investigation, the proposed HIWIGOA-LB scheme and the benchmarked schemes are evaluated based on mean throughput, response time, resource utilization rate, and degree of imbalance (DOI) with different numbers of cloudlets. Figs. 5 and 6 present the performance of the proposed HIWIGOA-LB scheme and the baseline BBSWOA-LB, APSOA-LB, HBOA-LB, and ABCMBOA-LB approaches evaluated based on the parameters of mean throughput and response time with different numbers of cloudlets. The response time is identified to be remarkably minimized by the proposed HIWIGOA-LB scheme as the parameters considered for optimization are globally optimized during the process of task scheduling. Further, the mean throughput is significantly enhanced by the proposed HIWIGOA-LB scheme as it adopts the grouping strategy to perform a balanced local and global optimization process. Thus, the results proved that the mean throughput achieved by the proposed HIWIGOA-LB scheme with different numbers of cloudlets is improved on an average by 14.56%, 12.39%, 11.85%, and 10.21%, better than the benchmarked approaches. Moreover, the proposed HIWIGOA-LB scheme with respect to response time is minimized by 13.96%, 11.24%, 10.78%, and 9.12%, better than the benchmarked BBSWOA-LB, APSOA-LB, HBOA-LB, and ABCMBOA-LB approaches.

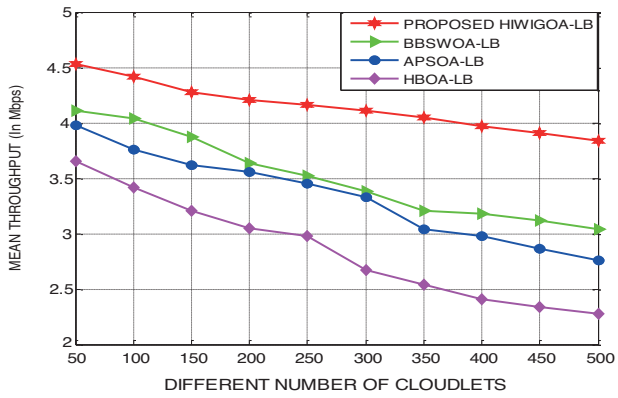

Figure 5: Proposed HIWIGOA-LB-mean throughput under different number of cloudlets

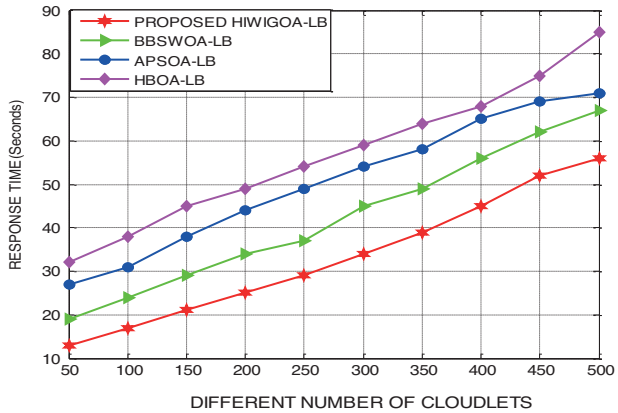

Figure 6: Proposed HIWIGOA-LB-response time under different number of cloudlets

Figs. 7 and 8 demonstrate the performance of the proposed HIWIGOA-LB scheme and the baseline BBSWOA-LB, APSOA-LB, HBOA-LB, and ABCMBOA-LB approaches evaluated based on the parameters of resource utilization rate and DOI with different numbers of cloudlets. This significant performance of the proposed HIWIGOA-LB scheme in terms of resource utilization rate and DOI is realized mainly due to the adoption of a grouping strategy that played an anchor role in determining a better optimal solution in the search space. The results proved that the resource utilization rate guaranteed by the proposed HIWIGOA-LB scheme with different numbers of cloudlets is improved on an average by 17.65%, 15.21%, 13.29%, and 11.84%, better than the benchmarked approaches. Moreover, the proposed HIWIGOA-LB scheme with respect to DOI is reduced by 16.24%, 14.29%, 12.89%, and 10.43%, which is superior to the benchmarked BBSWOA-LB, APSOA-LB, HBOA-LB, and ABCMBOA-LB approaches.

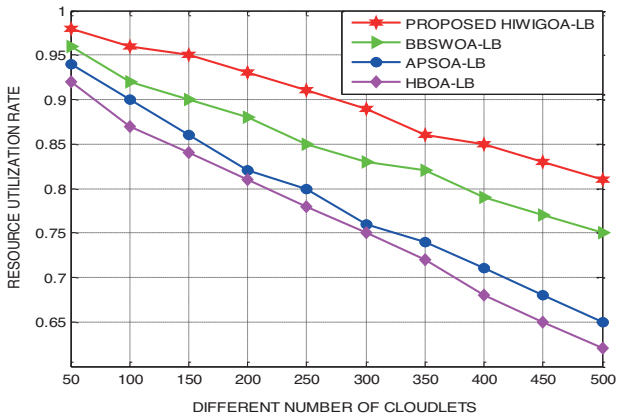

Figure 7: Proposed HIWIGOA-LB-resource utilization rate under different number of cloudlets

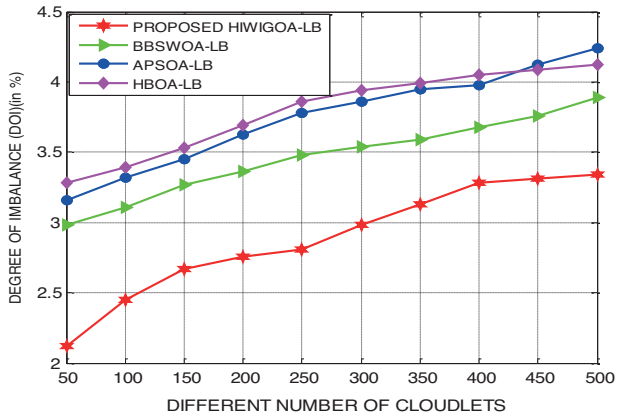

Figure 8: Proposed HIWIGOA-LB-DOI under different number of cloudlets

5 Conclusion

The proposed HIWIGOA-LB scheme achieves potential load balancing and task scheduling by optimally managing the degree of exploration and exploitation in a more balanced manner through the adoption of IWOA and GOA algorithms. It integrated IWOA and GOA using the strategy of random walk and grouping to improve the exploration capability during optimization and improve the exploitation by preventing the problem of local optimality, in which most optimization algorithms get stuck in local solutions. The simulation results of the proposed HIWIGOA-LB scheme confirmed a

reduced execution time of 14.21% with respect to makespan, which is comparatively better than the execution time minimization of 10.86%, 9.42%, 8.23%, and 7.92%, ensured by the baseline BBSWOA-LB, APSOA-LB, HBOA-LB, and ABCMBOA-LB approaches. In addition, the results also confirmed that the resource utilization rate guaranteed by the proposed HIWIGOA-LB scheme with different numbers of cloudlets is improved on an average by 17.65%, 15.21%, 13.29%, and 11.84%, better than the benchmarked approaches. As a part of the future scope, it has been decided to formulate a binary spider monkey optimization algorithm and compare it with the proposed HIWIGOA-LB scheme with different conditions of experimentation.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] D. C. Devi and V. R. Uthariaraj, "Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks," *The Scientific World Journal*, vol. 2016, no. 3, pp. 1–14, 2016.

[2] H.-C. Hsieh and M-L. Chiang, "The incremental load balance cloud algorithm by using dynamic data deployment," *Journal of Grid Computing*, vol. 17, no. 3, pp. 553–575, 2019.

[3] S. Bose, "Balancing load of cloud data center using efficient task scheduling algorithm," *International Journal of Computer Applications*, vol. 159, no. 5, pp. 1–5, 2017.

[4] K. M. Baalamurugan and S. V. Bhanu, "An efficient clustering scheme for cloud computing problems using metaheuristic algorithms," *Cluster Computing*, vol. 22, no. S5, pp. 12917–12927, 2018.

[5] M. Kumar and S. C. Sharma, "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing," *Procedia Computer Science*, vol. 115, pp. 322–329, 2017.

[6] B. Mondal, K. Dasgupta and P. Dutta, "Load balancing in cloud computing using stochastic hill climbing-a soft computing approach," *Procedia Technology*, vol. 4, pp. 783–789, 2012.

[7] P. Xu, G. He, Z. Li and Z. Zhang, "An efficient load balancing algorithm for virtual machine allocation based on ant colony optimization," *International Journal of Distributed Sensor Networks*, vol. 14, no. 12, pp. 155014771879379, 2018.

[8] M. Gamal, R. Rizk, H. Mahdi and B. E. Elnaghi, "Osmotic bio-inspired load balancing algorithm in cloud computing," *IEEE Access*, vol. 7, pp. 42735–42744, 2019.

[9] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal and S. Dam, "A genetic algorithm (GA) based load balancing strategy for cloud computing," *Procedia Technology*, vol. 10, pp. 340–347, 2013.

[10] S. Razzaghzadeh, A. H. Navin, A. M. Rahmani and M. Hosseinzadeh, "Probabilistic modeling to achieve load balancing in expert clouds," *Ad Hoc Networks*, vol. 59, no. 5, pp. 12–23, 2017.

[11] R. Gao and J. Wu, "Dynamic load balancing strategy for cloud computing with ant colony optimization," *Future Internet*, vol. 7, no. 4, pp. 465–483, 2015.

[12] R. Kumar and T. Prashar, "A bio-inspired hybrid algorithm for effective load balancing in cloud computing," *International Journal of Cloud Computing*, vol. 5, no. 3, pp. 218, 2016.

[13] W. LIU and L. GAO, "Task scheduling strategy based on load balance of cluster in heterogeneous cloud environment," *Journal of Computer Applications*, vol. 33, no. 8, pp. 2140–2142, 2013.

[14] K. R. R. Babu and P. Samuel, "Energy aware clustered load balancing in cloud computing environment," *International Journal of Networking and Virtual Organisations*, vol. 19, no. 2/3/4, pp. 305, 2018.

[15] K. Mahesh, "Workflow scheduling using improved moth swarm optimization algorithm in cloud computing," *Multimedia Research*, vol. 3, no. 3, pp. 36–43, 2020.

[16] M. R. Thanka, P. Uma Maheswari and E. B. Edwin, "An improved efficient: Artificial bee colony algorithm for security and QoS aware scheduling in cloud computing environment," *Cluster Computing*, vol. 22, no. S5, pp. 10905–10913, 2017.

[17] R. M. Alguliyev, Y. N. Imamverdiyev and F. J. Abdullayeva, "PSO-based load balancing method in cloud computing," *Automatic Control and Computer Sciences*, vol. 53, no. 1, pp. 45–55, 2019.

[18] J. Yao and J. He, "Load balancing strategy of cloud computing based on adaptive artificial bee colony algorithm," *Journal of Computer Applications*, vol. 32, no. 9, pp. 2448–2450, 2013.

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

[20] A. Ullah, "Artificial bee colony algorithm used for load balancing in cloud computing: Review," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 2, pp. 156, 2019.

[21] K. Mishra and S. K. Majhi, "A binary bird swarm optimization based load balancing algorithm for cloud computing environment," *Open Computer Science*, vol. 11, no. 1, pp. 146–160, 2021.

[22] L. D. Dhinesh Babu and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing*, vol. 13, no. 5, pp. 2292–2303, 2013.

[23] H. Anandakumar and K. Umamaheswari, "A bio-inspired swarm intelligence technique for social aware cognitive radio handovers," *Computers & Electrical Engineering*, vol. 71, no. 12, pp. 925–937, 2018.

[24] B. Kruekaew and W. Kimpan, "Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing," *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, pp. 496, 2020.

[25] K. N. Durai, R. Subha and A. Haldorai, "A novel method to detect and prevent SQLIA using ontology to cloud web security," *Wireless Personal Communications*, vol. 117, no. 4, pp. 2995–3014, 2020.