[Mansoura Engineering Journal](https://mej.researchcommons.org/home)

Volume 48 | Issue 3 Article 11

2023

Enhanced Load Balancing Based on Hybrid Artificial Bee Colony with Enhanced β-Hill climbing in Cloud

Maha Zeedan
Computer Science and Engineering Department, Faculty of Electronic Engineering, Menoufia University, Menouf, Egypt (e-mail: m_zeedan@yahoo.com),(tel:00201006768689)., m_zeedan@yahoo.com

Gamal Attiya
Computer Science and Engineering Department, Faculty of Electronic Engineering, Menoufia University, Menouf, Egypt (e-mail: gamal.attiya@yahoo.com),(tel:00201002907031)., gamal.attiya@yahoo.com

Nawal El-Fishawy
Nawal El-Fishawy, Computer Science and Engineering Department, Faculty of Electronic Engineering, Menoufia University, Menouf, Egypt (email: nelfishawy@hotmail.com),(tel:00201003095671)., nelfishawy@hotmail.com

Recommended Citation

Zeedan, Maha; Attiya, Gamal; and El-Fishawy, Nawal (2023) "Enhanced Load Balancing Based on Hybrid Artificial Bee Colony with Enhanced β-Hill climbing in Cloud," *Mansoura Engineering Journal*: Vol. 48 : Iss. 3 , Article 11.
Available at: https://doi.org/10.58491/2735-4202.3098

ORIGINAL STUDY

Enhanced Load Balancing Based on Hybrid Artificial Bee Colony with Enhanced β-Hill Climbing in Cloud

Maha Zeedan*, Gamal Attiya, Nawal El-Fishawy

Computer Science and Engineering Department, Faculty of Electronic Engineering, Menoufia University, Menouf, Egypt

Abstract

This paper proposes enhanced load balancer based artificial bee colony and β-Hill climbing for improving the performance metrics such as response time, processing cost, and utilization to avoid overloaded or under loaded situations of virtual machines. In this study, the suggested load balancer is called enhanced load balancing based on hybrid artificial bee colony with enhanced β-Hill climbing (ELBABCEβHC) to improve the response time, processing cost and the resource utilization. Our proposed approach starts by ranking the task then the greedy randomized adaptive search procedure (GRASP) is used in initializing populations. Further, the binary artificial bee colony (BABC) enhanced with the modified β-Hill climbing with the sinusoidal map strategy is applied to schedule tasks considering load balancing in cloud. The proposed approach is implemented in CloudAnalyst. The experimental results show that for different user groups all over the world. The performance of ELBABCEβHC algorithm outperforms round robin (RR), throttled load balancer (TLB), and active monitoring load balancing (AMLB) algorithms considering response time, processing cost and utilization.

Keywords: Algorithms, Artificial bee colony, Cloud computing, Load balancing, Scheduling

1. Introduction

Cloud is a distributed computing system consisting of a collection of interconnected data centers consist of high-performance hosts configured to numbers of virtual machines (VMS) on the same physical machine depending on virtualization. which deliver dynamically on-demand resources with different characteristics over the Internet based on quality of services (QoS) metrics and the service-level agreement (SLA) established between the service provider and the end-users. The cloud provider can offer three types of services as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), or Software as a Service (SaaS). There are many companies to transfer these services from cloud such as Amazon, Yahoo, Microsoft Google etc. via Internet connection using the resources hardware and Software. The users accept services from the cloud without paying attention to the details by sending user requests and receiving responses through the Internet (Ranjan and Buyya, 2009).

There are various challenges and issues when providing services to the end-users as the load balancing problem. Load balancing mechanism is used for redistributing the workload among the nodes trying to find an efficient mapping for a set of tasks to a set of computing machines so that no single node is overloaded or underloaded (Mell and Grance, 2011; Buyya et al., 2011).

There are mainly two types of load balancing algorithms considering the current state of the system, they are static or dynamic. Round robin (RR), and threshold algorithm (TH) are static load balancing that require prior knowledge about the applications and the resources. Dynamic load balancing algorithms as throttled load balancing algorithm (TA) and active monitoring load

Received 14 May 2022; revised 24 December 2022; accepted 25 December 2022.
Available online 16 October 2023

* Corresponding author at: Computer Science and Engineering Department, Faculty of Electronic Engineering, Menoufia University, Menouf, Egypt.
E-mail addresses: m_zeedan@yahoo.com (M. Zeedan), gamal.attiya@yahoo.com (G. Attiya), nelfishawy@hotmail.com (N. El-Fishawy).

https://doi.org/10.58491/2735-4202.3098

balancing algorithm (AMLB) depend on the current state of the system not the previous state (Mishra and Majhi, 2020).

Practically, for solving load balancing, conventional methods may take a long time to find an optimal solution. Researchers are satisfied with the metaheuristics for solving such problems which do not guarantee the optimality but provide acceptable solutions in a reasonable amount of time (Talbi, 2009).

Many classifications criteria may be used for metaheuristics. The most common one are population-based vs single-based.

Single-based metaheuristics include local search as hill climbing, and a greedy randomized adaptive search procedure (GRASP). Population-based metaheuristics include swarm intelligence (SI) optimization techniques which are nature inspired metaheuristics such as artificial bee colony, ant colony, and particle swarm algorithms. These algorithms consist of simple agents trying to solve optimization problems such as load balancing by interacting with each other (Luke and Edition, 2009).

Population-based metaheuristics have the main advantage that they are able to widely scan many search space regions at the same time beginning with a number of random solutions but this may cause premature convergence. To maintain the balance between diversification and intensification over the solution space, population-based algorithms is hybridized with other techniques to overcome such problems (Houssein et al., 2021).

For solving load balancing problems, this paper tackles load balancing problem in cloud using a new hybrid load balancing algorithm called ELBABCEβHC. It combines artificial bee colony with an extension of hill climbing called β-Hill climbing algorithm (Al-Betar, 2017) which in turn enhanced by a chaotic sinusoidal sequence strategy (Peitgen et al., 2006) for generating random values.

We summarize in Tables 1 and 2, respectively, the various notations used in defining load balancing problem and the notations used by the proposed ELBABCEβHC algorithm. Our main contributions in this research are:

Table 1. The notations used in defining the load balancing problem.

Notation	Description
F	The multiobjective function.
Z	The objective functions number.
$f_1(y), f_z(y)$	The objective functions that are counterdary to each other.
S	The decision space $S = \{y_1, y_2, y_3, \ldots\ldots y_d\}$.
PF^*	The pareto front.
ψ_i	The weighted coefficients set.
m	The virtual machines number.
n	The tasks number.
VMS	A list of m virtual machines $\mathrm{VMS} = \{\mathrm{vm}_1, \mathrm{vm}_2, .\mathrm{vm}_\mathrm{m}\}$.
$\mathrm{vm_i}$	A virtual machine i has processing speed (Mp) measured in Million Instructions per Second (MIPS) per processing element.
T	A set of n tasks $\mathrm{T} = \{\mathrm{t}_1, \mathrm{t}_2, \ldots\ldots \mathrm{t}_\mathrm{n}\}$
$\mathrm{t_j}$	The task j with the length (M) in million instructions (MI)
P_e	The processors numbers for running a task $\mathrm{T_j}$ on the virtual machinevm$_\mathrm{i}$
τ_{NL}	The network latency is the time taken by the system to respond to the user request.
d_p	The processing delay which is the time taken by the router to process the data packets.
d_q	The queuing delay or the waiting time which is the time data waits in the router buffer.
d_{pr}	The propagation delay which is the time taken by data to cross the transmission medium that depends on the data speed and the geographical distance.
τ_{TT}	The data transmission delay (time) which is the time taken by data to be transported from one machine to another.
τ_{exij}	The execution time for task $\mathrm{t_j}$ on $\mathrm{vm_i}$.
Ω_{kj}	The bandwidth measured in Bits per Seconds (B/S).
$\mathrm{D}_{kj}^{\mathrm{out}}$	The transmitted data amount in Bits(B).
τ_{arrij}	The arrival time of $\mathrm{t_j}$ at $\mathrm{vm_i}$.
τ_{RT}	The service response time τ_{RT} for a single user request (task) $\mathrm{t_j}$ assigned to the virtual machine $\mathrm{vm_i}$
Ϛʈ$_i$	The completion time at the virtual machine $\mathrm{vm_i}$.
x_{ij}	The decision variable $\mathrm{x_{ij}} \in \{0, 1\}$.
ɱȿ	The makespan (a schedule length).
Ϛʈ$_{Av}$	The average completion time.
Aû	The resource utilization.
C_i^{exe}	The cost of data processing per hour at the virtual machine $\mathrm{vm_i}$
$C_d{}^{exe}$	The total processing cost for all virtual machines.
a_{ik}	The resources of vm_i used by t_k
pp_i	The processing power of the virtual machine vm_i.
BA	The Budget for processing tasks in $.

Table 2. The notations used in formulating the proposed ELBABCEβHC.

Notation	Definition
Pu_E	The population of the bees.
P_k	The probability assigned to the K^{th} food source.
β-operator	The operator $\beta \in [0,1]$ utilized in β-hill climbing.
N-operator	To navigate the neighboring solutions of the current one.
R	$\mathscr{R} = \{r_1, r_2, \ldots r_i \ldots r_N\}$ utilized as a random value.
x_i	Current solution in β-hill climbing.
$\acute{x}_i$	New solution in β-hill climbing.
$U(0,1)$	Uniform distribution of the solution space.
u_{bi}	The solution space upper bound.
l_{bi}	The solution space lower bound.
MaxItr	Maximum iteration of the β-hill climbing.
Itr	Number of iterations of the β-hill climbing.

(1) This paper proposes an efficient load balancing algorithm in cloud called enhanced load balancing based on hybrid artificial bee colony with enhanced β-Hill climbing (ELBABCEβHC) to improve response time, processing cost and utilization.
(2) The proposed ELBABCEβHC approach is implemented in cloudAnalyst simulator (Wickremasinghe et al., 2010).
(3) The algorithm performance is evaluated for a large scale application as a Facebook considering peak hours and workloads and compared with the state of art algorithms.
(4) The experiments results confirmed that the proposed ELBABCEβHC minimizes **the service response time** $\boldsymbol{\tau_{RT}}$ and **the processing cost** $(C_d{}^{exe})$ while maximizing **the resource utilization (**$A\hat{u}$**)** compared to round robin (RR), throttled load balancer (TLB), and active monitoring load Balancing (AMLB).

The remainder of this paper is organized as: Section *II* is related work. Section *III* defines and formulate the load balancing problem as an optimization problem. Section *IV* proposes in detail the new hybrid approach. Section V presents the experimental results and analysis. Finally, Section VI presents the conclusion and Section VI presents the future work.

2. Related work

For solving load balancing problems, some approaches are investigating. We summarize the comparison of various existing load balancing algorithms in Table 3.

Hill climbing (Yuret and De La Maza, 1993; Arram et al., 2014) is a metaheuristic local searching technique. The authors in (Al-Betar, 2017) propose an extension version called β-hill climbing(βHC) that utilizes two stochastic operators called N-operator and β-operator in hill climbing to control the balance between the exploration and exploitation during the search. The authors in (Al-Betar et al., 2019) propose an adaptive β-hill climbing (AβHC) algorithm for adapting N and β operators according to the parameter K. We try to tackle load balancing problem in cloud using the proposed ELBABCEβHC algorithm that combines artificial bee colony (ABC) (Karaboga and Basturk, 2007; Pham and Castellani, 2015; Kim et al., 2017) with β-Hill climbing (Al-Betar, 2017; Zahid et al., 2018) and the chaotic sinusoidal strategy (Peitgen et al., 2006).

3. Modeling load balancing problem

This section describes our proposed architecture of the load balancer, the load balancing model, and the proposed load balancing algorithm with the constraints.

3.1. The system model

The general steps of the resource provisioning scenario by any cloud provider are the following (Mell and Grance, 2011; Buyya et al., 2011): (i)The application submits its tasks to the scheduling service. (ii)The scheduling algorithm finds each resource that matches the task requirements. (iii) The scheduling service will ask provision service to get resources as the algorithm determined. (iv)The provision service will send the provision request to a resource pool manager which communicates with seach other to start a number of virtual machines based on the requests from the provision service. (v) A worker instance (virtual machine) will be configured and running then connect to the master machine and will register themselves. (vi)The scheduling algorithm will be acknowledged once these virtual machines registered and will start allocating tasks to the virtual machines. (vii)Once the application tasks are finished, all the resources will be released.

Due to the task and the resources heterogeneity, it is hard to maintain the performance of the task scheduling and distribute equal workload among the servers ensuring high quality of services (QoS) and the service-level agreement (SLA). Therefore, the load balancing is the main issue in the cloud.

The framework architecture of our load balancer, enhanced load balancing based on hybrid artificial bee colony with enhanced β-Hill climbing (ELBABCEβHC) is implemented in Fig. 1.

In our proposed framework, when the user generates task request with the specified application Identification, the service broker receives the

Table 3. Comparison of previous existing load balancing algorithms.

Ref. in Year	Authors	Algorithm	The technique	Advantages	Disadvantages
(Phi et al., 2018)in 2018	Nguyen Xuan Phi, Vasudha, and Tyagi, S. S.	TMA: Throttled modified algorithm based on throttled load Balancer.	TMA updates and maintains two index tables: the available index table with the virtual machine available status and the busy index table with virtual machines not available status.	Improving virtual machine response time on cloud.	1)-The response time improves slightly. 2)-Does not consider Processing Time.
(Singh and Prakash, 2018) in 2018	Aditya Narayan, Singh, and Shiva Prakash	WAMLB: Weighted Active Monitoring Load Balancing in Cloud.	WAMLB strategy calculates the weight factor for each virtual machine based on the physical memory, processors number, the bandwidth, and the processor speed. The virtual machines available with the highest weight are selected for task execution.	Improving virtual machine response time.	Does not consider: 1)-CPU utilization. 2)-Makespan. 3)-Cost.
(Francis Saviour et al., 2020) in 2020	A. Francis Saviour Devaraj, Mohamed Elhoseny, S. Dhana Sekaran, E. Laxmi Lydia, and K. Shankar	FIMPSO: A hybrid of firefly and Improved Multi-Objective Particle Swarm Optimization (IMPSO) technique.	Firefly (FF) algorithm tries to minimize the search space where IMPSO technique is implemented to enhance the response by selecting the global best (Gbest) particle with a small distance of point to a line then global best (Gbest) particle candidates is elected.	Enhance response time, CPU, and memory utilization, reliability and throughput along with a make span	Does not consider other QoS performance metrics as the cost, and the profit.
(Balaji et al., 2021) In 2021	Balaji, K., P. Sai Kiran, and M. Sunil Kumar.	ACSO: Adaptive cat swarm optimization.	In ACSO, each cat duplicates its own position to seek memory pool. All the cat positions are compared using a fitness value which is a combination of energy, cost and memory utilization. Cats start moving from seeking mode to tracing mode, after seeing any pray. In tracing mode, cats conduct will be recreated with ACSO algorithm, and each cat follows the best position to update its velocity.	Enhance energy, cost, and memory utilization	Does not consider the response time or the processing Time.

(Mishra and Majhi, 2021) In 2021	Mishra, Kaushik, and Santosh Kumar Majhi.	BSO-LB: A bird swarm optimization load balancing algorithm.	BSO-LB considers tasks as birds and VMs as food patches. A small position value (SPV) rule has been applied with each iteration considering the binary position of each particle. BSO-LB analyzes task mapping onto under-loaded VMs, and the underloaded VMs.	Maximizing utilization with minimizing make-span and waiting time.	1)-Does not consider the increasing number of tasks and VMs. 2)-Does not consider other QoS metrics.
(Miao et al., 2021) In 2021	Zhang Miao, Peng Yong, Yang Mei, Yin Quanjun, and Xie Xu.	APDPSO: Adaptive Pbest discrete PSO.	APDPSO is a static load balancer. A particle changes its direction on the basis of two leaders, Pbest and Gbest. The hamming distance, which is used mainly in the signal processing is used in designing a new distance metric to update the position vectors (Pbest, Gbest) and update the velocity.	Balance computation load distribution and communication cost reduction.	1)-Does not consider other QoS metrics.
(Gupta et al., 2021) In 2021	Gupta, Abhishek, H. S. Bhadauria, and Annapurna Singh	Hyper heuristic Honey Bee-Based Load Balancing Technique	This technique is used where the honey bee represents the cloudlet. As honey bees search for food source, cloudlets will be allocated in VMs to be executed. To achieve better performance, the load of the best solution generated by this technique is evaluated using the VM capacity. Then, the load balancing will be done based on the current virtual machine processing time and the standard deviation of the load.	Enhance Makespan, and the processing time, and reduce the degree of imbalance.	1)-It does not consider other QoS performance metrics as the cost, the profit.

(*continued on next page*)

Table 3. (continued)

Ref. in Year	Authors	Algorithm	The technique	Advantages	Disadvantages
(Geetha and Parthasarathy, 2021) In 2021	R. Geetha, and V. Parthasarathy.	Integrated Genetic Algorithms (GA) and Artificial Neural Networks (ANN).	In this methodology, resource allocation is implemented through genetic algorithms and artificial neural networks by examining the hierarchical utilization with reference to job size and time taken to complete the jobs. The jobs with the same resource utilization and have less execution time are considered. The proposed system uses machine learning workflow to allocate the tasks to the available VMs.	Improve resource utilization, reduce execution time, and assigning priorities to the tasks present.	1)-Does not consider SLA negotiation in Cloud computing. 1)-Does not consider different pricing strategies.
(Negi et al., 2021) In 2021	Sarita Negi, Man Mohan Singh Rauthan, Kunwar Singh Vaisla, Neelam Panwar.	Clustering-based multiple objective dynamic load balancing (CMODLB).	The virtual machines (VMs) are initially clus tered into underloaded and overloaded VMs using Bayesian optimization-based enhanced K-means algorithm. Artificial neural network-based dynamic load balancing (ANN-LB) technique is implemented to such cluster. Further, the tasks are scheduled to underloading VMs using the variant of particle swarm optimization algorithm (TOP−SIS−PSO).	Improve completion time, makespan, and resource utilization.	1)-Does not consider other QoS metrics. 2)-Time complexity is higher.
(Thakur and Goraya, 2022) In 2022	Thakur, Avnish, and Major Singh Goraya	PPSO-DA: A phasor particle swarm optimization and dragonfly algorithm-based hybrid optimization algorithm	The implementation framework of the proposed hybrid optimization algorithm, PPSO-DA, is primarily based on PPSO which is a trigonometric variant of PSO incorporates with DA. The proposed framework is used to balance the load across active PMs and among their considered resource capacities.	Reduce the mean load imbalance and the mean resource capacity imbalance.	1)-Does not consider other QoS metrics. 3)-Time complexity is higher.

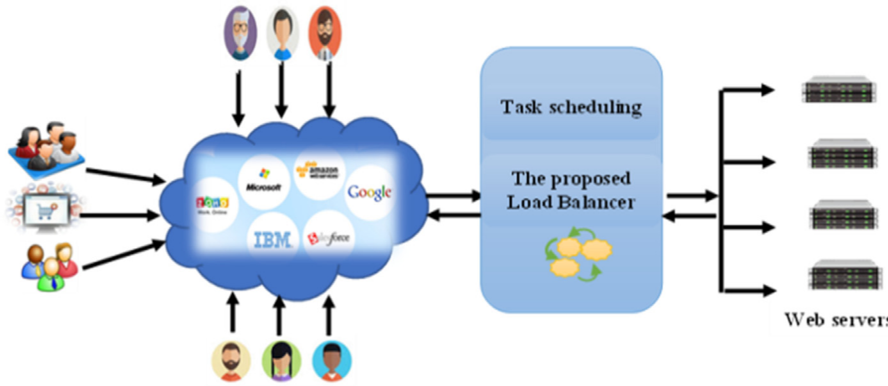

Fig. 1. The proposed ELBABCEβHC framework model.

request. The service broker selects the best data centers based on the service brokerage policy selected. In our proposed system, we consider the broker policy called the performance optimized routing algorithm where the broker maintains a list of the latest request processing times at each data center, then it adds the processing time of the best response data center to the appropriate network delay and selects the data center that would give the least total response time.

3.2. Important definitions

We assume collections of interdependent tasks T have been modelled as the set of n tasks $T = \{t_1, t_2, \ldots\ldots t_n\}$ to be allocated to the numbers of heterogeneous virtual machines $VMS = \{vm_1, vm_2, \ldots\ldots vm_m\}$.

Definition 1 (Tasks T): The task can be represented as t_j with j represents the identifier of t_j. These tasks have length $M = \{M_0, M_j\ldots, M_n\}$ measured in million instructions (MI) and the number of processing elements required for running tasks $P_e = \{p_{e_0}, p_{e_i}..p_{e_n}\}$.

Definition 2 (Virtual Machines VMS): These virtual machines with different processing cores (single-core and multi-core), and CPUs have different cycle times (Millions of Instructions Per Second (MIPS)). The virtual machine can be described as vm_i with i represents identifier, the processing speed measured in Million Instructions per Second ($MIPS$) per processing element is $Mp\{Mp_0, Mp_i\ldots Mp_m\}$, and $P_e = \{p_{e_0}, p_{e_i}\cdots p_{e_m}\}$ represents the number of processing elements in the virtual machines(vm_i).

Definition 3 (Network Latency τ_{NL}): This latency describes the time measured in milliseconds (ms). It is the sum of all possible delays including processing delay d_p, queuing delay (waiting time) d_q and propagation delay d_{pr}.

Definition 4 (Data Transfer Time τ_{TT}): This time that is calculated by dividing the size of the unit of data (in bits) by the available bandwidth (in bits/second).

Definition 5 (Response Time τ_{RT}): The time interval between the user request sent and the provider response received. It is the sum of the execution time τ_{ex} , the network latency τ_{NL} , the request (task) arrival time τ_{arr} and data transmission delay (data transfer time) τ_{TT} after the task is assigned to the virtual machine.

***Definition 6 (multi-objective optimization MOP*:**

A multi-objective optimization problem (Talbi, 2009) with the objective functions z $(z \geq 2)$ and feasible decision variables set S can be formulated as Equation (1):

$$PF^*(y) = \left\{ \begin{array}{c} \min F(y) = (f_1(y), f_2(y), \ldots\ldots.f_z(y)) \\ s.t. \\ y \in S \forall S = \{y_1, y_2, y_3, \ldots\ldots y_d\} \end{array} \right\} \quad (1)$$

Such problem can be expressed either maximum or minimum functions and can be transferred to each other by Equation (2):

$$max\{F(Y)\} \Leftrightarrow \min\{-F(Y)\} \quad (2)$$

When solving minimization problem, we obtain a set of solutions called a pareto-optimal or non-dominated. Equation (3) and Equation (4) should be satisfied for Pareto dominate:

$$1.\, f_\beta(y_1) \leq f_\beta(y_2) \quad \textit{for all indices } \beta \in \{1, 2, \ldots\ldots\ldots.z\} \quad (3)$$

$$2.\ f_\eta(y_1) < f_\eta(y_2) \quad \textit{for at least one index } \eta \in \{1, 2, \ldots..z\} \quad (4)$$

To deal with the multi-objective problem, the weighted sum is applied using a set of weighted coefficients ψ_i. The minimization formula for z objectives can be formulated as Equations (5) and (6):

$$F(X) = \min \sum_{i=1}^{z} [\psi_i \times f_i(X)] \quad for\ i\{1,2,3,\ldots\ldots z\} \tag{5}$$

$$\psi_i \geq 0, i \in \{1,2,3,\ldots\ldots z\} \tag{6}$$

3.3. Load balancing problem modeling

The proposed mathematical model for the enhanced load balancing algorithm can be formulated considering the following assumptions for simplicity:

(1) The execution of the tasks is a non-preemptive.
(2) All predecessor tasks should be completed before task execution.
(3) Due to the cloud heterogeneity, there are different execution times for the summited tasks on different virtual machines.

Let x_{ij} is defined as a binary decision variable given using Equation (7):

$$x_{ij} = \begin{cases} 1 & if\ task\ t_j\ assigned\ to\ vm_i \\ 0 & otherwies \end{cases} \tag{7}$$

We assume that task execution time τ_{ex} is constructed as $m \times n$ *matrix* using m of virtual machines *VMS* and n of tasks T. Let τ_{exij} be the execution time for task t_j corresponding to vm_i calculated using Equation (8):

$$\tau_{exij} = \frac{M_j}{Mp_i \times Pe_i} \tag{8}$$

The transfer time τ_{TTij} is determined using Equation (9)

$$\tau_{TTij}(t_k, t_j) = \frac{D_{kj}^{out}}{\Omega_{kj}} \tag{9}$$

The transfer time between the tasks running on the same virtual machine (vm) is 0.

The network latency $\tau_{NL}(vm_i, t_j)$ is determined using Equation (10)

$$\tau_{NLij} = d_{p_{ij}} + d_{q_{ij}} + d_{pr_{ij}} \tag{10}$$

The service response time τ_{RT} is the first objective function to be minimized for a single user request (task) t_j assigned to the virtual machine vm_i is calculated using (11):

$$\tau_{RTij} = \tau_{arrij} + \tau_{NLij} + \tau_{TTij} + \tau_{exij} \tag{11}$$

The total completion time $Çt_i$ of tasks allocated onto a virtual machine (vm_i) is calculated using Equation (12):

$$Çt_i = \sum_{j=1}^{n} \tau_{RTij} x_{ij} \forall i = 1, 2, ..m\ and\ j = 1, 2, ..n \tag{12}$$

The average completion time $Çt_{Av}$ is calculated by Equation (13):

$$Çt_{Av} = \frac{\sum_{i=1}^{m} Çt_i}{m} \forall i = 1, 2, \ldots.m \tag{13}$$

The makespan ($ɱʂ$) is calculated by Equation (14):

$$ɱʂ = \max_{i \in \{1\ldots m\}} (Çt_i) \forall i = 1, 2, 3\ldots\ldots.m \tag{14}$$

The second proposed objective function to be maximized is **the resource utilization ($A\hat{u}$)** can be calculated considering Equations 13 and 14 using Equation (15):

$$A\hat{u} = \frac{Çt_{Av}}{ɱʂ} \tag{15}$$

The third proposed objective function for minimization is **the data processing cost** ($C_d{}^{exe}$) can be calculated considering the completion time $Çt_i$ in Equation (12) and C_i^{exe} as the data processing cost per hour using Equation (16):

$$C_d{}^{exe} = \sum_{i=1}^{m} C_i^{exe} \times Çt_i \tag{16}$$

To solve the proposed load balancing related to task scheduling problem using weighted sum method in Equations (5) and (6), the multiobjective function ($F(y)$) can be formulated to minimize **the service response time** τ_{RT} and **the data processing cost** ($C_d{}^{exe}$) while **the resource utilization ($A\hat{u}$)** is maximized as Equation (17) with the constraints that should be satisfied as in Equations 18–20 can be modeled as:

$$\min(F) = \psi_1 \psi_2 \tau_{RT} + (1 - \psi_1) C_d{}^{exe} + (1 - \psi_2)(-A\hat{u}) \tag{17}$$

s.t.

$$\sum_{i=1}^{m} x_{ij} = 1 \forall t_j \tag{18}$$

$$\sum_{j=1}^{n} a_{ij} x_{ij} \leq pp_i \forall vm_i \tag{19}$$

$$C_d{}^{exe} \leq Budget(Application) \tag{20}$$

The constraints represent the tasks requirements and the cloud resources availability. The first one in Equation (18) assures the summation of task t_i to only one virtual machine vm_i. Secondly, Equation (19) guarantees that the resources required for all tasks assignment to the virtual machine vm_I don't exceed the processing power pp_i of vm_i. Thirdly, at Equation (20), the constraint ensures that the total processing cost must be less or equal to the dedicated budget to that application.

4. The proposed algorithm for load balancer based on hybrid artificial bee colony

Our proposed load balancer called enhanced load balancing approach based on hybrid artificial bee colony with enhanced β-Hill climbing (ELBAB-CEβHC) is composed of multi-phases: ranking list, population initialization and task allocation phases.

In the ranking list phase, a list of the submitted tasks is built using heterogeneous earliest finish time (HEFT) algorithm (Mazrekaj et al., 2019) based on the average execution time of each task on the virtual machines available. When initialize population, a greedy randomized adaptive search procedure (GRASP) (Resende and Ribeiro, 2019) is used. In the task allocation phase, the artificial bee colony algorithm (BABC) (Karaboga, 2005; Karaboga and Basturk, 2008) is used. The local search of the employee bees in BABC algorithm is enhanced by hybridizing BABC with the enhanced β-hill climbing algorithm (Al-Betar, 2017) that is combined with one chaotic sequence called Sinusoidal iterator (Peitgen et al., 2006; Lu et al., 2014). The local search of the onlooker bees in BABC is enhanced by using the mutation operator (Poli et al., 2008) for maintaining diversity in a population. The mutation (swap) is applied in the solution among the tasks at the virtual machine of the maximum completion time and the tasks at the virtual machine with minimum completion time to have the neighborhood solutions (food sources).

4.1. The ranking list strategy

In this phase, the tasks are ranked considering descending order suggested in (Mazrekaj et al., 2019). The ranking values $(rank(t_i))$ are calculated as Equation (21):

$$\boldsymbol{rank}(\boldsymbol{t_j}) = \boldsymbol{AVG}(\boldsymbol{ET_j}) + \max_{k \in succ(t_j)} (\boldsymbol{TT_{kj}} + \boldsymbol{rank}(\boldsymbol{t_k})) \tag{21}$$

4.2. Initialization phase using GRASP

GRASP (Resende and Ribeiro, 2019) presented in Algorithm 1 is used for initialization of the solutions (food sources) of the population.

Algorithm 1. Greedy randomized adaptive search procedure.

1. Input (Tasks, Virtual Machines).
2. Repeat
2.1. For *Tasks* $t = t_1, t_2, \ldots..t_n$ do
2.1.1. For *Machines* $VMS = \{vm_1, vm_2, \ldots vm_m\}$ do
2.1.2. Choose randomly vm_i for t_j.
2.1.3. *Solution* ← *Greedy Randomized Construction*.
2.1.4. *If solution is not feasible then*
2.1.5. *Solution* ← *Repair*(*Solution*).
2.1.6. End for.
2.2. End for.
2.3. *Solution* ← *Local Search*(*Solution*).
2.4. Update Solution (Solution, Best Solution).
2.5. Memorize the feasible solution (food source).
Until (The population is constructed).

4.3. Artificial bee colony algorithm

Algorithm 2. Standard BABC algorithm for task scheduling

1. Initialize
2. Repeat
2.1. Place employed bees on the food sources(solutions) in the search area.
2.2. Evaluate Fitness using Equations 17–20.
2.3. Apply roulette wheel selection to get fitness probability.
2.4. Place onlooker bees on the food sources (solutions) of higher probabilities.
2.5. For discovering new food sources(solutions), send the scouts to the search area.
3. UNTIL (requirements are met).

The binary artificial bee colony (BABC) (Karaboga, 2005; Karaboga and Basturk, 2008) is presented in (Algorithm 2).

The bee colony in ABC contains three groups of bees: employed bees assigned to each feasible solution (food source), onlooker bees watch the waggle dance of employed bees in the dance area within the hive to choose the better food sources, and scout bees search for the food sources randomly. The nectar amounts of these food sources correspond to the fitness of the associated solutions which can be calculated using the multi-objective function $\mathbf{min}(F)$ in Equations 17–20. A scout bee will search for a new solution randomly.

4.4. β-Hill climbing algorithm enhanced with sinusoidal map

The enhanced β-Hill Climbing algorithm with the sinusoidal map strategy (Peitgen et al., 2006; Karaboga and Basturk, 2008) is presented in Algorithm 3.

Algorithm 3. Enhanced β-Hill Climbing algorithm

1. Initialization
1. $x_i = l_{bi} + (u_{bi} - l_{bi})*U(0,1), \forall i = \{1,2,\ldots,m\}$.
2. Evaluate fitness function $f(x_i)$ using Equations 17–20.
3. *Set* $\mathscr{R} = \{r_1,..r_i...r_m\}, \forall R \in \{-1,1\}$ using Equation (23).
2. $Itr = 0$.
3. While ($Itr \leq MaxItr$) do
1. $\acute{x}_i = x_i \pm U(0,1)*(x_i - x_k), \forall i,k = \{1,2,.,m\}, i \neq k$.
2. *For* $i = 1,\ldots m$ *do*
2.1. *If* $r_i \leq \beta$ *then*
2.1.1. $\acute{x}_i = l_{bi} + (u_{bi} - l_{bi})*U(0,1)$
2.2. *End if*.
3. End for.
4. Evaluate fitness function $f(\acute{x}_i)$ using Equations 17–20.
5. If $(f(\acute{x}_i) \leq f(x_i))$ then
5.1. $x_i = \acute{x}_i$.
5.2. *End if*.
6. $Itr = Itr + 1$.
4. End while.

β-Hill climbing algorithm is used to enhance the local search in ELBABCEβHC algorithm considering the load of each virtual machine (vm_i) calculated as the total completion time $(Çƫ_i)$. β-hill climbing algorithm iteratively generates a new solution based on two operators: N − operator and β − operator. N-operator navigates the neighboring solutions of the current one which is considered as the source of exploitation. β-operator allows moving from one region to another based on the comparison with $\mathscr{R} = \{r_1, r_2, \ldots r_i \ldots r_N\}$ as a random set of values. β-operator is utilized in hill climbing as the source of the exploration.

The random set $\mathscr{R}$ is generated using the sinusoidal map of the total completion time $Çƫ_i$ of tasks that assigned onto each virtual machine (vm_i).

Table 4. Region definitions.

Continents	Id	Time Zone	Peak Hours (GTM) (Local Time)
North America	0	GMT − 6.00	7.00–9.00 pm
South America	1	GMT − 4.00	7.00–9.00 pm
Europe	2	GMT + 1.00	7.00–9.00 pm
Asia	3	GMT + 6.00	7.00–9.00 pm
Africa	4	GMT + 2.00	7.00–9.00 pm
Oceania	5	GMT + 10.00	7.00–9.00 pm

Table 6. Data center configuration.

Parameter	Parameter values
Number of data centers	3
Name	DC1,DC2,DC3
Region	DC1 at 0, DC2 at 3, DC3 at4.
Arch	X86
OS	LINUX
Virtual machine hypervisor management(VMM)	Xen
Cost per vm ($/Hr)	2.224
Memory Cost($/s)	0.55
Storage Cost($/s)	1.5
Data Transfer Cost ($/Gb)	0.1
Physical HW Units	40

Table 7. Virtual machines configuration.

Parameter	Parameter Values
VM Image Size	10000
VM Memory	512
VM Bandwidth	1000
Memory per Machine	{2048MB, 2096MB}
Storage per machine	100000
Number of processors	{2, 4}
VM Policy	Time Shared
User Grouping Factor	1000
Request Grouping Factor	100
Executable Instruction Length	250

$Çƫ_i$ is generated using sinusoidal function as Equation (22):

$$r_i = Sin\,(Çƫ_i), \forall Çƫ_i \in [-1,1], \forall i = 1,2,\ldots m \tag{22}$$

4.5. *The proposed ELBABCEbHC algorithm for scheduling with load balancing*

The proposed enhanced load balancing approach based on hybrid artificial bee colony with enhanced β-Hill climbing combined with sinusoidal map (ELBABCEβHC) algorithm is presented in Algorithm 4.

Table 5. User Base configuration.

	Region	Request per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
User Base 1	0	12	100	13	15	1,200,000	120,000
User Base 2	1	12	100	15	17	2,950,000	295,000
User Base 3	2	12	100	20	22	2,200,000	220,000
User Base 4	3	12	100	1	3	3,336,000	333,600
User Base 5	4	12	100	21	23	2,120,000	212,000

Algorithm 4. The proposed ELBABCEβHC load Balancing with task scheduling algorithm.

1. Build task priority list using Equation (21).
2. Initialize food source populations (feasible solutions) using GRASP in Algorithm 1.
3. Repeat
3.1. Place the employee bees on the food sources in the search area.
3.2. **For each** employee bee in the population
3.2.1. Evaluate the current solution fitness function using Equations 17–20.
3.2.2. Search the solution neighborhood using Enhanced β-Hill Climbing algorithm in Algorithm 2.
3.2.3. Memorize the best solution.
3.3. Place the onlooker bees on the food sources in the search area.
3.4. **For each** onlooker bee in the population
3.4.1. Evaluate the current solution fitness function using Equations 17–20.
3.4.2. Search the neighborhood solution using Bit Mutation Operator considering machines load.
3.4.3. Memorize the best solution.
3.4.4. Abandon the worst solutions.
3.4.5. Send scout bee to explore search area for new solutions using GRASP in Algorithm 1.
3.4.6. Relace the abandon solutions with the new solutions.
3.4.7. Sorting the best-found solutions in the population.
3.4.8. Select the best non-dominated solution.
4. Until (stopping criteria is met.)

5. The experiments results and analysis

The overall experimental setup, performance metrics, results, and analysis are described in this section to evaluate the proposed enhanced load balancing approach based on hybrid artificial bee colony with enhanced β-Hill climbing (ELBABCEβHC) algorithm.

5.1. Environment setup

The experiments were carried out by the simulators CloudSim version-3.0 and CloudAnalyst and using NetBeans IDE Version 8.0.2 (Wickremasinghe et al., 2010). The experimental environment running on Microsoft Windows 10 including Intel(R)-Core(TM)i7-7500U-2.70 GHz processor and 16.0 GB RAM. Our proposed simulation carries the proposed load balancing algorithm ELBABCEβHC coded in Java and evaluated by cloudAnalyst.

In cloudAnalyst framework, the environment parameters are defined in Tables 4–8. The regions definitions are defined in Table 4 where the world is divided into six regions according to the six continents. Our simulation Based on Facebook Users distribution around the world in 2021 presented in (https://worldpopulationreview.com/country-rankings/facebook-users-by-country). Assuming the requests are in the evening within a single time zone, the userbases configuration scaled by 0.1 are suggested in Table 5. Amazon EC2 instance pricing model in (https://aws.amazon.com/ec2/pricing/on-demand/) is used.

We also define the main characteristics of the data centers in Table 6 whereas the virtual machines configuration is described in Table 7 with other parameters such as: (1) User grouping factor as the users count in a single bundle. (2) Request Grouping Factor for a virtual machine of the requests number in a single bundle. (3) The length of the executable instruction (or request) parameter is defined in bytes. (4)A broker policy selected is the performance optimized routing algorithm. The suggested configuration for the proposed approach ELBABCEβHC is provided in Table 8.

Table 8. The configuration of the proposed ELBABCEβHC algorithm.

Population size	40
Employee Bees	20
Onlooker bees	20
Scout bees	Replace the worst solution.
The maximum iteration	100
Limit	5

Table 9. The response time by region for the proposed approach ELBABCEβHC (ms).

UserBases	Avg	Min	Max
UB1	79.34	62.45	91.78
UB2	320.85	244.35	391.87
UB3	358.43	271.67	430.56
UB4	91.57	87.23	143.89
UB5	130.87	105.32	195.85

Table 10. The request service time(ms) of different data centers using the proposed ELBABCEβHC.

Data center	Avg	Min	Max
DC1 at Region 0	60.66	18.45	82.67
DC2 at Region 3	75.447	25.46	95.46
DC3 at Region 4	43.76	11.57	64.042

Table 11. The overall response time(ms) obtained by RR, TLB, AMLB, and the proposed ELBABCEβHC.

Load balancer	Overall response time (ms)		
	Avg	Min	Max
Round Robin (RR)	250.71	98.36	360.34
Throttled load Balancer (TLB)	170.78	56.61	331.13
Active Monitoring Load Balancing (AMLB)	198.89	76.53	346.13
The proposed ELBABCEβHC	155.31	40.53	317.23

Table 12. The data center processing cost ($) obtained by RR, TLB, AMLB, and the proposed ELBABCEβHC.

Load balancer	Data processing cost ($)		
	DC1	DC2	DC3
Round Robin (RR)	13,841.09	17,654.69	9680.54
Throttled load Balancer (TLB)	10,613.07	13,200.21	7656.25
Active Monitoring Load Balancing (AMLB)	12,341.08	16,154.68	8180.54
The proposed ELBABCEβHC	10,041.45	12,354.75	7080.56

5.2. Experimental results

To measure the effectiveness of our proposed enhanced load balancing approach based on hybrid artificial bee colony with enhanced β-Hill climbing (ELBABCEβHC) algorithm, the performance of the proposed algorithm is compared with Round Robin (RR), Throttled load Balancer (TLB), and Active Monitoring Load Balancing (AMLB). These algorithms are implemented using five UserBases (UserBase 1: UserBase 5) in CloudAnalyst simulators. Tables 9—13 show the experimental results obtained by the proposed ELBABCEβHC and the other counterparts algorithms considering various parameters. The response time by the region for the proposed approach ELBABCEβHC (ms) is shown in Table 9. The request service time(ms) of different data centers using the proposed ELBABCEβHC is shown in Table 10. Further, the overall response time in milliseconds is presented in Table 11. The data processing cost ($) obtained is presented in Table 12. The data center utilization (%) is shown in Table 13.

5.3. Performance evaluation

A statistical analysis is described by Figs. 2—6. The average response times of the UserBases (UB1:UB5) using the proposed approach ELBABCEβHC are shown in Fig. 2.

It is obvious that UB1, UB4 and UB5 are processed in the closest data centers which reduce the

Table 13. The data center utilization (%) obtained by RR, TLB, AMLB, and the proposed ELBABCEβHC.

Load balancer	Utilization (%)		
	DC1	DC2	DC3
Round Robin (RR)	85.69	91.54	69.34
Throttled load Balancer (TLB)	79.23	87.86	61.67
Active Monitoring Load Balancing (AMLB)	82.65	89.23	63.98
The proposed ELBABCEβHC	76.78	85.32	56.67

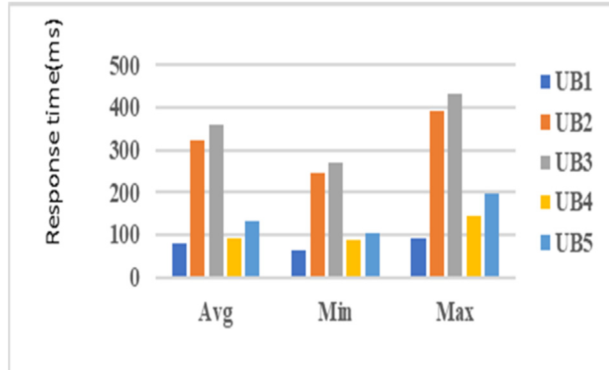

Fig. 2. The response time by region for the proposed approach ELBABCE β HC(ms).

transmission latency and in consequence the overall response time. As a result, UB1, UB4 and UB5 have better overall response time compared to UB2 and UB3. The request service times of different data centers using the proposed ELBABCEβHC are shown in Fig. 3. The data center (DC3) at region 4 (Africa) gives minimum request service time compared with DC1, DC2 because the number of userbases requests that are processed by DC3 is the minimum. The data center (DC1) at region 0(North America) has more requests to serve than DC3. The data center (DC2) at region 3(Asia) has the maximum request service time because it serves the greatest number of UserBases requests (UB3, UB4). Fig. 4 shows the overall response time obtained by RR, TLB, AMLB, and ELBABCEβHC.TLB is better than AMLB, and RR algorithms. ELBABCEβHC outperforms the three counterpart algorithms in terms of response time. Fig. 5 shows the data center processing cost obtained by RR, TLB, AMLB, and ELBABCEβHC. TLB is better than AMLB and RR but ELBABCEβHC outperforms the three

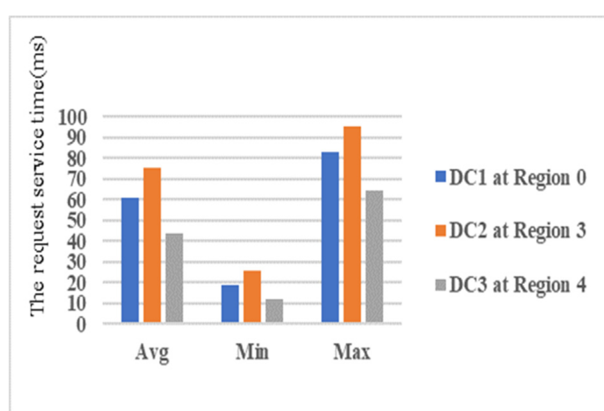

Fig. 3. The request service time(ms) of different data centers using the proposed ELBABCEβHC.

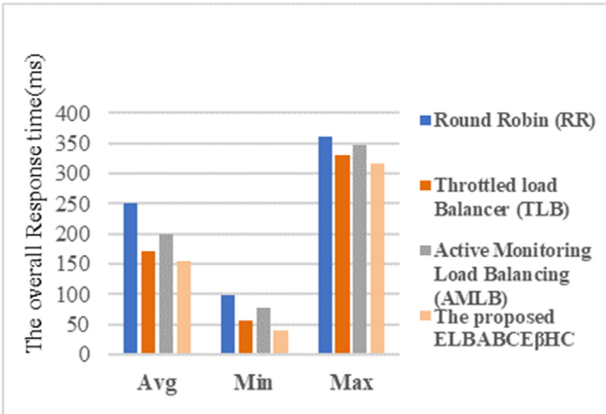

Fig. 4. The overall response time(ms) obtained by RR, TLB, AMLB and the proposed ELBABCEβHC.

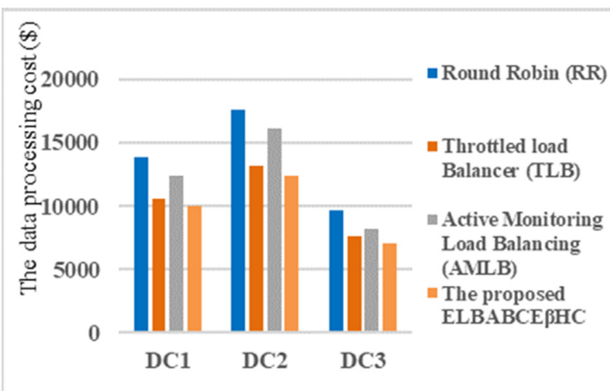

Fig. 5. The data center processing cost (S) obtained by RR, TLB, AMLB and the proposed ELBABCEβHC.

algorithms in terms of data processing cost. Fig. 6 shows that ELBABCEβHC gives the best results in terms of resource utilization considering three data centers distributed in three regions (North America,

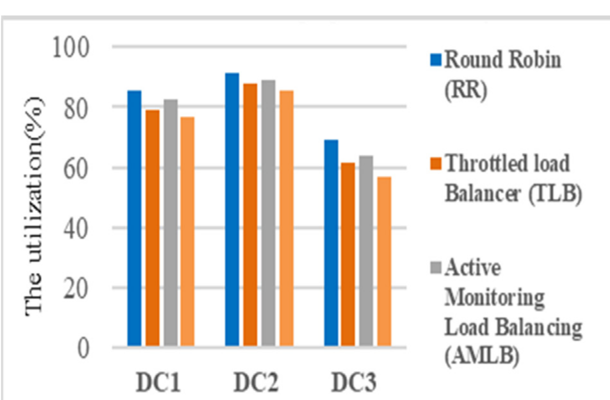

Fig. 6. The data center utilization (%) obtained by RR, TLB, AMLB and the proposed ELBABCEβHC.

Asia, and Africa). TLB algorithm is better than AMLB algorithm and the RR algorithm.

6. Conclusion

Load balancing is an essential issue for enhancing the overall service response time for the users in cloud with achieving minimum processing cost and maximum utilization taking into account the quality of services (QoS) and the service level agreement (SLA). The main issue is that the increasing in the overall response time causing the services cost raise due to the searching for the least loaded virtual machine. In our proposed approach for dynamic load balancing, we use the population-based metaheuristic. First, the population is initialized using greedy randomized adaptive search procedure (GRASP). Further, the local search in the binary artificial bee colony (BABC) algorithm is enhanced using β-hill climbing algorithm combined with sinusoidal sequence considering the virtual machine vm_i load. For maintaining the diversity in the population, the bit inversion mutation is used in the onlooker bees in BABC. According to the simulation results, it is obvious that the proposed algorithm ELBABCEβHC as a scheduling algorithm with load balancer gives the best results in terms of the overall response time, the data processing cost ($) and the data center utilization compared with Round Robin (RR), Throttled load Balancer (TLB), and Active Monitoring Load Balancing (AMLB) for load balancing of all the proposed five user groups requests in the globe using the data centers that are distributed in three regions (North America, Asia, and Africa).

7. The future work

The future work is to investigate other types of metaheuristics algorithms for solving load balancing issues related to the task scheduling problems in the cloud environment.

Authors contribution

We encourage authors to submit an author statement outlining their individual contributions to the paper using the relevant roles: 1-Conception or design of the work (Maha Zeedan, Gamal Attiya, and Nawal El-Fishawy). 2-Data collection and tools (Maha Zeedan). 3-Data analysis and interpretation (Maha Zeedan, Gamal Attiya, and Nawal El-Fishawy). 4-Investigation (Maha Zeedan, Gamal Attiya, and Nawal El-Fishawy). 5-Methodology (Maha Zeedan, Gamal Attiya, and Nawal El-Fishawy). 6-Project administration (Maha Zeedan, Gamal Attiya, and Nawal El-Fishawy). 7-Resources (Maha Zeedan, Gamal Attiya, and Nawal El-Fishawy). 8-Software

(Maha Zeedan)). 9-Supervision (Gamal Attiya, and Nawal El-Fishawy). 10-Drafting the article (Maha Zeedan, Gamal Attiya, and Nawal El-Fishawy). 11-Critical revision of the article (Maha Zeedan, Gamal Attiya, and Nawal El-Fishawy). 12-Final approval of the version to be published (Maha Zeedan, Gamal Attiya, and Nawal El-Fishawy).

The corresponding author is responsible for ensuring that the descriptions are accurate and agreed by all authors.

Funding statement

No funding or granting was received to assist with the preparation of this manuscript.

Conflicts of interest

Authors declare that they have no conflict of interest with respect to the research, authorship or publication of this article.

References

Al-Betar, M.A., 2017. β-Hill climbing: an exploratory local search. Neural Comput. Appl. 28 (Suppl 1), 153–168.
Al-Betar, M.A., Aljarah, I., Awadallah, M.A., Faris, H., Mirjalili, S., 2019. Adaptive β-hill climbing for optimization. Soft Comput. 23 (24), 13489–13512.
Arram, A., Ayob, M., Zakree, M., 2014. Comparative study of meta-heuristic approaches for solving traveling salesman problems. Asian J Appl Sci 7, 662–670.
Balaji, K., Sai Kiran, P., Sunil Kumar, M., 2021. An energy efficient load balancing on cloud computing using adaptive cat swarm optimization. Mater. Today Proc. https://doi.org/10.1016/J.MATPR.2020.11.106.
Buyya, R., Broberg, J., Goscinski, A., 2011. Cloud Computing: Principles and Paradigms. John Wiley & Sons, Inc; Hoboken, New Jersey/Canada, pp. 23–24.
Francis Saviour, D.A., et al., 2020. Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments. J. Parallel Distr. Comput. 142, 36–45.
Geetha, R., Parthasarathy, V., 2021. An advanced artificial intelligence technique for resource allocation by investigating and scheduling parallel-distributed request/response handling. J. Ambient Intell. Hum. Comput. 12, 6899–6909.
Gupta, A., Bhadauria, H.S., Singh, A., 2021. Load balancing based hyper heuristic algorithm for cloud task scheduling. J. Ambient Intell. Hum. Comput. 12, 5845–5852.
Houssein, E.H., Gad, A.G., Wazery, Y.M., Suganthan, P.N., 2021. Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. Swarm Evol. Comput. 62, 100841.
https://aws.amazon.com/ec2/pricing/on-demand/.
https://worldpopulationreview.com/country-rankings/facebook-users-by-country.
Karaboga, D., 2005. An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report-Tr06, vol. 200. Erciyes University, Engineering Faculty, Computer Engineering Department.
Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. 39, 459–471.
Karaboga, D., Basturk, B., 2008. On the performance of artificial bee colony (ABC) algorithm. Appl. Soft Comput. 8 (1), 687–697.
Kim, S.S., et al., 2017. Cognitively inspired artificial bee colony clustering for cognitive wireless sensor networks. Cogn Comput 9, 207–224.
Lu, H., et al., 2014. The effects of using chaotic map on improving the performance of multiobjective evolutionary algorithms. Math. Probl Eng. 2014, 924652.
Luke, S., Edition, Zeroth, 2009. "Essentials of Metaheuristics.".
Mazrekaj, A., et al., 2019. The Experiential Heterogeneous Earliest Finish Time Algorithm for Task Scheduling in Clouds. CLOSER.
Mell, P., Grance, T., 2011. The NIST Definition of Cloud Computing, Document 800–145. Nat. Inst. Stand. Technol, Gaithersburg, MD, USA.
Miao, Z., et al., 2021. A discrete PSO-based static load balancing algorithm for distributed simulations in a cloud environment. Future Generat. Comput. Syst. 115, 497–516.
Mishra, K., Majhi, S., 2020. A state-of-art on cloud load balancing algorithms. International Journal of computing and digital systems 9, 201–220.
Mishra, K., Majhi, S.K., 2021. A binary bird swarm optimization-based load balancing algorithm for cloud computing environment. Open Comput Sci 11 (1), 146–160.
Negi, S., et al., 2021. CMODLB: an efficient load balancing approach in cloud computing environment. J. Supercomput. 77, 8787–8839.
Peitgen, H.O., Jürgens, H., Saupe, D., 2006. Chaos and Fractals: New Frontiers of Science. Springer Science & Business Media.
Pham, D.T., Castellani, M., 2015. A comparative study of the Bees Algorithm as a tool for function optimisation. Cogent Eng 2, 1091540.
Phi, N.X., et al., 2018. Proposed load balancing algorithm to reduce response time and processing time on cloud computing. Int J Comput Netw Commun 10, 87–98.
Poli, R., et al., 2008. A Field Guide to Genetic Programming. Lulu.com.
Ranjan, R., Buyya, R., 2010. Decentralized overlay for federation of enterprise clouds. In: Handbook of Research on Scalable Computing Technologies (pp. 191–217). IGI Global.
Resende, M.G., Ribeiro, C.C., 2019. Greedy randomized adaptive search procedures: advances and extensions. In: Handbook of Metaheuristics, pp. 169–220.
Singh, A.N., Prakash, S., 2018. WAMLB: Weighted Active Monitoring Load Balancing in Cloud Computing. Big Data Analytics. Springer, Singapore, pp. 677–685.
Talbi, E.G., 2009. Metaheuristics: from Design to Implementation, vol. 4. John Wiley & Sons, pp. 308–319.
Thakur, A., Goraya, M.S., 2022. RAFL: a hybrid metaheuristic-based resource allocation framework for load balancing in cloud computing environment. Simulat. Model. Pract. Theor. 116, 102485.
Wickremasinghe, B., Calheiros, R.N., Buyya, R., 2010, April. Cloudanalyst: a cloudsim-based visual modeller for analysing cloud computing environments and applications. In: 2010 24th IEEE International Conference on Advanced Information Networking and Applications. IEEE, pp. 446–452.
Yuret, D., De La Maza, M., 1993. Dynamic Hill Climbing: Overcoming the Limitations of Optimization Techniques. Second Turk Symp Artif Intell Neural Netw. Citeseer.
Zahid, M., et al., 2018. Hill Climbing Load Balancing Algorithm on Fog Computing. International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. Springer, Cham.