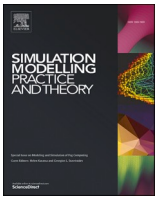
Simulation Modelling Practice and Theory 116 (2022) 102485

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat



RAFL: A hybrid metaheuristic based resource allocation framework for load balancing in cloud computing environment

Avnish Thakur *, Major Singh Goraya

Sant Longowal Institute of Engineering and Technology, Longowal, India

ARTICLE INFO

Keywords:
Cloud computing
Load balancing
Virtual machine placement
Phasor particle swarm optimization
Dragonfly algorithm

ABSTRACT

This paper proposes a hybrid metaheuristic based resource allocation framework named RAFL for load balancing in cloud computing environment. The objective is to proactively minimize the load imbalance across active physical machines and among their considered resource capacities (e.g., CPU and RAM). This prevents overloading/underloading of active physical machines and utilizes their considered resource capacities in a balanced manner. In the proposed framework, a phasor particle swarm optimization and dragonfly algorithm based hybrid optimization algorithm named PPSO-DA is used to generate an optimal resource allocation plan for balancing the load. Simulation experiments are performed using CloudSim simulator to measure the metrics of load imbalance across active physical machines and among their considered resource capacities. Results show that the proposed PPSO-DA algorithm outperforms phasor particle swarm optimization, dragonfly algorithm, comprehensive learning particle swarm optimization, memory based hybrid dragonfly algorithm, sine cosine algorithm, and elephant herding optimization, in finding an optimal resource allocation for balancing the load. The statistical analysis and benchmark testing also validates the relative superiority of PPSO-DA.

1. Introduction

Cloud computing [1–3] aims to provide on-demand ubiquitous virtual access to the data center resources, over the Internet. Cloud computing services minimize the need of hardware and software resources at consumer end. Rapidly growing adoption of cloud services by commercial and non-commercial users have greatly increased the load on data centers. Efficient workload distribution is imperative to optimally utilize the data center resources and provide quality services to the consumers. Inefficient distribution of workload may lead to overloading/underloading of active physical machines (PMs) in the data center and imbalanced usage of their resource capacities, e.g., CPU, GPU, and RAM. Overloading of a PM degrades its performance [4], whereas underloading leads to resource wastage [5]. Imbalanced usage of different resource capacities in a PM may lead to a situation where it might get exhausted of some of its resource capacities while having other resources in an adequate amount [6,7]. For example, CPU utilization of a PM may reach its maximum threshold while the utilization of its RAM and GPU are 60% and 40%, respectively, of their maximum thresholds. This prevents further deployment of tasks on the machine in spite of having some resource capacities in an adequate amount. In the proposed Resource Allocation Framework for Load balancing called RAFL, the virtual machine (VM) placement plan is generated with an objective to minimize load imbalance across active PMs and among their different considered resource capacities.

In literature [8–13] it is observed that in most of the frameworks, reactive load balancing approach is used, i.e., the load balancing

* Corresponding author.
E-mail address: avi.himachal@gmail.com (A. Thakur).

https://doi.org/10.1016/j.simpat.2021.102485
Received 13 June 2021; Received in revised form 29 December 2021; Accepted 30 December 2021
Available online 2 January 2022

procedures are executed in reaction to load imbalance. Task and VM migration are the primarily adopted approaches to balance the load which are time and resource intensive [14,15]. The proposed framework, RAFL uses proactive load balancing approach which aims to minimize load imbalance across active PMs and among their considered resource capacities while allocating the resources.

Load balancing in this paper is formulated as a multidimensional bin-packing problem which is considered as a NP-hard combinatorial optimization problem [16]. Metaheuristic based optimization algorithms are typically more viable in finding an optimal solution to NP-hard problems [17,18]. The proposed framework, RAFL uses a hybrid optimization algorithm named PPSO-DA with a load balancing objective to generate an optimal resource allocation plan for the VMs required to execute the given batch of tasks in each scheduling cycle. PPSO-DA hybridizes phasor particle swarm optimization (PPSO) [19] and dragonfly algorithm (DA) [20] to explore and exploit the search space in an effective and efficient manner to find an optimal solution. PPSO-DA utilizes the self-learning and social-learning approaches of PPSO to attract the search agents towards promising regions of the search space, and the feature of DA to distract the search agents outward the regions of the worst solution. The convergence rate of PPSO is faster compared to DA due to relatively high exploration in DA. The PPSO-DA algorithm uses the implementation framework of PPSO and includes the feature of DA to distract the search agents away from the region of worst solution using a novel approach. The magnitude of distraction experienced by a search agent is inversely proportional to its distance from the location of worst solution. An adequate balance between exploration and exploitation phases at different stages of the search process is orchestrated using a dynamic distraction coefficient.

The prime contributions of this paper are as follows:

- A resource allocation framework, RAFL is proposed to proactively balance the load across active PMs and among their considered resource capacities.
- A hybrid metaheuristic based optimization algorithm, PPSO-DA is proposed to explore and exploit the search space in an effective and efficient manner.
- The resource allocation plan is generated simultaneously for the batch of multiple VMs in each scheduling cycle.

The moving averages of load imbalance across active PMs and among their considered resource capacities are used as the measures of performance in the proposed framework. In RAFL, PPSO-DA outperforms the related baseline algorithms (i.e., PPSO and DA), comprehensive learning particle swarm optimization (CLPSO) [21], memory based hybrid dragonfly algorithm (MHDA) [22], sine cosine algorithm (SCA) [23], and elephant herding optimization (EHO) [24] in balancing load across active PMs and among their considered resource capacities (i.e., CPU and RAM).

The rest of this paper is organized as follows: Section 2 discusses the related work. System model and problem formulation are presented in Section 3. Section 4 discusses the preliminary optimization algorithms, i.e., PPSO and DA. Section 5 explains the working principle of the proposed hybrid optimization algorithm, PPSO-DA. Section 6 first explains the experiment setup followed by discussion and comparative analysis of the results. Section 6 also performs the statistical analysis and benchmark testing of the compared algorithms. Section 7 concludes the work presented in this paper.

2. Related work

Related work is organized around heuristic and metaheuristic based load balancing frameworks proposed for cloud computing environment and the hybrid optimization algorithms based on DA and PSO variants. Authors in [25] proposed a multi-capacity aware host selection algorithm using bin-packing approach. This heuristic aims to minimize load imbalance across different considered resource capacities in the active PMs. A task is allocated resources on a PM whose resource availabilities best matches the task's resource requirements. Authors in [26] proposed an algorithm based on first-fit and best-fit heuristics for scheduling workload in cloud computing environment. Proposed algorithm is easy to implement and quite fast to provide an immediate solution but do not guarantee to minimize load imbalance across different resource capacities of a PM. Authors in [27] proposed a task scheduling framework to improve resource utilization, average response time, load balancing, and deadline violation rate. In this framework tasks are classified into three categories: CPU intensive, memory intensive and input/output intensive, in accordance to their resource requirements. PMs are sorted in three lists: CPU intensive, memory intensive, and input/output intensive, in the decreasing order of load on their CPU, memory, and input/output resources, respectively. Tasks are allocated the required resources by searching their related category PM list using first-fit algorithm. This framework do not consider the combination of different resources for the categorization of tasks and sorting of PMs which can lead to resource capacity imbalance in the active PMs. The VM placement framework proposed in [28] aims to maximize resource utilization and minimize the power usage. The PMs to place the requested VMs are selected such that the load imbalance across multiple resource capacities is minimized. This framework periodically sorts the active PMs in accordance to their resource utilization, and then consolidates the workload by transferring VMs from the PMs with low resource utilization onto the other suitable PMs in order to minimize the number of active PMs. This altogether minimizes power consumption and improves resource utilization. Authors in [29] proposed a VM placement framework to achieve the desired objectives of load balancing, resource utilization, and energy conservation. The framework employs best-fit resource allocation policy to select an appropriate PM for the placement of a VM. A VM migration policy is used to further minimize load imbalance across active heterogeneous PMs, especially when an overload condition occurs.

Metaheuristic based optimization algorithms are extensively used in finding an optimal solution for problems with large search space as in case of load balancing in cloud computing. Authors in [30] have proposed an ant colony system based optimization algorithm to allocate physical resources for the requested VM configurations. The objective is to simultaneously minimize resource

wastage and power consumption. The proposed algorithm aims to deduce an optimal VM placement plan to utilize different resource capacities of active PMs in a balanced manner in order to minimize the resource wastage, whereas to minimize power consumption it also simultaneously aims to minimize the number of active PMs. Authors in [31] proposed an ant colony system based VM consolidation framework to minimize energy consumption of the data center while maintaining the required quality of service. This framework uses live migration of VMs to realize the VM consolidation plan generated by the ant colony system based optimization algorithm, using minimal number of VM migrations. Simulation results show that the proposed VM consolidation approach minimizes energy consumption and improves resource utilization. This framework also resolves the problem of overloading/underloading of PMs. The load balancing framework proposed in [32] aims to maximize the resource utilization of active PMs and minimize power consumption in the data center. The framework uses a modified genetic algorithm to optimally allocate the PMs for hosting the required VMs. The VM placement problem in [33] has been formulated as a bin-packing problem with a primary objective to utilize multiple resources of a PM in a balanced manner. A genetic algorithm based approach has been used to deduce the resource allocation plan for the VMs to successfully execute the given set of tasks. Authors in [34] have proposed a task migration based load balancing. The extra tasks are migrated from an overloaded VM to a compatible and suitable VM, preferably on an underloaded PM. A particle swarm optimization (PSO) based algorithm has been implemented to deduce an optimal VM reallocation plan for the tasks to be migrated. The major challenge in this approach is the availability of compatible VMs on the suitable PMs. Authors in [35] have proposed a PSO based task scheduling framework for cloud computing environment. The primary objective is to minimize the makespan time imbalance across active heterogeneous VMs and to maximize the resource utilization. The framework uses a PSO based optimization algorithm to deduce an optimal plan to place the given set of tasks onto the appropriate VMs to achieve the desired objectives.

In literature, DA and PSO based hybrid optimization algorithms have been proposed to solve different engineering problems. Authors in [22] proposed a DA and PSO based hybrid optimization algorithm which aims to utilize the exploration capability of DA and the exploitation capability of PSO to obtain an optimal solution. The original DA has been modified to keep track of the personal best solutions found by the dragonflies. DA and PSO algorithms are executed alternatively to update the position of global best solution until the stopping criterion is not met. In each execution cycle, the position of particles in PSO is initialized with the personal best solutions of the dragonflies. Results show a relatively faster convergence rate of the proposed algorithm and its relative superiority in finding an optimal solution. Similar methodology has been adopted in [36] and [37] for optimal power flow problem and unit commitment problem, respectively. Authors in [38] proposed a binary DA and binary PSO based hybrid optimization algorithm for the feature selection problem. In the proposed algorithm, the position of search agents and their direction and magnitude of movement are updated alternatively using DA and PSO until the stopping criteria is not met. There is a cyclic feed of updated magnitude and direction of movement of the search agents between DA and PSO. This leads to a synergic balance between exploration and exploitation of the search space throughout the search process. Authors in [39] proposed a DA and PSO based hybrid optimization algorithm for underwater image enhancement. It uses the implementation framework of DA and maintains the memory of personal best solution obtained by each dragonfly. In the presence of neighbouring dragonflies the step vector is deduced using the PSO equation for velocity update whereas in the absence of neighbouring dragonflies levy flight is used as in case of DA. This helps in maintaining an appropriate balance between exploration and exploitation of the search space throughout the search process. Results show relatively better performance of the proposed algorithm for the considered image enhancement problem.

In the above discussed research papers [25–35], it is observed that load balancing in cloud is generally achieved using VM/task migration which leads to slowdown and downtime. Further most of the frameworks either aim to balance load across active PMs or among their different considered resource capacities. The framework proposed in this paper aims to proactively balance the load across active PMs as well as among their considered resource capacities while allocating the resources.

Heuristic based algorithms are generally greedy in nature as they usually search for an immediate solution and hence do not guarantee an optimal solution especially in case of problems with large search space as in case of load balancing in cloud. Metaheuristics on the other hand iteratively tries to improve the initially obtained solution through exploration and exploitation of the search space using different intelligent approaches to find an optimal solution. The proposed framework, RAFL, uses a hybrid metaheuristic based optimization algorithm, PPSO-DA, to generate an optimal resource allocation plan for the given batch of tasks.

The previously proposed DA and PSO based hybrid optimization algorithms discussed in the related study typically use DA and PSO algorithms to run alternatively until the stopping criteria is not met [22,36–39]. This increases the time and space complexity of the algorithm. However, PPSO-DA follows a novel approach to use the characteristic of DA to distract the search agents outward the region of worst solution in the framework of PPSO. This helps in concentrating the search process in the promising regions of the search space to obtain an optimal solution.

3. System model and problem formulation

In this research work, single data center scenario is considered for cloud computing environment. The data center is modeled as a set $DC = \{PM_1, PM_2, PM_3, \ldots\}$ of heterogeneous PMs. A physical machine PM_i is characterized by the processing power of its CPU (PM_i^{CPU}) measured in million instructions per second (MIPS) and the memory size of its RAM (PM_i^{RAM}) measured in gigabyte (GB). A physical host machine PM_i can host multiple heterogeneous VMs represented as a set $VM_i = (VM_{i1}, VM_{i2}, \ldots, VM_{i|VM_i|})$. A virtual machine VM_{ij} on PM_i is characterized by the processing power of the CPU core (VM_{ij}^{CPU} in MIPS) and RAM (VM_{ij}^{RAM} in GB) allocated to it. The VMs hosted on a PM are executed in separate address spaces of the RAM. A dedicated VM is dynamically deployed for each task based on its resource requirement. The PMs which are currently hosting any number of VMs are called active PMs.

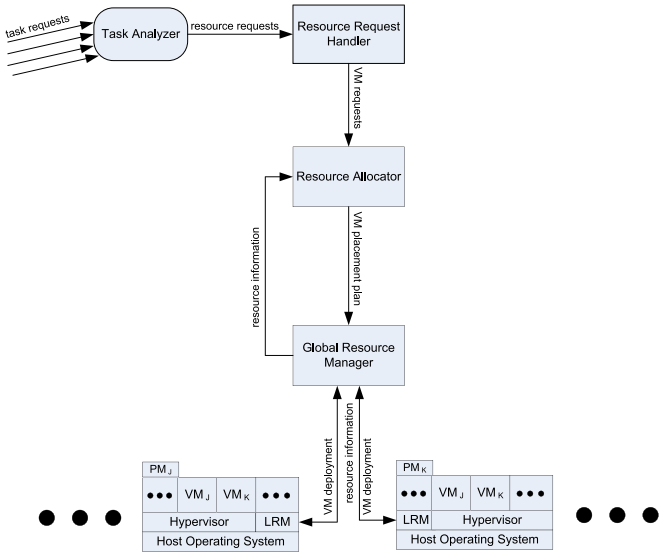


Fig. 1. Architecture of RAFL

A batch of independent and non-preemptive tasks $t_1, t_2, \ldots, t_{n_k}$ with heterogeneous resource requirements is generated to be scheduled in each scheduling cycle. A task t_j is characterized by its arrival time, deadline time, and length measured in million instructions (MI).

The architecture of the proposed framework, RAFL, is shown in Fig. 1. RAFL comprises of a dedicated local resource manager (LRM) for each PM, a global resource manager (GRM), a task analyzer (TA), a resource request handler (RRH), and a resource allocator (RA). The LRMs manage the computing resources of their respective PMs, and report the usage and availability of different resource capacities to the GRM. The GRM manages the overall resource usage and resource availability information of the data center, and report it to RA. When a batch of tasks arrives at the data center, TA analyzes the task requests to deduce the minimal resource capacities required to successfully execute the individual tasks within deadline. RRH uses the resource requirement information deduced by TA to decide the minimal VM configurations required to satisfy the resource requests. RA generates the resource allocation plan for the VM requests deduced by the RRH, simultaneously for different category of VMs on the basis of CPU and RAM configuration. The resource allocation logic of RA implements the proposed hybrid metaheuristic, PPSO-DA, to generate an optimal VM placement plan which aims to minimize load imbalance across active PMs and among their considered resource capacities. The VM placement plan is realized by GRM through the LRMs of corresponding PMs.

3.1. Objective function

In this paper, the load balancing problem is formulated to minimize load imbalance across active PMs and among their considered resource capacities using load imbalance factor (LIF) and resource capacity imbalance factor (RCIF), respectively.

Definition 1. LIF is the measure of load imbalance across active PMs of the data center. RAFL aims to minimize LIF in each scheduling cycle. It is calculated as

$$LIF = c_1 \times \frac{\sum_{\forall active\ PMs} \left| U^{CPU}_{APM_i} - Umean^{CPU}_{active\ PMs} \right|}{Umean^{CPU}_{active\ PMs}} + c_2 \times \frac{\sum_{\forall active\ PMs} \left| U^{RAM}_{APM_i} - Umean^{RAM}_{active\ PMs} \right|}{Umean^{RAM}_{active\ PMs}} \tag{1}$$

where APM_i denotes the i-th PM in the current set of active PMs; $U^{CPU}_{APM_i}$ and $U^{RAM}_{APM_i}$, respectively, are the percent CPU and RAM utilizations of APM_i; $Umean^{CPU}_{active\ PMs}$ and $Umean^{RAM}_{active\ PMs}$ are the percent mean CPU and mean RAM utilizations, respectively, of the active PMs; c_1 and c_2 are the weightage constants such that $c_1 + c_2 = 1$. In this paper, c_1 and c_2 have been assigned an equal value (i.e., c_1=c_2=0.5) to give an equal weightage to CPU and RAM.

Definition 2. RCIF is the measure of load imbalance among different considered resource capacities in active PMs. RAFL aims to minimize RCIF in each scheduling cycle. It is calculated as

$$RCIF = \sum_{\forall active\ PMs} \frac{\left| U^{CPU}_{APM_i} - U^{RAM}_{APM_i} \right|}{\frac{1}{2} \times \left(U^{CPU}_{APM_i} + U^{RAM}_{APM_i} \right)} \tag{2}$$

In this research work, a batch of m VMs are to be placed on the physical host machines in the set of n target PMs, such that the load

imbalance across active PMs and among their considered resource capacities is minimal. The VM placement problem under consideration is formulated as a minimization problem. The objective function is formulated using *LIF* and *RCIF* measures as

$$f = c_a * LIF + c_b * RCIF \tag{3}$$

where c_a and c_b are the weightage constants for load imbalance across active PMs and utilization imbalance among their resource capacities, respectively, such that $c_a + c_b = 1$. In this paper, c_a and c_b are assigned equal values (i.e., c_a=c_b=*0.5*) in order to equally prioritize the balancing of load across active PMs and balanced utilization of their considered resource capacities. Let P be the set of target PMs and Q be the set of VMs to be placed. Let us consider a binary variable x_{ij} as

$$x_{ij} = \begin{cases} 1 \text{ if } VM_j \text{is assigned to } PM_i \\ 0 \text{ otherwise} \end{cases} \tag{4}$$

The VM placement problem considered in this research work is subject to the following equality and inequality constraints:

$$\sum_{i=1}^{n} x_{ij} = 1 \ \forall VM_j \in Q \tag{5}$$

$$\sum_{j=1}^{m} VM_j^{CPU} x_{ij} \leq PM_i^{CPU} \times \left(\frac{T_{PM_i}^{CPU} - U_{PM_i}^{CPU}}{100} \right) \forall PM_i \in P \tag{6}$$

$$\sum_{j=1}^{m} VM_j^{RAM} x_{ij} \leq PM_i^{RAM} \times \left(\frac{T_{PM_i}^{RAM} - U_{PM_i}^{RAM}}{100} \right) \forall PM_i \in P \tag{7}$$

where $T_{PM_i}^{CPU}$ and $T_{PM_i}^{RAM}$ denote the maximum percent utilization thresholds of CPU and RAM, respectively, for PM_i. Equality constraint in Eq. (5) ensures that the resources are allocated only once for a VM. The inequality constraints in Eqs. (6) and (7), respectively, ensure that the CPU and RAM requirement of the VMs to be assigned to a PM do not exceed its remaining CPU and RAM capacities.

4. Preliminary optimization algorithms

Optimization algorithms typically follow two main strategies, i.e., exploration and exploitation of the search space to find an optimal solution. In exploration different regions of the search space are explored for the solutions, whereas in exploitation only the specific regions of the search space are searched. A proper balance and transition between the exploration and exploitation phases is very crucial to the performance of an optimization algorithm. Different algorithm specific parameters are used to control the exploration and exploitation of the search space by an optimization algorithm throughout the search process. An optimal adjustment of parameters is cumbersome yet very significant in the working of optimization algorithms.

4.1. Phasor PSO

PPSO is a trigonometric variant of PSO [40–43]. Experiment results in [19] show that PPSO outperforms various prominent variants of PSO in finding global optimal solution. The implementation framework of PPSO is similar to that of PSO. The search agents (i.e., particles) are used to explore and exploit the search space to find an optimal solution for the problem under consideration. The position of a particle in the search space represents a possible solution to the given problem. PPSO keeps track of the best solutions found by different particles in the swarm as the search process progresses. The movement of a particle in PPSO is influenced by the best position found so far by the corresponding particle and the overall best position found so far by any particle in the swarm. In the original version of PPSO algorithm, the value of inertia weight is set to zero, i.e., previous velocity of the particle do not impact its current velocity. Equations to update the position and velocity vectors of the i-th particle in iteration *itr* of PPSO algorithm are as follows:

$$\overrightarrow{X_i^{itr+1}} = \overrightarrow{X_i^{itr}} + \overrightarrow{V_i^{itr+1}} \tag{8}$$

$$\overrightarrow{V_i^{itr+1}} = P\left(\theta_i^{itr}\right)\left(\overrightarrow{pbest_i^{itr}} - \overrightarrow{X_i^{itr}}\right) + G\left(\theta_i^{itr}\right)\left(\overrightarrow{gbest^{itr}} - \overrightarrow{X_i^{itr}}\right) \tag{9}$$

where $\overrightarrow{X_i^{itr}}$ and $\overrightarrow{V_i^{itr}}$, respectively, are the position and velocity vectors of the i-th particle in the iteration *itr*, $P(\theta_i^{itr})$ and $G(\theta_i^{itr})$, respectively, are the trigonometric acceleration coefficients to decide the magnitude of i-th particle's velocity vector towards personal best and global best solutions, $\overrightarrow{pbest_i^{itr}}$ is the position vector of the best solution found until iteration *itr* by the i-th particle, and $\overrightarrow{gbest^{itr}}$ is the position vector of the overall best solution found until iteration *itr* by any particle of the swarm.

$$P\left(\theta_i^{itr}\right) = \left|\cos\left(\theta_i^{itr}\right)\right|^{2\sin\left(\theta_i^{itr}\right)} \tag{10}$$

$$G\left(\theta_i^{itr}\right) = \left|\sin\left(\theta_i^{itr}\right)\right|^{2\cos\left(\theta_i^{itr}\right)} \tag{11}$$

The velocity is subjected to the following constraint:

$$V_i^{itr+1} \leq \left|\cos\left(\theta_i^{itr}\right)\right|^2 (X_{max} - X_{min}) \tag{12}$$

where X_{max} and X_{min}, respectively, denote the maximum and minimum boundary values of the search space. The phase angle (θ_i^{itr}) is the only independent parameter in PPSO. Initially, it is generated uniformly at random in (0, 2π) for different search agents, and then it is updated in successive iterations for every particle according to following equations:

$$\theta_i^{t=0} = Z_i 2\pi \tag{13}$$

$$\theta_i^{itr+1} = \theta_i^{itr} + \left|\sin\left(\theta_i^{itr}\right) + \cos\left(\theta_i^{itr}\right)\right| \times 2\pi \tag{14}$$

where $\theta_i^{t=0}$ and θ_i^{itr}, respectively, are the phase angles of i-th particle at time step $t = 0$ and in the iteration itr, and Z_i is a random number in (0, 1).

The rise and fall of $P(\theta_i^{itr})$ and $G(\theta_i^{itr})$ in the similar or opposite directions with similar or different rates help avoid premature convergence. This behavior of $P(\theta_i^{itr})$ and $G(\theta_i^{itr})$ balances the exploration and exploitation of the search space throughout the search process and make PPSO a stochastically sound algorithm.

4.2. Dragonfly algorithm

DA [20] is inspired from the swarming behavior of dragonflies. The behavior of dragonflies is depicted through five primary parameters in DA: separation, alignment, cohesion, attraction towards food source, and distraction away from an enemy.

Separation parameter is the measure of separation of a dragonfly from other dragonflies in its neighborhood. The measure of separation along k-th dimension for i-th dragonfly in the swarm is calculated as

$$s_{ik} = -\sum_{j=1}^{n} x_{ik} - x_{jk} \tag{15}$$

where x_{ik} is the k-th dimension of the position vector of i-th dragonfly in the swarm, x_{jk} is the k-th dimension of the position vector of j-th dragonfly in the neighborhood of i-th dragonfly, and n is the total number of dragonflies in the neighborhood of i-th dragonfly.

Alignment parameter measures the mean velocity of dragonflies in the neighborhood of a dragonfly. The alignment of i-th dragonfly of the swarm along k-th dimension is calculated as

$$a_{ik} = \frac{\sum_{j=1}^{n} v_{jk}}{n} \tag{16}$$

where v_{jk} is the k-th dimension of the velocity vector of j-th dragonfly in the neighborhood of i-th dragonfly.

Cohesion parameter measures the cohesiveness of a dragonfly with its neighboring dragonflies. The cohesion of i-th dragonfly along its k-th dimension is calculated as

$$c_{ik} = \frac{\sum_{j=1}^{n} x_{jk}}{n} - x_{ik} \tag{17}$$

Survival is the primary objective of any swarm in nature. Hence, different individuals of a swarm must work in a collaborative manner to search food sources and stay away from the enemies. In DA, the measures of attraction towards a food source and distraction outward the region of an enemy for the i-th dragonfly along k-th dimension are calculated as follows:

$$f_{ik} = F_k - x_{ik} \tag{18}$$

$$e_{ik} = E_k + x_{ik} \tag{19}$$

where f_{ik} and e_{ik} are the measures of attraction towards a food source and distraction outward an enemy region, respectively, along the k-th dimension of i-th dragonfly. F_k and E_k, respectively, are the k-th dimensions of the position vectors of food source ($\overrightarrow{F}$) and enemy ($\overrightarrow{E}$).

The implementation framework of DA is based on the framework of PSO. It uses a step vector ($\overrightarrow{\Delta X}$) analogous to velocity vector in PSO to update the position vectors of dragonflies. The position of a dragonfly in the search space represents a possible solution to the given problem. The k-th dimension of the step vector for the i-th dragonfly is calculated as

$$\Delta X_{ik}^{itr+1} = \omega \Delta X_{ik}^{itr} + \mu_s s_{ik}^{itr} + \mu_a a_{ik}^{itr} + \mu_c c_{ik}^{itr} + \mu_f f_{ik}^{itr} + \mu_e e_{ik}^{itr} \tag{20}$$

where ΔX_{ik}^{itr}, s_{ik}^{itr}, a_{ik}^{itr} and c_{ik}^{itr}, respectively, are the step vector, separation, alignment and cohesion of the i-th dragonfly along k-th

dimension in the iteration *itr*; f_{ik}^{itr} and e_{ik}^{itr}, respectively, are the measures of attraction towards a food source and distraction outward an enemy region for *i*-th dragonfly along its *k*-th dimension, in the iteration *itr*. The constants in Eq. (20), i.e., ω, μ_s, μ_a, μ_c, μ_f, and μ_e, respectively, are the inertia weight, separation weight, alignment weight, cohesion weight, food factor, and enemy factor. These constants are tuned adaptively for transition between exploration and exploitation of the search space by the dragonflies.

DA unlike PSO does not keep track of the personal best solutions found by different dragonflies in the swarm. In DA, a dragonfly primarily moves under the influence of best and worst positions found by the swarm, and the velocity and position of dragonflies in its neighborhood. To decide the neighborhood of a dragonfly, a radius parameter is associated with each dragonfly. The radius is gradually increased proportional to the number of iterations as the optimization process progresses in order to make a gradual transition from exploration to exploitation of the search space. The position vector of a dragonfly with one or more dragonflies in its neighborhood is updated as

$$\overrightarrow{X_i^{itr+1}} = \overrightarrow{X_i^{itr}} + \overrightarrow{\Delta X_i^{itr+1}} = \left[x_{i1}^{itr} + \Delta x_{i1}^{itr+1},\ x_{i2}^{itr} + \Delta x_{i2}^{itr+1},\ \ldots,\ x_{id}^{itr} + \Delta x_{id}^{itr+1}\right] \tag{21}$$

In case if there are no dragonflies in the neighborhood of a dragonfly, its position vector is updated using a random walk (*Lévy flight*) as given in Eq. (22) in order to improve its explorative behavior.

$$\overrightarrow{X_i^{itr+1}} = \overrightarrow{X_i^{itr}} + L(d) \times \overrightarrow{X_i^{itr}} \tag{22}$$

where *d* is the dimensionality of the position vector and $L(d)$ is the *Lévy* flight.

5. The proposed hybrid optimization algorithm

The implementation framework of the proposed hybrid optimization algorithm, PPSO-DA, is primarily based on the framework of PPSO [19]. The search agents (i.e., particles in PPSO-DA) are attracted towards the promising regions of the search space using personal and social learning parameters, as in PPSO. PPSO-DA incorporates the feature of DA to distract the search agents outward the non-promising regions of search space in order to concentrate the search agents in the exploration and exploitation of the promising regions to find an optimal solution. The position vector of a search agent in the search space represents a possible solution to the problem under consideration.

In this paper, the considered VM placement problem is formulated as a minimization optimization problem. PPSO-DA is used to generate an optimal VM placement plan for the given set of heterogeneous task requests using intelligent and cooperative behavior of the swarm. A swarm in PPSO-DA is represented as follows:

$$\psi = \begin{bmatrix} \overrightarrow{X_1} \\ \overrightarrow{X_2} \\ \vdots \\ \overrightarrow{X_i} \\ \vdots \\ \overrightarrow{X_n} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1k} & \ldots & x_{1d} \\ x_{21} & x_{22} & \ldots & x_{2k} & \ldots & x_{2d} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i1} & x_{i2} & \ldots & x_{ik} & \ldots & x_{id} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nk} & \ldots & x_{nd} \end{bmatrix} \tag{23}$$

where ψ denotes the swarm, *n* is the population of swarm, $\overrightarrow{X_i}$ is the position vector of *i*-th particle in the swarm, x_{ik} is the *k*-th dimension of the *i*-th particle, and *d* is the dimensionality of a particle. The position vector of a particle in this research work represents a possible VM placement plan for the given batch of task requests where different dimensions correspond to different task requests. In case any constraint specified in Eqs. (5)-(7) is violated along a dimension, the neighborhood space corresponding to that particular dimension in the search space is searched. The objective function, *f* is computed for different particles in order to measure the appropriateness of corresponding solutions.

In PPSO-DA, the position of different particles is initialized randomly in the search range as follows:

$$x_{ik}^{t=0} = \lfloor Z * L + 0.5 \rfloor \tag{24}$$

where $x_{ik}^{t=0}$ represents the randomly initialized *k*-th dimension of *i*-th particle at time step $t = 0$, *Z* is a random number in (0, 1), and *L* is the length of target host list. The velocity corresponding to different dimensions of a particle is initialized in the range [$-\alpha_0 L$, $\alpha_0 L$] where $\alpha_0 \in [0, 1]$. At time step $t = 0$, particles are at their initial positions in the search space, i.e., only one solution is known to each particle. Hence, at $t = 0$ the position of a particle ($\overrightarrow{X_i^{t=0}}$) is also the best position known to particle (i.e., personal best position of the particle) and is represented as $\overrightarrow{pbest_i^{t=0}}$ for the *i*-th particle. The global best and global worst solutions at $t = 0$ are obtained as follows:

$$f_{gb}^{t=0} = min\left(f\left(X_i^{t=0}\right)\right)\ (i = 1,\ 2,\ 3,\ \ldots,\ n) \tag{25}$$

$$f_{gw}^{t=0} = max\left(f\left(X_i^{t=0}\right)\right)\ (i = 1,\ 2,\ 3,\ \ldots,\ n) \tag{26}$$

where $f_{gb}^{t=0}$ and $f_{gw}^{t=0}$, respectively, are the objective function values corresponding to the global best ($\overrightarrow{gbest^{t=0}}$) and global worst ($\overrightarrow{gworst^{t=0}}$) solutions known to the swarm, at $t = 0$. After time step $t = 0$, the location of personal best, global best, and global worst solutions is updated as follows:

$$pbest_i = \begin{cases} pbest_i, \ if\ f(X_i) \geq f(pbest_i) \\ X_i, \ if\ f(X_i) < f(pbest_i) \end{cases} \tag{27}$$

$$gbest = \begin{cases} gbest, \ if\ f(X_i) > f(gbest) \\ X_i, \ if\ f(X_i) < f(gbest) \end{cases} \tag{28}$$

$$gworst = \begin{cases} gworst, \ if\ f(X_i) < f(gworst) \\ X_i, \ if\ f(X_i) > f(gworst) \end{cases} \tag{29}$$

where $pbest_i$, $gbest$, and $gworst$, respectively, represents personal best solution found by the i-th particle, global best solution, and global worst solution. The VM placement problem considered in this paper obtains multiple global best and global worst solutions as the optimization process progresses. Two repositories are maintained to separately store the best and the worst solutions. A repository is updated under two scenarios in different manners. First, if fitness value of the current solution is equal to that of the solutions in a repository, then the current solution is added to the corresponding repository. Second, if the current solution dominates the solutions in a repository with respect to fitness value, then all the solutions currently in the repository are removed and the current solution is added to the repository. When required, the best and the worst solutions are chosen randomly from their respective repositories.

The particles move around the search space using their personal and social learning behaviors in order to explore and exploit different regions of the search space to find an optimal solution. The position and velocity vectors of different particles are updated in each iteration as follows:

$$\overrightarrow{X_i^{itr+1}} = \overrightarrow{X_i^{itr}} + \overrightarrow{V_i^{itr+1}} \tag{30}$$

$$\overrightarrow{V_i^{itr+1}} = P(\theta_i^{itr}) \times \left(\overrightarrow{pbest_i^{itr}} - \overrightarrow{X_i^{itr}}\right) + G(\theta_i^{itr}) \times \left(\overrightarrow{gbest^{itr}} - \overrightarrow{X_i^{itr}}\right) + sign'\left(\overrightarrow{gworst^{itr}} - \overrightarrow{X_i^{itr}}\right) \times \delta(itr) \times D\left(\overrightarrow{gworst^{itr}} - \overrightarrow{X_i^{itr}}\right) \tag{31}$$

where $\overrightarrow{X_i^{itr}}$ and $\overrightarrow{V_i^{itr}}$, respectively, are the position and velocity vectors of the i-th particle in the iteration itr, $\overrightarrow{pbest_i^{itr}}$ is the position vector of the best solution found until iteration itr by the i-th particle, $\overrightarrow{gbest^{itr}}$ is the position vector of the global best solution, $\overrightarrow{gworst^{itr}}$ is the position vector of the global worst solution, and θ_i^{itr} is the phase angle for the i-th particle in the iteration itr. $P(\theta_i^{itr})$ and $G(\theta_i^{itr})$, respectively, are the personal and social learning coefficients, $sign'(\overrightarrow{gworst^{itr}} - \overrightarrow{X_i^{itr}})$ gives the direction of distraction, $\delta(itr)$ is the distraction coefficient to systematically control the magnitude of distraction, and $D(\overrightarrow{gworst^{itr}} - \overrightarrow{X_i^{itr}})$ gives the magnitude of distraction outwards a global worst location, for the i-th particle in the iteration itr. The initialization and updating procedures for θ_i^{itr}, $P(\theta_i^{itr})$ and $G(\theta_i^{itr})$ are adopted as in the original PPSO algorithm [19]. Distraction in this work is formulated analogous to the phenomenon of magnetic repulsion, using Eqs. (32)-(34). The magnitude of distraction experienced by a particle is inversely proportional to its distance from the location of worst solution.

$$sign'\left(gworst_k^{itr} - x_{ik}^{itr}\right) = \begin{cases} -1, \ if\ gworst_k^{itr} - x_{ik}^{itr} > 0 \\ 1, \ if\ gworst_k^{itr} - x_{ik}^{itr} < 0 \\ sign'\left(gworst_k^{itr} - gbest_k^{itr}\right), \ if\ gworst_k^{itr} - x_{ik}^{itr} = 0 \end{cases} \tag{32}$$

$$\delta(itr) = S(4) - S\left(\left|4 - 8\left(\frac{itr}{\alpha_1 \times MAX\_ITR}\right)\right|\right) \tag{33}$$

$$D\left(gworst_k^{itr} - x_{ik}^{itr}\right) = \frac{\alpha_2 L}{(S(4) - S(0))} \times \left(S(4) - S\left(\frac{4}{L} \times \left|gworst_k^{itr} - x_{ik}^{itr}\right|\right)\right) \tag{34}$$

where α_1 and α_2 are the real-valued coefficients in the range (0, 1], used to control the magnitude of resulting distraction, MAX_ITR is the maximum number of iterations, and S denotes the sigmoid function whose value is calculated as

$$S(x) = \frac{1}{1 + e^{-x}} \tag{35}$$

where e is the Euler's number.

The stepwise procedure of PPSO-DA to find an optimal solution for the given problem is presented in Algorithm 1.

Algorithm 1. *PPSO-DA for resource allocation*

Input: Population of swarm: n, Maximum number of iterations: MAX_ITR, Number of task requests: n_k, Target host list size: L.
Output: VM placement plan for the given batch of tasks.
Initialize the population of swarm using Eq. (24).
Initialize the phase angle, θ for all the particles of swarm using Eq. (13).
Check the feasibility of generated solutions using Eqs. (5)-(7) and in case of any constraint violation search the neighborhood positions in the search space for feasible solutions.
While stopping criteria is not met do
For all particles do
Evaluate the corresponding solution using Eqs. (1)-(3).
Update the personal best, global best and global worst solutions using Eqs. (27)-(29).
End for
For all particles do
Update position vector using Eqs. (30)-(35).
Update phase angle using Eq. (14).
Repeat step 3.
End for
End while
Generate the VM placement plan using global best solution.

The PPSO-DA algorithm is comprised of two main phases, i.e., population initialization and iterative position update. In population initialization phase, position vector, velocity vector, and phase angle, θ are initialized for each particle in the swarm. The time-complexity of this phase depends on population size (n) and dimensionality of the particles. In the considered VM placement problem, dimensionality of particles is equal to the number of tasks (n_k) in the given batch of task requests. Hence, the time-complexity of population initialization phase is $O(n*n_k)$. The iterative position update phase includes position update, velocity update, phase angle update, and fitness evaluation of each particle in the successive iterations. The time-complexity of iterative position update phase depends on population size, dimensionality of particles, and the number of iterations used to find an optimal solution. The worst case time-complexity of this phase is $O(MAX\_ITR*n*n_k)$.

6. Simulation results and discussion

Experiments are performed using CloudSim simulator [44] to compare the performance of the proposed hybrid optimization algorithm, PPSO-DA with PPSO, DA, CLPSO, MHDA, SCA, and EHO in finding an optimal solution for the VM placement problem considered in the proposed resource allocation framework, RAFL. The related algorithms, i.e., PPSO and DA are discussed thoroughly in Sections 4.1 and 4.2, respectively, and the algorithm MHDA is briefly discussed in Section 2. A brief description of other compared algorithms, i.e., CLPSO, SCA, and EHO is as follows:

- In CLPSO, unlike PSO different dimensions of a particle may learn from the corresponding dimensions of the personal best solutions of different particles in the swarm. This prevents the particles from getting trapped in the local optima. A particle keeps on learning from the same computed position vector until the solution does not improve for the given number of iterations known as refreshing gap. In this paper, a refreshing gap of 7 is selected on the basis of results obtained in [21].
- In SCA, the search agents move under the influence of the best position found so far in the search space. The transition between the phases of exploration and exploitation is orchestrated using sine/cosine functions.
- In EHO, different search agents are assigned to different groups on the basis of their fitness. The movement of the search agents in a group is influenced by the position of the fittest search agent in the group. This leads to distributed exploitation of the search space. The movement of the best search agent in a group is influenced by the centroid position of all the search agents in the respective group. The search agent with worst fitness value in each group is moved to a random location in the search space to improve the stochastic behavior of the algorithm. In the current implementation, the number of groups is 4 and each group comprises of 5 members. The scale factors for the movement of group members towards the fittest member of the group and of the fittest members towards the centroid of their groups are taken as 0.5 and 0.1, respectively, as given in [24].

The values of control parameters for PPSO, DA, CLPSO, MHDA, SCA, and EHO are adopted from the papers [19,20,21,22,23] and [24], respectively. The control parameters for PPSO-DA are selected as discussed in Section 5. The population of search agents and maximum number of iterations considered for experiments are 20 and 200, respectively, for all the algorithms.

The datacenter is assumed to be comprised of 200 PMs of heterogeneous configurations with CPU cores of 250, 500, and 750 MIPS. Each VM is allocated a dedicated CPU core. This study considers three different configurations of PMs and six different configurations of VMs for experiments, as given in Tables 1 and 2, respectively. A VM configuration constitute of the processing power of CPU core allocated to it and the maximum amount of host machine's RAM the virtual machine is allowed to use for execution. Experiments are conducted for four set of tasks, comprising 5×10^3, 1×10^4, 2×10^4, and 3×10^4 number of tasks with heterogeneous resource requirements. For all set of tasks, 30 independent simulation runs are executed to examine the effectiveness and robustness of the algorithms.

Table 1
PM configurations

CPU processing power (MIPS)	RAM (GB)
2000	2
2500	2.5
3000	3

Table 2
VM configurations

Processing power (MIPS)	RAM (GB)
250	{0.25, 0.5}
500	{0.25, 0.5}
750	{0.5, 0.75}

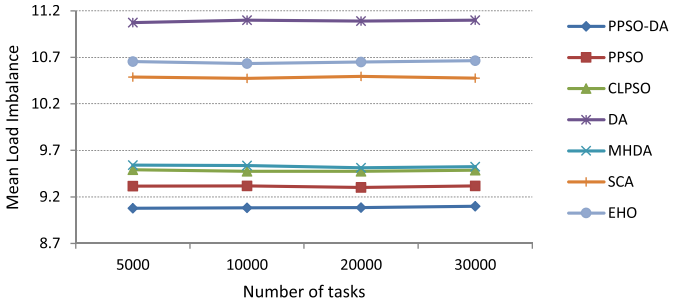


Fig. 2. Mean L_i comparison of the algorithms for different number of tasks

In RAFL, tasks are assumed to arrive in batches following poisson distribution with mean interval time of 100s. The number of tasks in a batch and the length of tasks follows uniform distribution in the range [20–30] and $[1 \times 10^5, 2 \times 10^5]$MI, respectively. A dedicated VM is allotted to each task. The mean CPU and memory usage of a VM depends on the nature of workload it is executing [45–47]. In this paper, the mean CPU and memory usage of a task corresponds to the minimal VM configuration required to successfully execute the task. A set of target PMs is deduced in each scheduling cycle to include the available CPU cores two times the tasks' requirement on the PMs with an appropriate availability of RAM.

The performance of the considered algorithms is evaluated on the basis of following metrics:

- *Load imbalance* metric, L_i measures the load imbalance across active PMs in a scheduling cycle. It is calculated as

$$L_i = \frac{1}{n_a} \times \left(c_3 \times \sum_{\forall active\ PMs} \left| U^{CPU}_{APM_i} - Umean^{CPU}_{active\ PMs} \right| + c_4 \times \sum_{\forall active\ PMs} \left| U^{RAM}_{APM_i} - Umean^{RAM}_{active\ PMs} \right| \right) \tag{36}$$

where n_a is the current total number of active PMs; c_3 and c_4 are the weightage constants such that $c_3 + c_4 = 1$. The mean of L_i metric corresponding to different scheduling cycles in a simulation run is called *mean load imbalance* (*mean* L_i).

- *Resource capacity imbalance* metric, RC_i measures the utilization imbalance among different considered resource capacities of active PMs in a scheduling cycle. It is calculated as

$$RC_i = \frac{\sum_{\forall active\ PMs} \left| U^{CPU}_{APM_i} - U^{RAM}_{APM_i} \right|}{n_a} \tag{37}$$

The mean of RC_i metric corresponding to different scheduling cycles in a simulation run is called *mean resource capacity imbalance* (*mean* RC_i).

6.1. Mean load imbalance

Mean load imbalance is the mean of L_i metric calculated over number of scheduling cycles required to execute the given number of tasks in a simulation run. The mean load imbalance metric comparison of different considered algorithms over 30 independent simulation runs for different number of tasks is graphically presented in Fig. 2. Results show that PPSO-DA consistently achieves the

Table 3
Mean L_i comparison of the algorithms for different number of tasks

Number of tasks	PPSO-DA	PPSO	CLPSO	DA	MHDA	SCA	EHO
5000	**9.077**	9.316	9.492	11.073	9.542	10.487	10.655
10000	**9.082**	9.317	9.475	11.100	9.537	10.474	10.633
20000	**9.085**	9.301	9.475	11.091	9.514	10.495	10.651
30000	**9.098**	9.319	9.486	11.099	9.526	10.476	10.665

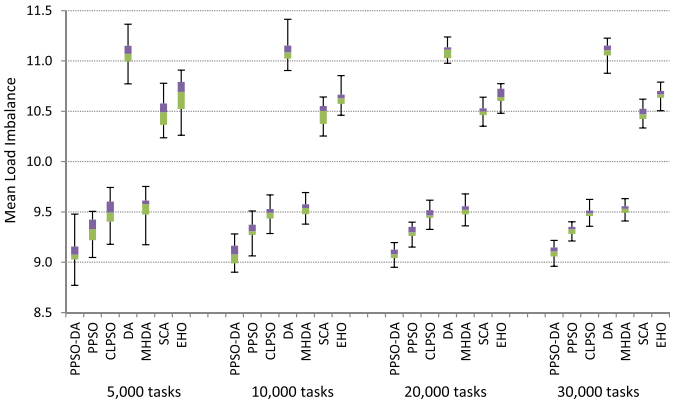


Fig. 3. Box plot of mean load imbalance over 30 independent simulation runs

Table 4
Mean L_i best value comparison of the algorithms for different number of tasks

Number of tasks	PPSO-DA	PPSO	CLPSO	DA	MHDA	SCA	EHO
5000	**8.770**	9.047	9.179	10.772	9.174	10.237	10.262
10000	**8.901**	9.064	9.286	10.905	9.379	10.253	10.461
20000	**8.951**	9.151	9.326	10.977	9.362	10.351	10.481
30000	**8.960**	9.212	9.358	10.878	9.410	10.334	10.507

Table 5
Mean L_i worst value comparison of the algorithms for different number of tasks

Number of tasks	PPSO-DA	PPSO	CLPSO	DA	MHDA	SCA	EHO
5000	**9.478**	9.506	9.744	11.366	9.753	10.779	10.908
10000	**9.282**	9.510	9.670	11.414	9.694	10.643	10.855
20000	**9.196**	9.399	9.617	11.239	9.679	10.640	10.776
30000	**9.218**	9.403	9.624	11.227	9.633	10.621	10.791

best relative mean performance in balancing load across active PMs. This implies that PPSO-DA is a comparatively more effective and reliable algorithm for the considered VM placement problem in cloud computing environment. The numerical values of mean L_i metric corresponding to different algorithms for different number of tasks are given in Table 3. Results show that PPSO-DA has a relative mean improvement of 2.51%, 4.36%, 22.07%, 4.89%, 15.38% and 17.23% compared to PPSO, CLPSO, DA, MHDA, SCA, and EHO, respectively, in mean L_i metric.

As depicted from the box plot in Fig. 3, the PPSO-DA algorithm obtains the fittest best and least worst value compared to different considered algorithms over 30 independent simulation runs for different number of tasks. This shows the comparative potential of PPSO-DA for the considered VM placement problem. The numerical data of best and worst mean L_i values over 30 independent simulation runs for different number of tasks is given in Tables 4 and 5, respectively.

6.2. Mean resource capacity imbalance

Mean resource capacity imbalance is the mean of RC_i metric calculated over number of scheduling cycles required to execute the given number of tasks. The mean resource capacity imbalance metric comparison for different considered algorithms over 30 independent simulation runs for different number of tasks is graphically presented in Fig. 4. Results show that PPSO-DA outperforms other considered algorithms in balancing load among different considered resource capacities (i.e., CPU and RAM) in active PMs. This implies that PPSO-DA is a comparatively more effective and reliable algorithm for the considered VM placement problem in cloud computing environment. The numerical values of mean RC_i metric corresponding to different algorithms for different number of tasks

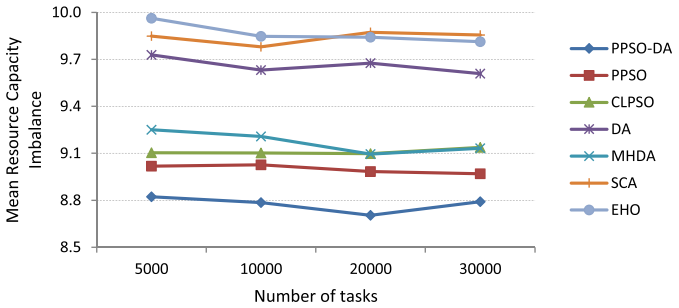


Fig. 4. Mean RC_i comparison of the algorithms for different number of tasks

Table 6
Mean RC_i comparison of the algorithms for different number of tasks

Number of tasks	PPSO-DA	PPSO	CLPSO	DA	MHDA	SCA	EHO
5000	**8.823**	9.018	9.104	9.728	9.250	9.849	9.962
10000	**8.785**	9.027	9.102	9.632	9.208	9.780	9.847
20000	**8.704**	8.984	9.098	9.676	9.095	9.872	9.841
30000	**8.791**	8.969	9.138	9.608	9.131	9.855	9.813

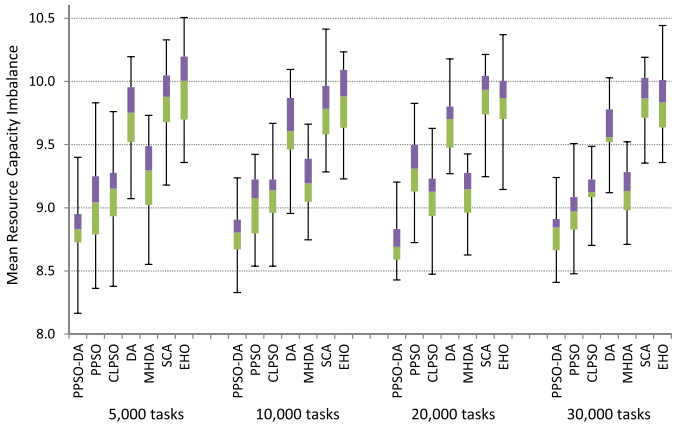


Fig. 5. Box plot of mean resource capacity imbalance over 30 independent simulation runs

Table 7
Mean RC_i best value comparison of the algorithms for different number of tasks

Number of tasks	PPSO-DA	PPSO	CLPSO	DA	MHDA	SCA	EHO
5000	**8.164**	8.362	8.378	9.072	8.552	9.180	9.358
10000	**8.330**	8.537	8.539	8.956	8.746	9.284	9.229
20000	**8.428**	8.724	8.474	9.270	8.627	9.247	9.145
30000	**8.410**	8.479	8.703	9.120	8.710	9.354	9.358

are given in Table 6. Results show that PPSO-DA has a relative mean improvement of 2.55%, 3.82%, 10.09%, 4.50%, 12.12%, and 12.42% compared to PPSO, CLPSO, DA, MHDA, SCA, and EHO, respectively, in mean RC_i metric.

As depicted from the box plot in Fig. 5, the PPSO-DA algorithm obtains the fittest best and least worst value compared to different considered algorithms over 30 independent simulation runs for different number of tasks. This shows the comparative potential of PPSO-DA for the considered VM placement problem. The numerical data of best and worst mean RC_i values over 30 independent simulation runs for different number of tasks is given in Tables 7 and 8, respectively.

6.3. Statistical analysis

In this section, Friedman and Wilcoxon sign rank tests have been performed for the comparative statistical analysis of the

Table 8
Mean RC_i worst value comparison of the algorithms for different number of tasks

Number of tasks	PPSO-DA	PPSO	CLPSO	DA	MHDA	SCA	EHO
5000	**9.400**	9.831	9.761	10.196	9.731	10.330	10.506
10000	**9.237**	9.423	9.669	10.095	9.662	10.415	10.235
20000	**9.203**	9.481	9.628	10.180	9.427	10.213	10.371
30000	**9.239**	9.508	9.487	10.029	9.523	10.192	10.442

Table 9
Relative ranks of algorithms in Friedman test corresponding to mean L_i and mean RC_i

Number of tasks	Evaluation metrics	PPSO-DA	PPSO	CLPSO	DA	MHDA	SCA	EHO
5000	Mean L_i	**1.07**	2.17	3.20	6.97	3.57	5.17	5.87
	Mean RC_i	**1.63**	2.57	2.73	5.67	3.43	5.77	6.20
10000	Mean L_i	**1.03**	2.17	3.20	7.00	3.60	5.20	5.80
	Mean RC_i	**1.53**	2.53	3.07	5.37	3.23	6.00	6.13
20000	Mean L_i	**1.00**	2.03	3.37	7.00	3.60	5.07	5.93
	Mean RC_i	**1.27**	2.67	3.20	5.57	3.07	6.10	6.13
30000	Mean L_i	**1.00**	2.03	3.27	7.00	3.70	5.00	6.00
	Mean RC_i	**1.67**	2.2	3.3	5.47	2.93	6.30	6.13

Table 10
p-values in Wilcoxon sign rank test corresponding to mean L_i and mean RC_i (PPSO-DA versus other compared algorithms)

Number of tasks	Evaluation metrics	PPSO	CLPSO	DA	MHDA	SCA	EHO
5000	Mean L_i	2.60×10^{-5}	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
	Mean RC_i	2.85×10^{-2}	1.03×10^{-3}	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
10000	Mean L_i	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
	Mean RC_i	3.16×10^{-3}	5.30×10^{-5}	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
20000	Mean L_i	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
	Mean RC_i	2.61×10^{-4}	2.40×10^{-5}	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
30000	Mean L_i	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
	Mean RC_i	2.56×10^{-2}	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$

Table 11
p-values in *t*-test corresponding to mean L_i and mean RC_i (PPSO-DA versus other compared algorithms)

Number of tasks	Evaluation metrics	PPSO	CLPSO	DA	MHDA	SCA	EHO
5000	Mean L_i	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
	Mean RC_i	2.11×10^{-2}	2.66×10^{-4}	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
10000	Mean L_i	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
	Mean RC_i	2.05×10^{-4}	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
20000	Mean L_i	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
	Mean RC_i	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
30000	Mean L_i	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
	Mean RC_i	8.14×10^{-3}	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$

Table 12
Average of best values obtained over 30 runs for the considered benchmarks

Function	PPSO	CLPSO	DA	MHDA	SCA	EHO	PPSO-DA
F1	**0.0**	2.074E-33	7.930E-58	**0.0**	**0.0**	2.142E-45	**0.0**
F2	**0.0**	7.691E-05	5.680E-18	5.710E-21	3.174E-62	3.593E-04	**0.0**
F3	**0.0**	3.083	1.443E-34	3.961E-57	**0.0**	2.083E-27	**0.0**
F4	7.694E-37	9.720E-03	4.568E-07	7.267E-14	1.409E-17	**2.084E-39**	1.008E-15
F5	2.944	1.246	**4.208E-02**	1.793	4.530	7.463	4.632
F6	1.793E-31	2.224E-27	1.466E-32	3.624E-27	**0.0**	7.214E-09	3.953E-21
F7	1.500E-04	2.348E-03	1.420E-05	7.301E-03	2.352E-04	4.438E-03	**2.019E-06**
F8	-2639.769	-3257.240	-8753.583	-6642.130	-7355.286	-6214.583	**-9005.031**
F9	**0.0**	8.341	3.892E-19	7.329E-14	3.122E-17	**0.0**	**0.0**
F10	7.820E-15	4.473E-14	3.227E-15	**4.913E-18**	8.957E-16	1.524E-11	3.561E-09
F11	1.639E-02	1.621E-07	**0.0**	**0.0**	4.640E-09	**0.0**	4.622E-11
F12	8.410E-03	2.486E-02	**0.0**	2.826E-11	1.732E-02	7.544E-11	4.129E-06

considered optimization algorithms. Friedman test has been used to rank the relative performance of the considered algorithms with respect to mean L_i and mean RC_i metrics. In Friedman test, lowest and highest ranks correspond to the relatively best and worst performers, respectively. The relative ranks of different algorithms deduced using Friedman test as given in Table 9 shows that PPSO-DA consistently achieves the best relative rank compared to other considered algorithms with respect to mean L_i and mean RC_i metrics for different number of tasks.

In this paper, non-parametric Wilcoxon sign rank test and *t*-test are performed to test whether the results obtained by PPSO-DA for mean L_i and mean RC_i metrics over 30 independent simulation runs differ significantly from the results obtained using other compared algorithms. A significance level of 0.05 has been used to test the following hypothesis. The null hypothesis, H_0 is: 'There is no significant difference between the results of two algorithms' and the alternate hypothesis, H_a is: 'The difference between the results is significant'. If the computed p-value is less than the level of significance then we reject null hypothesis, H_0 and accept the alternate hypothesis, H_a. The p-values computed in Wilcoxon sign rank test and *t*-test as given in Tables 10 and 11, respectively, are all less than the considered significance level of 0.05, which indicates that the difference between the results is significant.

Friedman test, wilcoxon sign rank test, and *t*-test statistically proves the superiority of PPSO-DA over other compared algorithms for the considered VM placement problem.

6.4. Benchmark testing

This section presents the benchmark testing of the compared optimization algorithms on twelve CEC'05 benchmark problems [48]. The specifications of considered benchmark functions are given in appendix A. The functions F1-F7 are unimodal functions, i.e., their solution space has only one optimal solution, whereas the functions F8-F12 are multimodal functions, i.e., their solution space has multiple local optimal solutions. The unimodal benchmark functions are considered suitable for testing the exploitation capability of an optimization algorithm, whereas multimodal benchmark functions are considered for testing the exploration capability. The number of dimensions, N_d considered for each benchmark function is 30. The compared algorithms are run 30 times for each benchmark function. The maximum number of function evaluations, *maxNFE* is considered as the termination criteria for each run. According to IEEE-CEC benchmark, $maxNFE = 10^4 \times N_d$. The mean of best solution values obtained over 30 independent runs is reported in Table 12, to present the effectiveness of compared algorithms for the considered benchmark problems.

The PPSO-DA algorithm performs best on four out of seven unimodal benchmark functions followed by SCA and PPSO which performs best on three unimodal functions. In multimodal benchmark testing, the DA, MHDA, EHO and PPSO-DA optimization algorithms give comparatively superior performance with best results on two out of five multimodal benchmark problems. The overall performance of PPSO-DA is best among the compared algorithms, as it performs best on six out of twelve considered benchmark problems.

7. Conclusion

In this paper, a hybrid metaheuristic based resource allocation framework, RAFL for proactive load balancing in cloud computing environment is presented. The framework aims to simultaneously balance load across active PMs and among their considered resource capacities. This is realized using a hybrid optimization algorithm, PPSO-DA proposed in this paper. PPSO-DA hybridizes PPSO and DA to explore and exploit the search space in an effective and efficient manner. PPSO-DA utilizes the self-learning and social-learning approaches of PPSO to attract the search agents towards promising regions of the search space, and the feature of DA to distract the search agents outward the region of worst solutions. RAFL uses PPSO-DA to generate an optimal VM placement plan with objective to minimize load imbalance across active PMs and among their considered resource capacities. Simulation results for the load balancing problem considered in this paper show that the proposed hybrid optimization algorithm, PPSO-DA is a comparatively robust, reliable, and scalable algorithm. The statistical analysis and benchmark testing also validates the relative superiority of PPSO-DA.

As future work, we plan to further extend this research work for solving other optimization problems in cloud computing, e.g., energy conservation and resource utilization. We also plan to implement this work using container-based approach.

Acknowledgements

We are extremely thankful to the editor and all the anonymous reviewers for their precious time and valuable suggestions. We acknowledge the Ph.D. institute fellowship received from Sant Longowal Institute of Engineering and Technology, India to carry out the research work.

Appendix A

See Tables A.1 and A.2 for unimodal and multimodal benchmark functions, respectively.

Table A.1
Unimodal benchmark functions

Function	Mathematical expression	N_d	Range	f_{min}
F1	$F_1(x) = \sum_{i=1}^{N_d} x_i^2$	30	[-100,100]	0
F2	$F_2(x) = \sum_{i=1}^{N_d} \lvert x_i \rvert + \prod_{i=1}^{N_d} \lvert x_i \rvert$	30	[-10,10]	0
F3	$F_3(x) = \sum_{i=1}^{N_d} (\sum_{j=1}^{i} x_j)^2$	30	[-100,100]	0
F4	$F_4(x) = max\{\lvert x_i \rvert; 1 \le i \le N_d\}$	30	[-100,100]	0
F5	$F_5(x) = \sum_{i=1}^{N_d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	0
F6	$F_6(x) = \sum_{i=1}^{N_d} (\lfloor x_i + 0.5 \rfloor)^2$	30	[-100,100]	0
F7	$F_7(x) = \sum_{i=1}^{N_d} i x_i^4 + random[0,1)$	30	[-1.28,1.28]	0

Table A.2
Multimodal benchmark functions

Function	Mathematical expression	N_d	Range	f_{min}
F8	$F_8(x) = \sum_{i=1}^{N_d} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[-500,500]	$-418.9829 \times N_d$
F9	$F_9(x) = \sum_{i=1}^{N_d} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[-5.12,5.12]	0
F10	$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{N_d}\sum_{i=1}^{N_d} x_i^2}\right) - \exp\left(\frac{1}{N_d}\sum_{i=1}^{N_d} cos(2\pi x_i)\right) + 20 + e$	30	[-32,32]	0
F11	$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{N_d} x_i^2 - \prod_{i=1}^{N_d} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600,600]	0
F12	$F_{12}(x) = \frac{\pi}{N_d}\{10\sin(\pi y_1) + \sum_{i=1}^{N_d-1}(y_i - 1)^2[1 + 10sin^{2(\pi y_{i+1})}] + (y_{N_d} - 1)^2\} + \sum_{i=1}^{N_d} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m; & x_i > a \\ 0; & -a < x_i < a \\ k(-x_i - a)^m; & x_i < -a \end{cases}$	30	[-50,50]	0

References

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generation computer systems 25 (6) (2009) 599–616, https://doi.org/10.1016/j.future.2008.12.001.
[2] Armbrust M., et al., "Above the clouds: A berkeley view of cloud computing," Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS, 28(13), 2009.
[3] P. Mell, T. Grance, The NIST definition of cloud computing, National Institute of Standards & Technology, Gaithersburg, MD, USA, SP 800-145, 2011.
[4] S. Vargaftik, I. Keslassy, A. Orda, LSQ: Load Balancing in Large-Scale Heterogeneous Systems with Multiple Dispatchers, IEEE/ACM Transactions on Networking 28 (3) (2020) 1186–1198, https://doi.org/10.1109/TNET.2020.2980061.
[5] D. Meisner, B.T. Gold, T.F. Wenisch, PowerNap: eliminating server idle power, ACM SIGARCH Computer Architecture News 37 (1) (2009) 205–216, https://doi.org/10.1145/2528521.1508269.
[6] W. Leinberger, G. Karypis, V. Kumar, Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints, in: *Proc. of 1999 International Conference on Parallel Processing*, Japan, September, 1999, pp. 404–412, https://doi.org/10.1109/ICPP.1999.797428.
[7] H.I. Christensen, A. Khan, S. Pokutta, P. Tetali, Approximation and online algorithms for multidimensional bin packing: A survey, Computer Science Review 24 (2017) 63–79, https://doi.org/10.1016/j.cosrev.2016.12.001.
[8] A. Thakur, M.S. Goraya, A taxonomic survey on load balancing in cloud, Journal of Network and Computer Applications 98 (2017) 43–57, https://doi.org/10.1016/j.jnca.2017.08.020.
[9] M. Xu, W. Tian, R. Buyya, A survey on load balancing algorithms for virtual machines placement in cloud computing, Concurrency and Computation: Practice and Experience 29 (12) (2017) 4123–4138, https://doi.org/10.1002/cpe.4123.
[10] P. Kumar, R. Kumar, Issues and challenges of load balancing techniques in cloud computing: A survey, ACM Computing Surveys 51 (6) (2019) 1–35, https://doi.org/10.1145/3281010.
[11] D.A. Shafiq, N.Z. Jhanjhi, A. Abdullah, Load balancing techniques in cloud computing environment: A review, Journal of King Saud University-Computer and Information Sciences (2021), https://doi.org/10.1016/j.jksuci.2021.02.007.
[12] S. Padhy, J. Chou, MIRAGE: A consolidation aware migration avoidance genetic job scheduling algorithm for virtualized data centers, Journal of Parallel and Distributed Computing 154 (2021) 106–118, https://doi.org/10.1016/j.jpdc.2021.03.004.
[13] N. Garg, D. Singh, M.S. Goraya, Energy and resource efficient workflow scheduling in a virtualized cloud environment, Cluster Computing 24 (2) (2021) 767–797, https://doi.org/10.1007/s10586-020-03149-4.
[14] T. Wood, P.J. Shenoy, A. Venkataramani, M.S. Yousif, Black-box and Gray-box Strategies for Virtual Machine Migration, in: *Proc. of the 4th USENIX Conference on Networked Systems Design & Implementation*, Cambridge, USA, April, 2007.
[15] Y. Wu, M. Zhao, Performance modeling of virtual machine live migration, in: *Proc. of 2011 IEEE 4th International Conference on Cloud Computing*, Washington, DC, USA, July, 2011, pp. 492–499, https://doi.org/10.1109/CLOUD.2011.109.
[16] M. Gabay, S. Zaourar, Vector bin packing with heterogeneous bins: application to the machine reassignment problem, Annals of Operations Research 242 (1) (2016) 161–194, https://doi.org/10.1007/s10479-015-1973-7.
[17] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Information sciences 237 (2013) 82–117, https://doi.org/10.1016/j.ins.2013.02.041.
[18] H. Singh, S. Tyagi, P. Kumar, S.S. Gill, R. Buyya, Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions, Simulation Modelling Practice and Theory 111 (2021), 102353, https://doi.org/10.1016/j.simpat.2021.102353.
[19] M. Ghasemi, E. Akbari, A. Rahimnejad, S.E. Razavi, S. Ghavidel, L. Li, Phasor particle swarm optimization: a simple and efficient variant of PSO, Soft Computing 23 (19) (2019) 9701–9718, https://doi.org/10.1007/s00500-018-3536-8.

[20] S. Mirjalili, Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, Neural Computing and Applications 27 (4) (2016) 1053–1073, https://doi.org/10.1007/s00521-015-1920-1.
[21] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE transactions on evolutionary computation 10 (3) (2006) 281–295, https://doi.org/10.1109/TEVC.2005.857610.
[22] S.R. KS, S. Murugan, Memory based hybrid dragonfly algorithm for numerical optimization problems, Expert Systems with Applications 83 (2017) 63–78, https://doi.org/10.1016/j.eswa.2017.04.033.
[23] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Knowledge-based systems 96 (2016) 120–133, https://doi.org/10.1016/j.knosys.2015.12.022.
[24] G.G. Wang, S. Deb, L.D.S. Coelho, Elephant herding optimization, in: *Proc. of 3rd International Symposium on Computational and Business Intelligence*, Indonesia, December, 2015, pp. 1–5, https://doi.org/10.1109/ISCBI.2015.8.
[25] M. Sheikhalishahi, R.M. Wallace, L. Grandinetti, J.L. Vazquez-Poletti, F. Guerriero, A multi-dimensional job scheduling, Future Generation Computer Systems 54 (2016) 123–131, https://doi.org/10.1016/j.future.2015.03.014.
[26] P. Silva, C. Perez, F. Desprez, Efficient heuristics for placing large-scale distributed applications on multiple clouds, in: Proc. of 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2016, pp. 483–492, https://doi.org/10.1109/CCGrid.2016.77.
[27] L. Zuo, S. Dong, L. Shu, C. Zhu, G. Han, A multiqueue interlacing peak scheduling method based on tasks' classification in cloud computing, IEEE Systems Journal 12 (2) (2016) 1518–1530, https://doi.org/10.1109/JSYST.2016.2542251.
[28] M.K. Gupta, A. Jain, T. Amgoth, Power and resource-aware virtual machine placement for IaaS cloud, Sustainable Computing: Informatics and Systems 19 (2018) 52–60, https://doi.org/10.1016/j.suscom.2018.07.001.
[29] S. Chhabra, A.K. Singh, Dynamic hierarchical load balancing model for cloud data centre networks, Electronics Letters 55 (2) (2019) 94–96, https://doi.org/10.1049/el.2018.5427.
[30] Y. Gao, H. Guan, Z. Qi, Y. Hou, L. Liu, A multi-objective ant colony system algorithm for virtual machine placement in cloud computing, Journal of computer and system sciences 79 (8) (2013) 1230–1242, https://doi.org/10.1016/j.jcss.2013.02.004.
[31] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, H. Tenhunen, Using ant colony system to consolidate VMs for green cloud computing, IEEE Transactions on Services Computing 8 (2) (2014) 187–198, https://doi.org/10.1109/TSC.2014.2382555.
[32] J. Kumar, A.K. Singh, A. Mohan, Resource-efficient load-balancing framework for cloud data center networks, ETRI Journal 43 (1) (2021) 53–63, https://doi.org/10.4218/etrij.2019-0294.
[33] W. Wei, K. Wang, K. Wang, H. Gu, H Shen, Multi-resource balance optimization for virtual machine placement in cloud data centers, Computers & Electrical Engineering 88 (2020), 106866, https://doi.org/10.1016/j.compeleceng.2020.106866.
[34] F. Ramezani, J. Lu, F.K. Hussain, Task-based system load balancing in cloud computing using particle swarm optimization, International journal of parallel programming 42 (5) (2014) 739–754, https://doi.org/10.1007/s10766-013-0275-4.
[35] J.P.B. Mapetu, Z. Chen, L. Kong, Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing, Applied Intelligence 49 (9) (2019) 3308–3330, https://doi.org/10.1007/s10489-019-01448-x.
[36] S. Khunkitti, A. Siritaratiwat, S. Premrudeepreechacharn, R. Chatthaworn, N.R. Watson, A hybrid DA-PSO optimization algorithm for multiobjective optimal power flow problems, Energies 11 (9) (2018) 2270, https://doi.org/10.3390/en11092270.
[37] S. Khunkitti, N.R. Watson, R. Chatthaworn, S. Premrudeepreechacharn, A. Siritaratiwat, An improved DA-PSO optimization approach for unit commitment problem, Energies 12 (12) (2019) 2335, https://doi.org/10.3390/en12122335.
[38] M.A. Tawhid, K.B. Dsouza, Hybrid binary dragonfly enhanced particle swarm optimization algorithm for solving feature selection problems, Mathematical Foundations of Computing 1 (2) (2018) 181–200, https://doi.org/10.3934/mfc.2018009.
[39] R. Prasath, T. Kumanan, Underwater image enhancement with optimal histogram using hybridized particle swarm and dragonfly, The Computer Journal (2021), https://doi.org/10.1093/comjnl/bxab056.
[40] J. Kennedy, R. Eberhart, Particle swarm optimization, in: *Proc. of ICNN'95-International Conference on Neural Networks*, November, 1995, pp. 1942–1948, https://doi.org/10.1109/ICNN.1995.488968.
[41] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, in: *Proc. of 1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*, October, 1997, pp. 4104–4108, https://doi.org/10.1109/ICSMC.1997.637339.
[42] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Proc. of 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence, May, 1998, pp. 69–73, https://doi.org/10.1109/ICEC.1998.699146.
[43] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE transactions on Evolutionary Computation 6 (1) (2002) 58–73, https://doi.org/10.1109/4235.985692.
[44] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A. De Rose, R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and experience 41 (1) (2011) 23–50, https://doi.org/10.1002/spe.995.
[45] Findeisen P., "Determining processor usage by a thread," U.S. Patent 7426731, September 16, 2008.
[46] C. Reiss, A. Tumanov, G.R. Ganger, R.H. Katz, M.A. Kozuch, Heterogeneity and dynamicity of clouds at scale: Google trace analysis, in: *Proc. of the 3rd ACM Symposium on Cloud Computing*, October, 2012, pp. 1–13, https://doi.org/10.1145/2391229.2391236.
[47] W. Song, Z. Xiao, Q. Chen, H. Luo, Adaptive resource provisioning for the cloud using online bin packing, IEEE Transactions on Computers 63 (11) (2013) 2647–2660, https://doi.org/10.1109/TC.2013.148.
[48] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, KanGAL report 2005005 (May 30, 2005).

Avnish Thakur received his B.E. degree in computer science and engineering from Sambhram Institute of Technology, Bangalore, India, in 2010 and his M.E. degree in software engineering from Thapar University, Patiala, India, in 2013. He is currently working toward Ph.D. degree in computer science at Sant Longowal Institute of Engineering and Technology, Longowal, India. His current research interests include load balancing in cloud and metaheuristic optimization algorithms.

Major Singh Goraya received his B.E. degree in computer science and engineering from Sant Longowal Institute of Engineering and Technology, Longowal, India, in 1997 and his master's and Ph.D. degrees in computer science and engineering from Punjabi University, Patiala, India, in 2003 and 2013, respectively. He is currently working as a Professor in the Department of Computer Science and Engineering at Sant Longowal Institute of Engineering and Technology, Longowal, India. His current research interests include fault tolerance, load balancing, energy conservation, and service mapping in cloud.