Energy Informatics

RESEARCH **Open Access**

Energy efficient resource management in data centers using imitation-based optimization

V. Dinesh Reddy[1,2*], G. Subrahmanya V. R. K. Rao[3] and Marco Aiello[2]

*Correspondence: dineshvemula@gmail.com

[1] Department of CSE, SRM University-AP, Amaravati 522502, Andhrapradesh, India
[2] Department of Service Computing, University of Stuttgart, 70569 Stuttgart, Germany
[3] mokSa.ai, Detroit, USA

Abstract

Cloud computing is the paradigm for delivering streaming content, office applications, software functions, computing power, storage, and more as services over the Internet. It offers elasticity and scalability to the service consumer and profit to the provider. The success of such a paradigm has resulted in a constant increase in the providers' infrastructure, most notably data centers. Data centers are energy-intensive installations that require power for the operation of the hardware and networking devices and their cooling. To serve cloud computing needs, the data center organizes work as virtual machines placed on physical servers. The policy chosen for the placement of virtual machines over servers is critical for managing the data center resources, and the variability of workloads needs to be considered. Inefficient placement leads to resource waste, excessive power consumption, and increased communication costs. In the present work, we address the virtual machine placement problem and propose an Imitation-Based Optimization (IBO) method inspired by human imitation for dynamic placement. To understand the implications of the proposed approach, we present a comparative analysis with state-of-the-art methods. The results show that, with the proposed IBO, the energy consumption decreases at an average of 7%, 10%, 11%, 28%, 17%, and 35% compared to Hybrid meta-heuristic, Extended particle swarm optimization, particle swarm optimization, Genetic Algorithm, Integer Linear Programming, and Hybrid Best-Fit, respectively. With growing workloads, the proposed approach can achieve monthly cost savings of €201.4 euro and CO_2 Savings of 460.92 lbs CO_2/month.

Keywords: Data center, Energy efficiency, Optimization, Resource scheduling, Imitation

Introduction

Cloud computing data centers (CDCs) are rapidly expanding in number and size to keep pace with the escalating demand for highly efficient computing and data storage solutions. The main operational cost of CDCs can be ascribed to power. In fact, according to estimates, high-end cloud servers will use around 3–13% of global electricity by 2030 compared to 1% of 2010 (Andrae and Edler 2015). According to Andrae et al., the forecasted electricity usage of data centers in 2030 is estimated at around 8000 TWh (Andrae and Edler 2015). The demand for data center (DC) power has risen from 1–2% in 2022 to 3%-4% in 2023. By 2030, the power demand in DCs will surge by 160% compared to 2023, as illustrated in Fig. 1 (Brian et al. 2024). The graph indicates that since 2023, the power consumption of AI-equipped DCs has significantly increased and is expected to continue growing until 2030, alongside DCs without AI (ex-AI). This rise in power demand is also anticipated to result in a more than 100% increase in CO_2 emissions from DCs, reaching approximately 215–220 million tons by 2030 compared to 2022. Despite AI's numerous benefits across various sectors and its crucial role in today's world, it is imperative to regulate its use to mitigate carbon emissions and environmental impacts. Reducing the energy consumption of a data center is crucial for cutting operational expenses and promoting sustainability.

Generally, the approaches for managing energy consumption in CDCs fall under four categories: host load prediction techniques, resource overcommitment, Virtual Machine (VM) placement, and workload consolidation. In the first approach, efficient prediction algorithms are employed to know the expected workload beforehand in order to schedule the jobs efficiently. The focus is to keep an appropriate number of physical machines (PMs) active while making the remaining servers sleep or switch to other low-power modes to save energy. The host load prediction is the prediction of CPU load, and these predictions are based on the CPU usage history. Host load prediction may go off target when the load fluctuates drastically at small timescales. The resource overcommitment scheme allocates VM resources to PMs over their actual capacities, trusting that most of the time, VMs do not fully utilize their allocated resources. However, such an approach may lead to PM overloading, where the requests from all the VMs assigned to a PM exceed the PM's capacity. Consequently, the performance of some or all VMs running on an overloaded PM degrades. Moreover, efficiency is highly dependent on the projections made by the operators. The goal of workload consolidation is efficient utilization of the resources, which can be achieved through live migration. Live migration transfers a VM among physical servers to reduce power consumption by improving utilization.

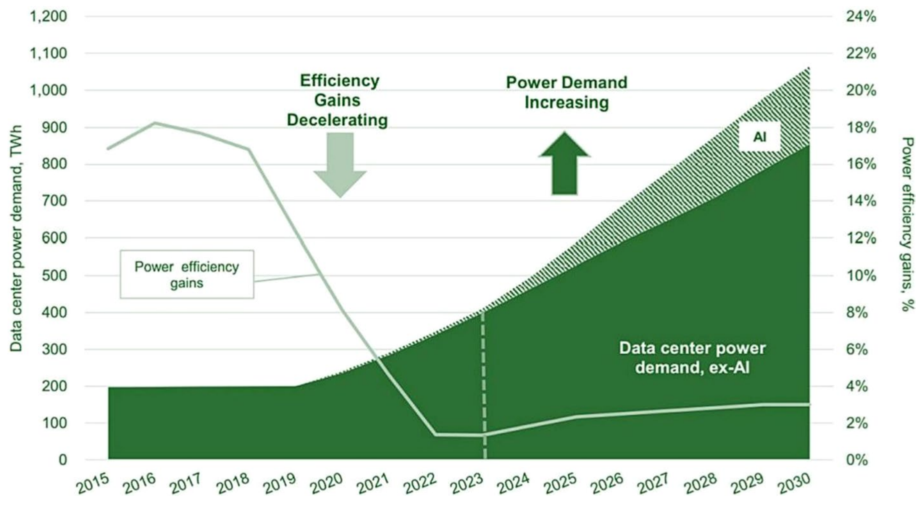

Fig. 1 Surge in data center power use in TWh(LHS) with an AI kicker on the way and power efficiency gains (RHS) (Brian et al. 2024)

VM placement is the process of mapping the most appropriate PMs in the data center to deploy a pool of newly created VMs.

A good VM placement technique has a direct impact on the performance and power consumption of cloud data centers. Our research focuses on exploiting virtualization capabilities and proposing a novel VM placement and workload consolidation technique aimed at decreasing the energy usage of the entire data center while fully conforming to a strict service level agreement (SLA) for the service users. An SLA is a commitment between a service provider and a client as agreed upon in the contract. The essence is to switch the idle PMs to low-power modes (i.e., hibernate) to eliminate the idle power consumption. Besides, the VMs can also be dynamically consolidated to the minimal number of PMs based on their current resource requirements using efficient live migration algorithms. VM consolidation is the solution to making utilization as high as possible and improving scalability, availability, and user demand.

The problem of VM placement has been the topic of several research efforts with heuristics, meta-heuristics, and hybrid approaches. Heuristics approaches like First-ft, Best-fit, First-fit decreasing and others have lower complexity but will not guarantee the optimal solution (Jangiti et al. 2020; Hobaei-Arani et al. 2017; Azizi et al. 2020; Beloglazov and Rajkumar 2012; Keller et al. 2012). Most of the meta-heuristics approaches have slow convergence rates, weakness in local search, and the possibility of getting stuck into local optima in the case of high-dimensional search spaces (Kumar and Raza 2015; Xu et al. 2020; Reddy et al. 2020a; Zhao et al. 2023; Javadi-Moghaddam and Zahra 2023). Further, these approaches involve many parameters and results will depend on how one tunes these parameters to yield optimal solutions. Although hybrid meta-heuristics have shown improvement in the performance, but they are very complex and time consuming (Reddy et al. 2020b; Vijaya and Srinivasan 2024; Parida et al. 2024; Abualigah and Alkhrabsheh 2022; Chen et al. 2019). Further, these algorithms fail to balance between exploration and exploitation. To solve these issues, by considering the migration overhead and possible SLA violations, we explore design methodologies aimed at minimizing energy consumption across the entire data center. We propose minimizing total energy consumption by using an imitation-based optimization (IBO) algorithm. IBO is developed based on the human imitation mechanism. The imitation behaviour of the individual solution in the IBO algorithm makes them move to better locations in the search space. IBO has a high ability to trade-off between the imitated movement (exploration) and sub-random sequence (exploitation) by choosing the best among the two. IBO minimizes the need for extensive parameter tuning, making it more accessible and practical for real-world applications without compromising on the quality of the VM placement. Further, the method is designed to maintain high performance even as the scale of the data center increases, ensuring efficient management of large and complex cloud environments.

The key contributions of the present work are:

- A systematic model for complex resource allocation problems in a heterogeneous cloud environment.
- The development of a novel optimization algorithm (IBO) based on imitation to handle the resource allocation problem and reduce energy consumption.

The remainder of the paper is organized as follows. In "Related works", a short review of VM placement solution from the literature is presented. In "Problem formalization", we illustrate the resource scheduling model for heterogeneous cloud computing. Then, a novel population-based optimization algorithm called IBO for solving the VM placement problem is presented in "VM allocation using imitation based optimization". In "Experimental results", we discuss the effects of our proposed scheme on reducing the energy consumption of the data center, followed by conclusions and future research directions in "Discussion and conclusion".

Related works

Cloud management encompasses a range of cloud operations. Recent research has focused on the scheduling of workflows and resource sharing. Two key factors in cloud computing are: (i) scheduling of VMs and tasks with maximum resource utilization and (ii) meeting user requirements. Researchers have introduced several task-scheduling algorithms in the past few years. Although various scheduling algorithms have been proposed to achieve different objectives, there is still potential to improve the exploration and exploitation capabilities of these algorithms and to accelerate convergence toward the optimal solution.

Homsi et al. used the concept of a virtual machine pool, where the cloud tasks are differentiated by their type (Homsi et al. 2016). Authors used Green Workload Packing and Consolidation algorithm (GWPC) to allocate VMs and map service requests onto them. Boosting server utilization rates decreases the power consumption of the data center. GWPC adopts the reneging model to judiciously expunge service requests. However, the use of a traditional first-fit bin-packing algorithm may not guarantee the optimal solution. Mann et al. proposed an approximate virtualization algorithm using constraint programming technique for multicore-aware VM placement (Mann 2016). Constraint programming (CP) techniques often struggle with increasing problem size due to the exponential growth of the search space with the number of VMs and hosts. CP models do not adapt well to dynamic changes, such as varying workloads, dynamic resource availability, and changing placement criteria.

An optimized topology network diagram is established during virtual machine mapping by da Silva and da Fonseca (2016). The VM and server were modelled using the network diagram. The algorithm aims to minimize the communication costs between VMs by placing them closer to each other in terms of the data center's network structure. The focus of the work is the VMs communication demands and not the server's energy efficiency. Vilaplana et al. proposed GreenC, considering heterogeneity, workload, and communication distances for VM placement (Vilaplana et al. 2015). GreenC policy consolidates as many VMs as possible in a single host without overpassing its available resources. Energy consumption increases when adding more VMs to a host and leads to a sub-optimal solution. The authors considered only the availability of the resources when placing a VM without looking for an efficient host. Corradi et al. developed a virtual machine management framework to optimize network resources and reduce energy consumption (Corradi et al. 2014). The problem of re-optimizing the paths between virtual machines and data sources and re-balancing resource utilization in the cloud infrastructure has been tackled in Palmieri et al. (2016), where the greedy randomized adaptive search procedure (GRASP) has been used as a meta-heuristic. GRASP can get trapped in local optima due to its greedy nature. It also exhibits slow convergence. For very large-scale problems, GRASP is less effective compared to other meta-heuristics designed for scalability.

Chen et al. present a job-scheduling method for cloud computing that aims to reduce energy consumption while taking into account task dependency (Chen et al. 2022). The key objective of the proposed method is to split each job into smaller tasks and allocate them to virtual machines. The task scheduling is achieved using an efficient multi-objective artificial swarm algorithm. However, incorrect dependency management can lead to sub-optimal scheduling decisions, potentially negating the benefits of energy savings. The approach presented in Kumar et al. (2020) optimizes the allocation of virtual machines to physical hosts in cloud data centers, aiming to improve energy efficiency and enhance the quality of services for users. The method utilizes a modified lion optimization algorithm, which employs a levy flight distribution to randomly generate the population across the solution space. While Levy flights aid in exploring diverse solution spaces, they can sometimes hinder convergence to the global optimum. Poorly tuned parameters such as the step size and flight exponent may lead to sub-optimal performance or even instability in convergence.

Assudani et al. proposed inventive particle swarm optimization (IPSO) and the merge sort with divide and conquer (MSDC) approach, which can potentially improve the overall performance and resource allocation efficiency of cloud computing systems (Assudani and Balakrishnan 2022). The MSDC approach offers a dynamic resource allocation solution that can adapt to changing workload patterns and further improve the efficiency of cloud computing systems. The divide and conquer approach used by MSDC optimizes local decisions within subsets, but the merging phase may not always result in globally optimal resource allocations. Balancing efficiency with the need for global optimization and real-time responsiveness can be challenging in cloud environments. Ajmera et al. propose a Green-Particle Swarm Optimization (GPSO) algorithm to optimize the trade-off between energy consumption and Service Level Agreement (SLA) violations in Virtual Machine consolidation (Ajmera and Tewari 2023). VM consolidation distributes VMs across a pool of heterogeneous servers to enhance resource utilization and diminish power consumption. However, operating servers near full capacity increases the likelihood of SLA violations. GPSO can sometimes converge prematurely to a local optimum rather than the global optimum, especially in large search spaces. GPSO can cause network overhead due to the communications among the green particles.

The proposal presented in Abdullahi et al. (2022) adaptively tunes parameters for faster convergence speed using the symbiotic organisms search with the adaptive computation of benefit factors. But the authors did not consider energy consumption. Furthermore, the need for tuning and maintaining multiple parameters is time-consuming. Transferring tasks between domains with different feature spaces poses a challenge, especially in network traffic, where delays can lead to critical tasks not being delivered on time. To address this issue, the authors of Abualigah and Alkhrabsheh (2022) propose an optimization method for task scheduling called MVO-GA. MVO-GA combines a genetic algorithm (GA) and a multi-verse optimizer (MVO). However, both MVO and GA are prone to premature convergence to local optima (Pandey et al. 2012), and the performance is sensitive to many parameters. In Mangalampalli et al. (2022), a task scheduling approach is introduced that utilizes the Cat Swarm Optimization (CSO) algorithm to optimize various parameters. They calculated priorities for each task and virtual machine to facilitate the appropriate mapping of tasks onto VMs. The seeking mode in CSO involves evaluating multiple candidate solutions in each iteration. This can increase the time required for convergence. Further, achieving an optimal balance between exploration and exploitation can be challenging in CSO.

The hybrid firefly-bat algorithm (HFBA) integrates the dimension-based firefly algorithm and the modified bat algorithm to improve the population-diversity and global-exploration ability of original algorithm (Chen et al. 2019).

However, this algorithm is resource-intensive and time-consuming, which is a drawback in practical applications where computational efficiency is crucial. The performance of the algorithm heavily relies on the appropriateness of parameter tuning. For instance, the penalty function approach used for state variables processing is sensitive to the penalty coefficients. Bivasa et al. present a novel hybrid metaheuristic technique for virtual machine placement (VMP) aimed at optimizing energy consumption, carbon emissions, and quality of service parameters (Parida et al. 2024). The SSEPC algorithm, involving salp swarm optimization and emperor penguins colony algorithm, has a complex multi-stage process. The time complexity of finding a suitable server involves multiple stages. This complexity might result in higher computational overhead, especially in large-scale cloud environments. While the hybrid approach improves performance metrics, its scalability in real-world cloud environments with extensive and dynamic workloads is not thoroughly evaluated. The potential trade-offs between energy efficiency and user satisfaction under high-priority request scenarios have also not been analyzed.

Vijaya et al. propose "Ant Colony Optimization with Sine Cosine Algorithm" which is a hybrid approach combining the strengths of Ant Colony Optimization (ACO) and the Sine Cosine Algorithm (SCA) (Vijaya and Srinivasan 2024). Despite its potential benefits, there are several limitations to this research. Both ACO and SCA require careful tuning of multiple parameters (e.g., pheromone evaporation rate, control parameters, sine and cosine coefficients). This complexity can make the implementation and optimization process cumbersome and time-consuming. Although SCA helps in local search,

Table 1 Summary of the related work on VM placement techniques

Paper	Methods	Disadvantages
Homsi et al. (2016)	Green Workload Packing and Consolidation	GWPC adopts the reneging model to judiciously expunge service requests. The use of a traditional first-fit bin-packing algorithm may not guarantee the optimal solution.
Mann (2016)	Constraint Programming	CP techniques often struggle with scalability. CP can be less adaptable to environmental changes, and the solutions may be sub-optimal.
da Silva and da Fonseca (2016)	topology-aware VM Placement algorithm	Focused on the VMs communication demands than the server's energy efficiency, which may lead to sub-optimal solutions.
Vilaplana et al. (2015)	Green Cloud	Authors considered only the availability of the resources while placing a VM without looking for an efficient host.
Palmieri et al. (2016)	Greedy randomized adaptive search procedure	GRASP can get trapped in local optima due to its greedy nature and exhibit slow convergence. For very large-scale problems, GRASP is less effective compared to other meta-heuristics designed for scalability.
Chen et al. (2022)	multi-objective artificial swarm algorithm	Incorrect dependency management can lead to sub-optimal scheduling decisions, potentially negating the benefits of energy savings. Chances of getting stuck into local optima.
Kołodziej et al. (2015); Reddy et al. (2020a)	Genetic algorithm	Computationally expensive. Slow convergence rates. Can get stuck in local optima.
Ibrahim et al. (2020); Dashti et al. (2016)	Particle swarm optimization	Slow convergence rate. Weakness in local search. Possibility of getting stuck in local optima in high-dimensional search spaces.
Kumar et al. (2020)	Modified Lion Optimization algorithm	The use of Levy flights sometimes hinders convergence to the global optimum. Poorly tuned parameters such as the step size and flight exponent may lead to sub-optimal performance or even instability in convergence.
Assudani and Balakrishnan (2022)	IPSO and the merge sort with divide and conquer approach	The divide and conquer approach used by MSDC optimizes local decisions within subsets, but the merging phase may not always result in a globally optimal resource allocations.
Ajmera and Tewari (2023)	Green-Particle Swarm Optimization	GPSO can sometimes converge prematurely to a local optimum, especially in large search spaces. GPSO can cause network overhead due to the communications among the green particles.
Abdullahi et al. (2022)	symbiotic organisms search with adaptive computation of benefit factors	The authors didn't consider energy consumption. The need to tune and maintain multiple parameters is time-consuming.
Mangalampalli et al. (2022)	Cat Swarm Optimization	The seeking mode in CSO involves evaluating multiple candidate solutions in each iteration. This can increase the time required for convergence.

Table 1 (continued)

Paper	Methods	Disadvantages
Abualigah and Alkhrabsheh (2022)	Combined genetic algorithm and multi-verse optimizer	MVO and GA are prone to premature convergence, and the performance of this approach is sensitive to many parameters.

there is still a risk of the algorithm getting trapped in local optima, particularly in highly multimodal landscapes. The balance between exploration and exploitation is critical and difficult to achieve consistently. While the hybrid algorithm aims to improve efficiency, the time complexity could become a serious issue when dealing with very large datasets or problem spaces. The performance of this hybrid algorithm heavily depends on the quality of the heuristic information and the initial solution set. Poor heuristics can lead to sub-optimal solutions and increased convergence time. Ndayikengurukiye et al. propose Grey Wolf Optimization (GWO) algorithm for VM placement, which is a powerful and widely used optimization approach (Ndayikengurukiye et al. 2024). GWO is prone to premature convergence, it is sensitive to its parameters and to the number of iterations; furthermore, the population of solutions can lose diversity, leading to a lack of exploration in the search space.

The summary of the related work is presented in Table 1, showing how researchers have predominantly examined optimization algorithms tailored for homogeneous data centers. In this paper, we develop a novel optimization algorithm for virtual machine placement in a heterogeneous environment. In our previous works, we have used particle swarm optimization (PSO), GA, and Hybrid approaches for solving the VM placement problem (Reddy et al. 2019, 2020b). While the approaches were effective, we also observed some disadvantages. The genetic algorithm approach requires a small amount of information, and using the operators is difficult. GA is computationally expensive and bound to have over-fitting with the sample data. Due to the randomness involved in the operators, GA has very slow convergence rates, and it can get stuck in local optima. Implementing PSO is easier and is based on the concept of social interaction. PSO can be tuned quickly. The disadvantages of PSO are its slow convergence rate, weakness in local search, and the possibility of getting stuck into local optima in the case of high-dimensional search spaces. This is partially alleviated with PSO variants, which support controlling the velocity and the stable convergence.

Apart from PSO, and GA researchers used several other meta-heuristic approaches like ACO (Vijaya and Srinivasan 2024; Zhao et al. 2023), bees algorithm (Javadi-Moghaddam and Zahra 2023), bacterial foraging optimization (BFO) (Xu et al. 2020), etc. to solve VM placement problem. But most of these algorithms involve many parameters and results will depend on how we tune these parameters to yield optimal solutions. To solve these problems, we invented an optimization algorithm with a limited number of parameters and more exploration capabilities based on imitating behavior of the humans. The proposed algorithm mainly focuses on improving the convergence speed of the scheduling algorithm that appropriately places a VM onto a server, considering various parameters like CPU utilization, bandwidth utilization, etc. The proposed approach can improve the performance and energy efficiency of the heterogeneous data centers, as we will show later in the paper.

Problem formalization

In the operational management of a cloud-based data center, the task involves strategically distributing applications with varying computing resources among virtual machines (VMs), which are responsible for handling a multitude of workloads simultaneously. The method used to select and allocate these VMs significantly impacts the data center's server utilization and power consumption. To ensure efficient, large-scale heterogeneous scheduling, reducing the data center's energy consumption while maintaining high-performance levels is essential. Essentially, a virtual machine allocation refers to mapping virtual machines onto their respective physical hosts so that a VM is allocated to only one server and the sum of the resource requirements of VMs on a server does not exceed the available resources.

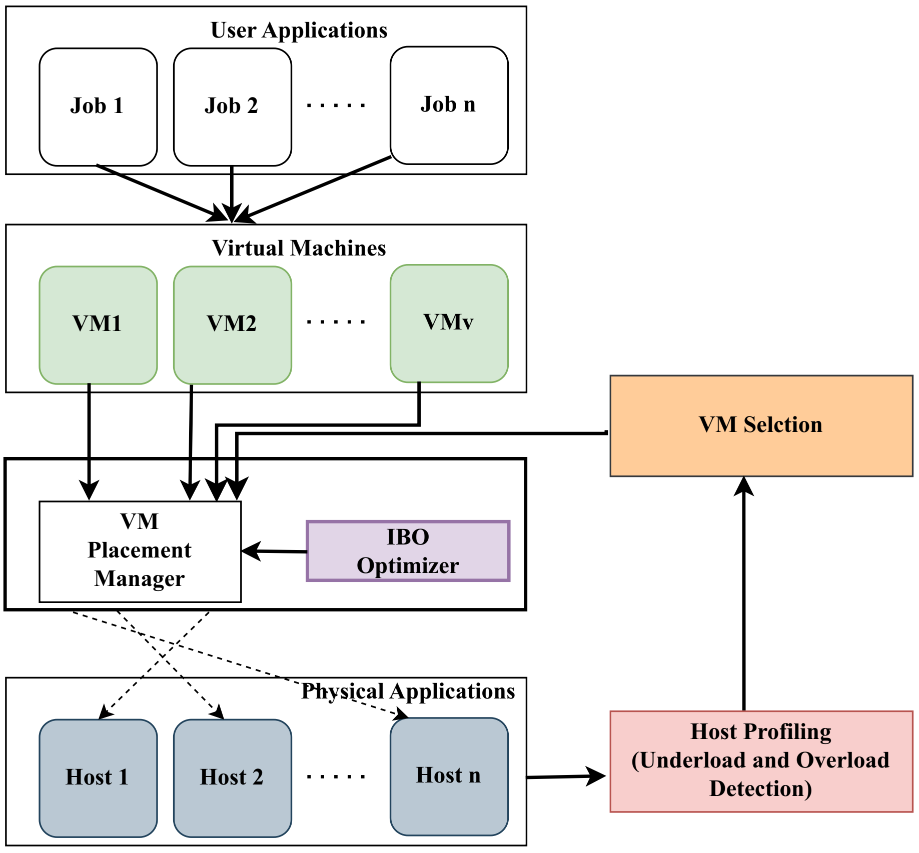

Fig. 2 Representation of the VM placement problem in a data center

Each data center consists of several physical machines/servers. The hypervisor creates and manages VMs on these servers. For each user request (e.g., deploying an application, running a service), the cloud management system allocates a VM to handle it. These VMs will be placed onto the hosts by VM Placement Manager (VPM) as shown in Fig. 2. Further, data centers use monitoring tools for host profiling. These collect and analyze data about the hosts to understand their performance and resource utilization. When a host is over/underutilized, the VM Selection module picks the VMs from these hosts for migration. The VPM will map these VMs to other efficient hosts along with the new incoming requests. All the servers in the data center are heterogeneous in nature, and they have a fixed number of processing units, memory, and bandwidth. Each of these servers also has its own power consumption characteristics. Let $V = VM_1, VM_2, ..., VM_{nv}$ be the set of virtual machines that need to be allocated across the servers $S = S_1, S_2, S_3, ..., S_{np}$. All the User tasks are managed by the VMs in the data center. Each particle or solution to this placement problem is modelled as an array consisting of the identifier of the corresponding server on which the VM (V_i) is to be placed, e.g,

$$Solution_i = (H_{10}, H_5, ...H_m, ..., H_2).$$

The above solution places the V_1 on to H_{10}, V_2 on to H_5, and so on. The fitness of this solution is the power consumed by all the servers after placing these VMs onto servers. As commonly done, we assume that the power of each server has a linear relationship with its CPU utilization (Reddy et al. 2019; Beloglazov and Buyya 2012; Fu and Zhou 2015). The SPEC 2008 benchmark benchmark provides a standardized way to compare the energy efficiency of different server configurations and helps in understanding the power consumption characteristics of servers under varying load conditions. We have used these power consumption characteristics of each server at increasing utilization levels (0%, 10%, 20%...100%). Using this data, we estimated the power consumption of a host resorting to the following equation derived from (Reddy et al. 2019; Beloglazov et al. 2012):

$$S_i(u) = k * S_i^{max} + (1 - k) * S_i^{max} * u, \tag{1}$$

where S_i^{max} is the power consumption of a server at 100% utilization, u is the CPU Usage of the server that represents how much processing power is being used, and k represents the percentage of power drawn when the server is idle. The utilization of the CPU varies with time due to the workload variability. Therefore, the total energy consumption by a host can be defined as an integral of the power consumption function over a period of time as shown in Eq. (2) (Beloglazov et al. 2012).

$$E = \int_{t_0}^{t_1} S_i(u(t))dt \tag{2}$$

VM allocation using imitation based optimization

We invented an imitation-based optimization algorithm rooted in swarm intelligence to optimize convergence time and enhance the consistency of attaining optimal solutions. This approach leverages human learning and imitating behavior. Essentially, particles within the swarm mimic the behavior of a leader. As particles observe the leader's actions, they memorize some or all of the leader's behavior and attempt to replicate it with a certain level of confidence.

Imitation plays a crucial role in social learning, where particles imitate the observed behavior to achieve desired outcomes. However, achieving the exact result may vary, leading to differences in particle states. The observed individual, or leader, is a key factor in this optimization strategy. Similar to social dynamics, individuals are influenced more by those with status and power. In our context, particles are more likely to mimic a leader if they exhibit a superior fitness value compared to other particles in the swarm. Furthermore, the leader's influence is particularly potent if they have attained a favorable position through effective actions. This imitation-based approach aims to expedite convergence and enhance the reliability of obtaining optimal solutions in optimization problems.

Algorithm 1 Mahalanobis distance calculation

```
1: **Input:**Two particles $P1$ and $P2$ with same number of dimensions converted as matrices
2: **Output:** Mahalanobis Distance between the given particles
3: Center each Column of matrices $P1$ and $P2$ to their mean to get $\hat{P}1$ and $\hat{P}2$.
4: Compute the covariance matrices for $\hat{P}1$ and $\hat{P}2$ as:
       $$C_1 = \frac{1}{n1} * \hat{P}1^T * \hat{P}1$$
       $$C_2 = \frac{1}{n2} * \hat{P}2^T * \hat{P}2$$
   where $n1, n2$ are the number of rows in $\hat{P}1, \hat{P}2$ matrices respectively.
5: Compute the pooled covariance (PC) matrix using $C_1$ and $C_2$ as:
       $$PC = \frac{1}{n1 + n2}[n1 * C_1 + n2 * C_2]$$
6: Compute the difference in means of given matrices $P1$ and $P2$ as:
       $$MeanDiff = mean(P1) - mean(P2)$$
7: Compute the Mahalanobis distance as:
       $$MD = \left\{ \begin{array}{ll} \sqrt{MeanDiff * PC^{-1} * MeanDiff^T} & if\ inverse\ of\ PC\ is\ possible \\ 1 & otherwise \end{array} \right\}$$
```

IBO is a population/swarm-based algorithm that uses a group of individuals representing candidate solutions upon initialization. Each individual starts with a random position, indicating a potential mapping of VMs to servers. Each individual's fitness is assessed, indicating the energy consumption of the given mapping. Individual solution that has minimum energy consumption is designated as the leader. The information about the leader is communicated to all individuals. The proposed IBO method has two phases: the Grouping phase and the Imitation phase.

Grouping phase

The first phase of "grouping" is when the individuals similar to the best participant are grouped together. The grouping process considers factors such as proximity of solutions, similarity in characteristics, or other criteria. In the proposed algorithm, proximity is computed using the Mahalanobis Distance (MD), as described in Algorithm 1. MD calculates the distance between two features/columns based on correlations. MD is used to identify different patterns and is analyzed with respect to a reference axis (McLachlan 1999). Unlike other multivariate measurements, Mahalanobis Distance takes into account the covariance structure of the data, which makes it robust to outliers. This makes it particularly useful when dealing with high-dimensional data where variables may be correlated. Mahalanobis Distance is normalized and not affected by the scale of the variables (De Maesschalck et al. 2000). Similarity/proximity between the two candidate solutions is computed using MD as described in Algorithm 1. First, we calculate the covariance matrices using the equations in Line 4. Using these covariance matrices, we calculate the pooled covariance matrix, as shown in Line 5. In Line 6, we calculate the mean difference row vector. Finally, MD is calculated using the given equation in Line 7.

Initially, fitness of all the individuals in the swarm is calculated and the one with with the lowest fitness value is designated as the Leader. In the grouping phase, we proceed to calculate the MD between the leader and all other individuals. This value indicates how far an individual is from the Leader. The smaller the value of MD, the more similar are the individuals. Then, all the individuals whose MD value is less than the grouping factor (*gf*) are added to a group called "explorers", and the remaining individuals are added to a group called "succeeders." In our proposed approach, we update only the individuals in the succeeders group as they are moving away from the Leader. Updating the individuals is done in the Imitation phase. If the *gf* value is high, more individuals will be added to the explorers group. This will decrease the exploration capability as these individuals are not updated. The tuning of this parameter is thus crucial for the successful execution of the optimization. After extensive experiments we found that a *gf* value of 0.5 balances the exploration and exploitation properties well.

Imitation phase

Algorithm 2 Individual Update

```
1:  Input: current individual ($CI$), random individual from explorer group ($RI$).
2:  Output: Updated individual.
3:  Calculate the mean difference (Diff) between each column of $CI$ and $RI$ as:
        $Diff = mean(CI) - mean(RI)$
4:  $m$ = Number of rows in $CI$,
5:  $n$ = Number of columns in $CI$.
6:  for i=1 to $m$ do
7:      for j=1 to $n$ do
8:          $CI[m][n] = CI[m][n] + Diff[n]$
9:      end for
10: end for
11: Calculate the fitness of the updated individual $CI$.
12: Create a new random individual ($N$) and Calculate the fitness of $N$.
13: if fitness($CI$) < fitness($N$) then
14:     Return $CI$
15: else
16:     Return $N$
17: end if
```

In the imitation phase, each individual in the succeeders' group is updated by imitating a random candidate from the explorers' group, as shown in Algorithm 2. Chosing a random candidate from the explorers' group will increase the exploration capability more than following a single best particle; as done in PSO and other algorithms. First, we calculate the mean difference between the current individual and a random candidtae from the explorers (line 3). This mean difference is then added to the each element of the current individual (line 8). This process decreases the MD between these two individuals. After updating each individual, we generate a random individual and compare the fitness of updated and random individuals (line 13–16). The individuals with good fitness values will be returned from the imitation phase. This random particle generation increases the exploitation capabilities of the proposed algorithm.

Then, Algorithm 3 forms the core of the imitation-based optimization. The algorithm starts with a random population where each individual represents a VM-Host mapping. The fitness of each individual is calculated based on a certain criterion, in this case, energy consumption. The individual with the lowest energy consumption value is designated as the leader. All individuals in the population are informed of this. The population is then divided into two groups: explorers and succeeders. This is done using the proximity of each individual to the leader, calculated using MD (Algorithm 1). If the MD between an individual and the leader is less than the grouping factor, it is added to the explorer's group. Otherwise, it is added to the succeeders group. Individuals in the succeeders group attempt to update their current position using the Algorithm 2.

The imitation process is repeated until one of the following stopping conditions is met. The best participant is re-evaluated after each iteration, and the process continues up to maximum iterations (*k*) or until there are continuous improvements for a specified number of times (*r*) or until stabilization for a certain number of times (*s*). The solution with the highest fitness (lowest energy consumption) after the final iteration is the best participant and is returned as the final solution.

Algorithm 3 Optimization using IBO

```
Input: List of VMs and Servers
Output: Mapping of VMs to Servers
Init:
    $grouping\ factor(gf) = 0.4, Flag = false, max = 0,$
    $population\ size = 40, r = 3, improvement = 0, noChange = 0, and\ s = 10$
Create a population of individuals ($X_i$) representing VM to Server mappings randomly.

for all individual $\in$ population do
    fitness[individual] = compute the fitness (energy consumption) of the individual using Equation. 1.
    $L[individual] = X_i(t)$. // L is the personal best
end for
$\hat{GL}(t)$= return the individual with minimum energy consumption in the population. // Global Best
initFitness = fitness.
while $max <= 150$ do
    for all individual $\in$ population do
        $Dist$ = Calculate the proximity between the $\hat{GL}(t)$ and individual using MD.
        if $Dist < gf$ then
            $explorers$.add(individual).
        else if
            then$succeeders$.add(individual).
        end if
    end for
    for all individual $\in$ succeeders do
        $X_i(t+1)$= Update the individual using Algorithm 2.
        fitness[individual]= Compute the fitness of $X_i(t+1)$.
        if fitness[individual] < initFitness[individual] then
            $L[individual] = X_i(t+1)$.
        end if
    end for
    $\hat{GL}(t+1)$= return the individual with minimum energy consumption in the population.
    if fitness $[\hat{GL}_j(t+1)] < fitness[\hat{GL}_j(t)]$ then
        $\hat{GL}_j(t) = \hat{GL}_j(t+1)$.
        $improvement$ ++;
        Flag = True;
        if $improvement > r$ AND Flag then
            Return $\hat{GL}_j(t)$.
        end if
    else
        if fitness$[\hat{GL}_j(t)] == fitness[\hat{GL}_j(t+1)]$ then
            $noChange$ ++;
            Flag = false;

            if $noChange > s$ then
                Return $\hat{GL}_j(t)$.
            end if
        else
            Flag = false.
            $improvement = noChange$= 0.
        end if
    end if
    max++;
end while
Return $\hat{GL}_j(t)$.
```

Migration is necessary for system maintenance. Maintainance does not disrupt operations, optimization of resource pools and fault prevention. We perform migration of the VMs to increase the efficiency and to ensure the SLAs. We choose two extreme states of the hosts for migration: over-utilization and under-utilization. If a server is under-utilized, then we select all the VMs in that host and migrate them to another efficient host. If a server is over-utilized, then we have to carefully select the virtual machines for migration until the host utilization drops below the upper threshold. We have used a selection algorithm called MBS-VM proposed by Reddy et al. (2020b) that considers the utilization of memory, bandwidth, and size of the VM. These selected VMs along with the new request from users will be optimally placed on to the new hosts using the IBO algorithm.

Experimental results

To evaluate the proposed IBO algorithm and approach, we resort to CloudSim (Calheiros et al. 2011) and simulate a small data center with 100 servers. The CloudSim toolkit is the de facto standard for research in data center optimization. It provides a simulation platform that enables the modeling of virtualized environments and facilitates on-demand resource provisioning and management. We analyzed the proposed approach in detail, focusing on data center energy consumption, active servers, convergence, and consistency of the algorithm.

Experimental setup

In a cloud computing environment, a VM request is characterized by several key parameters, typically including:

- *Bandwidth:* Bandwidth refers to the data transfer rate between the VM and the network. It is a measure of the volume of data that can be transmitted over the network in a given amount of time, usually measured in megabits per second (MBPS) or gigabits per second (GBPS).
- *RAM (Random access memory):* The RAM parameter specifies the amount of memory required for the VM. It is a critical factor in determining the VM's ability to run applications and process data efficiently. RAM is usually measured in gigabytes (GB).
- *MIPS (Million instructions per second):* MIPS is a measure of the amount of computational power in the unit of time. In our case, it indicates the number of million instructions the VM can execute per second, reflecting the processing capability of the underlying CPU.
- *Storage:* Storage refers to the amount of secondary memory required by VM to store the application data, logs, backups, etc. Storage is usually measured in gigabytes (GB).

These parameters are crucial in the allocation and scheduling of VMs in a cloud environment. Providers need to ensure that each VM is allocated sufficient resources to meet its specified requirements, optimizing the overall performance and efficiency of the cloud infrastructure. Resource requests from users are heterogeneous in nature. In our experiments, we used three types of VMs. Each VM type is characterized by various RAM, MIPS, and bandwidth requirements. The characteristics of these VM types are presented in Table 2. We simulate a data center that has multiple servers that differ in terms of hardware specifications and possibly network configurations. We have used three types of servers with various capabilities, as listed in Table 3.

The initial parameters used in our experiments are presented in Table 4. We iterate the particle updating until three continuous best solutions are found, or there is no improvement in the solution for 10 iterations, or the maximum number of iterations (150) is reached. We have chosen the maximum iterations as 150 after rigorous experiments. Initially, we started with 50 iterations and gradually increased the maximum iterations. We observed that increasing the maximum iterations from 50 to 100 led to an improvement in the optimal solution. However, after 150 iterations, only a little change was observed in the optimal solution. Therefore, we fixed the maximum iterations to 150. Similar experiments were performed for the grouping factor. If the grouping factor is above 0.5, we observed that the exploration capability of the model decreases. Therefore, we set the *gf* to 0.4.

Effect on energy consumption

We compare the results of our proposed IBO with the solutions of the best-known approaches, namely, Hybrid best-fit heuristic (HBF) (Jangiti et al. 2020), GA (Kołodziej et al. 2015), PSO (Kumar and Raza 2015), Hybrid approach (Reddy et al. 2020b), Extended PSO (EPSO) (Potu et al. 2021), and Integer Linear Programming (ILP) (Gonzalez and Tang 2020). We consider several measures to evaluate the efficiency of the proposed approach. First, we compare the number of active hosts. The lower the number of active hosts, the lower the energy consumption of the data center. Active hosts are observed for various approaches, with 20 hosts and 50 VMs in the data center. The results are shown in Fig. 3. The proposed approach reduces the number of active machines by placing more workload on an efficient physical machine. Figure 3 shows that the Hybrid, EPSO, and IBO approaches result in 9, 7, and 7 active hosts, respectively. In contrast, the number of active physical machines with HBF, GA, PSO, and ILP are 15, 14, 12, and 12, respectively. Thus, IBO uses fewer hosts while ensuring strict SLAs.

For analyzing the energy consumption, we performed extensive experiments on a data center with 100 Hosts and varying the number of VMs from 50 to 300. The energy consumption (in kWh) details of various approaches are presented in Table 5 and Fig. 4. With 300 virtual machines, IBO, EPSO, and Hybrid approaches lead to an energy consumption of 2.08 kWh, 2.12 kWh, and 2.25 kWh, respectively; whereas HBF, GA, PSO, and ILP approaches consume 3.15 kWh, 2.93 kWh, 2.47 kWh, and 2.50 kWh, respectively as shown in Table 5. The proposed approach is capable of exploring the full search space and thus providing the optimal solution. Thus, with the proposed IBO, the energy consumption decreased at an average of 7%, 10%, 11%, 28%, 17%, and 35% compared to Hybrid, EPSO, PSO, GA, ILP, and HBF, respectively.

Migrations and SLA performance analysis

The number of VM migrations is a critical metric in cloud and virtualization management, reflecting the health, efficiency, and performance of the infrastructure. By understanding and managing VM migrations, organizations can ensure optimal resource utilization, maintain high availability, and minimize downtime. We use the proposed VM allocation method to improve the utilization of the servers in a data center alongside

Table 2 Virtual machines requirements

	Bandwidth (MBPS)	RAM	Storage (GB)	MIPS
VM Type 1	100	100	2.5	613
VM Type 2	100	200	2.5	870
VM Type 3	100	300	2.5	1740

Table 3 Physical machines configurations

Host Type	Bandwidth (GBPS)	RAM (GB)	Storage (GB)	MIPS
G4 Xeon 3040	30	8.5	125	3000
G4 Xeon 3075	30	8.5	125	3000
G3 Pentium D 930	30	8.5	125	3000

Table 4 Initial parameters

Parameters	Value
Populations size	40
Maximum Iterations	150
Learning parameters	$k_1 = 3, k_2 = 2$
Grouping factor	0.4
Mutation probability	0.8
Crossover probability	0.7
Selection	Roulette wheel selection

VM migration procedures that support virtualized cloud computing. VM migration must happen without violating the prerequisite of execution and workload confinement.

We present the analysis of the total count of migrations for each placement algorithm in a data center with 100 hosts. Figure 5 shows the comparative analysis of the total number of migrations by increasing the number of VMs from 50 to 300. One notices that the IBO approach entails the minimum migrations in all cases compared to EPSO and other approaches. The overall SLA violations for 50 and 100 virtual machines are largely consistent across all approaches. The results in Table 6 clearly show that the proposed approach has lower SLA violations compared to the other approaches.

Performance analysis in terms of consistency and convergence

The convergence rate is the time each approach takes to reach the global optimum solution. This work considers the average number of iterations for achieving global optimum. We plot the convergence of the IBO, Hybrid, and EPSO algorithms for 100 hosts and varying VM requests, ranging from 50 to 300.

The convergence rates of IBO, EPSO, and Hybrid approaches for varying VM requests are given in Fig. 6. For example, with 100 VMs, the average number of iterations for reaching the global optimum is 25, 28, and 32 for IBO, EPSO, and Hybrid, respectively. From the experiments, we observe that the IBO approach performs better for a higher number of virtual machines. IBO decreases the convergence time up to 35% and 13%

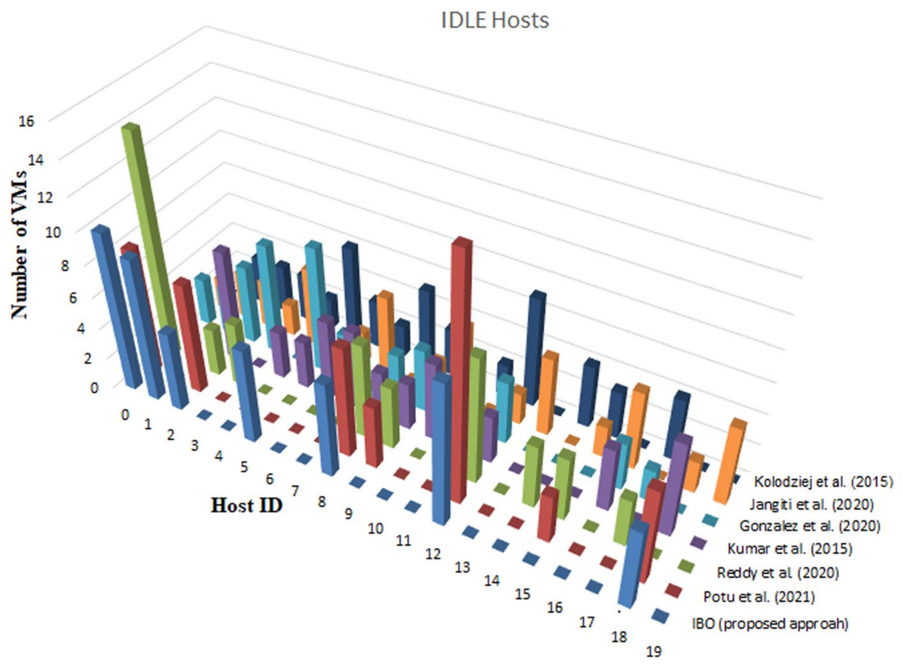

Fig. 3 Number of VMs allocated for each active host in a data center with 20 hosts and 50 VMs

compared to Hybrid and EPSO, respectively. Furthermore, it demonstrates greater consistency in finding the optimal solution and a faster convergence rate even with a higher number of VM requests. IBO and EPSO are clearly superior to Hybrid in terms of convergence. As the number of VMs increases, IBO outperforms EPSO and Hybrid approaches. The consistency of each algorithm is calculated as the number of times the global optimum solution is found versus the number of runs. In other words, we calculated how many times the global optimum was achieved when repeating the experiment for a fixed number of times. Figure 7 plots how many times each optimum is achieved. We compared the proposed algorithm with other algorithms by running the experiment with 200 virtual machines and repeating it 50 times. From the experimental analysis, we see that the proposed approach reached the global optimum 18 times (36%), whereas the EPSO achieved the same global optima only 7 times (14%). Further, we observed that PSO, GA, ILP, and HBF are not able to achieve the global optimum point at 1.12 while our algorithm does. Figure 7 also shows that IBO and EPSO are able to explore all the optimal points, whereas the remaining algorithms fail and get stuck in local optima.

Robustness and scalability analysis

As data centers grow in size and complexity, scalability becomes a significant challenge. Many existing VM placement methods struggle to maintain performance and efficiency at scale. We study the scalability and robustness of the proposed approach by including more varied scenarios, such as varying data center sizes and increasing the workload. First, we perform experiments by increasing the data center size (number of hosts) from 500 to 5000 and fixing the number of VMs to 1000. We use the same parameters, VM, and Host characteristics discussed in "Experimental setup".

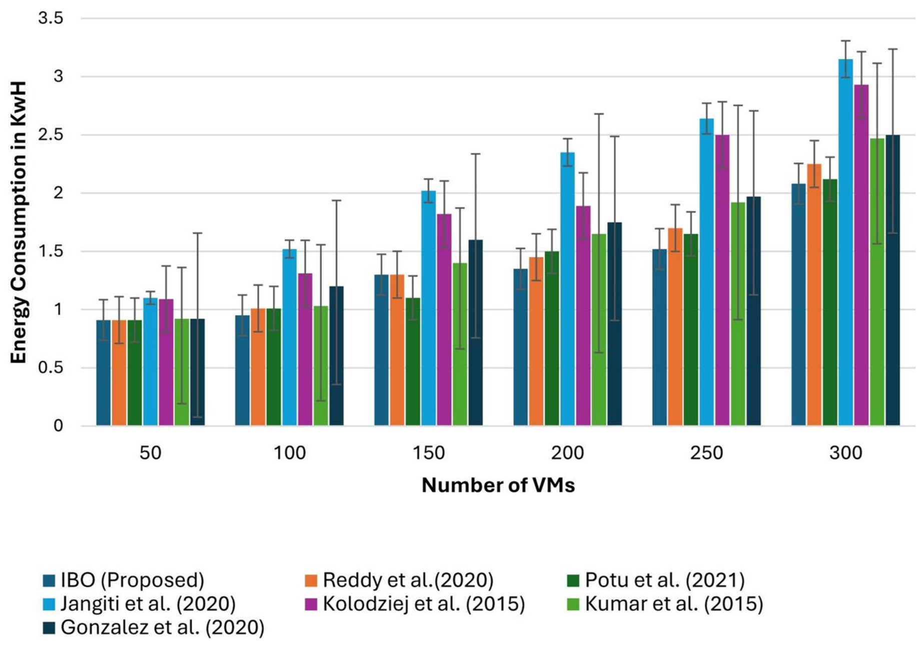

Fig. 4 Comparative analysis of energy consumption for state-of-the-art approaches in a data center with 100 hosts and varying VMs

Table 5 Energy consumption of state-of-the-art approaches with 100 hosts and varying VMs

Energy consumption (kWh)						
Approach/VMs	**50**	**100**	**150**	**200**	**250**	**300**
IBO (Proposed)	0.91	0.95	1.3	1.35	1.52	2.08
Reddy et al. (2020a, b)	0.91	1.01	1.30	1.45	1.70	2.25
Potu et al. (2021)	0.91	1.01	1.10	1.50	1.65	2.12
Jangiti et al. (2020)	1.10	1.52	2.02	2.35	2.64	3.15
Kolodziej et al. (2015)	1.09	1.31	1.82	1.89	2.50	2.93
Kumar and Raza (2015)	0.92	1.03	1.40	1.65	1.92	2.47
Gonzalez and Tang (2020)	0.92	1.20	1.60	1.75	1.97	2.50

The analysis helps understand how the proposed IBO approach is able to find the global optimal solution even in large search spaces. From the results in Table 7, we see that there is not much difference in the energy consumption of the data center, even if the number of hosts is increasing. This indicates that the proposed algorithm is able to choose efficient hosts and place all the VMs in a minimum number of hosts. The rapidly changing workload patterns in cloud environments require adaptable and flexible VM placement strategies. We tested the proposed approach by increasing the number of VMs from 500 to 5.000 in a data center with 200 hosts. With 5000 VMs, the proposed approach can save 43.8% energy consumption compared to HBF approach (Jangiti et al. 2020). This leads to monthly energy savings of 501 kWh. Assuming the cost of €0.402/KWh and an emission factor of 0.92 lbs CO_2 /kWh, the monthly cost savings will be €201.4 euro and CO_2 Savings will be 460.92 lbs CO_2/month. Looking at Table 8, one notices that the proposed approach has

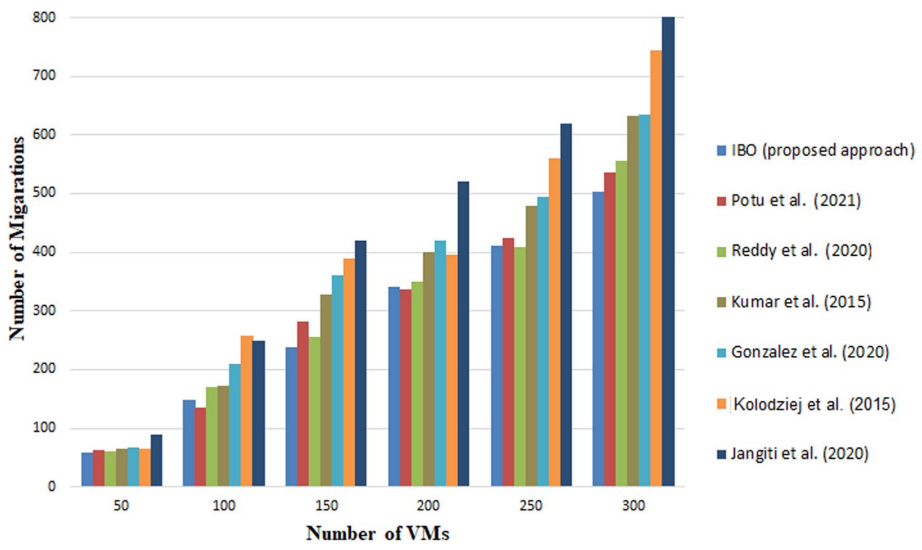

Fig. 5 Comparative analysis of the total number of migrations for various placement algorithms with 100 hosts and increasing number of VMs

Table 6 Overall SLA violations

Overall SLA violations (%)						
Approach/VMs	**50**	**100**	**150**	**200**	**250**	**300**
IBO (proposed)	0.41	1.14	1.12	1.2	1.67	2.09
Reddy et al. (2020a, b)	0.72	1.25	1.32	1.25	2.03	2.26
Potu et al. (2021)	0.72	1.21	1.26	1.19	1.72	2.17
Gonzalez and Tang (2020)	0.75	1.27	1.4	1.5	1.95	2.2
Kolodziej et al. (2015)	0.75	1.34	1.54	1.66	2.21	2.41
Kumar and Raza (2015)	0.94	1.38	1.52	1.7	2.5	2.78
Jangiti et al. (2020)	1.02	1.75	1.69	1.84	2.4	3.13

less energy consumption compared to the state-of-the-art approaches, even with the growing workload. Further, we analyzed the total simulation time in CloudSim. Total simulation time includes the time for VM placement, VM selection, migration, and job execution. With 200 hosts and 5000 VMs, our proposed approach took 1 h 30 min, whereas other approaches took up to 2 h 25 min. GA, PSO, and Hybrid approaches took more time to converge than the proposed approach, and the remaining approaches had more migrations than the proposed approach, which led to more simulation time.

We compared our proposed approach with each of the other approaches by performing the following statistical tests. We use the T-test to compare the means of the two groups to see if they are statistically significantly different from each other. The p-value indicates the probability of obtaining results as extreme as those observed. A low p-value (typically < 0.05) suggests that the observed differences are unlikely to be due to chance, thereby indicating statistical significance. To conduct this test, we have repeated the experiment with 100 hosts and 500 VMs for 30 times with each approach. We use paired two-tailed T-test (Zimmerman 1997) to see the obtained

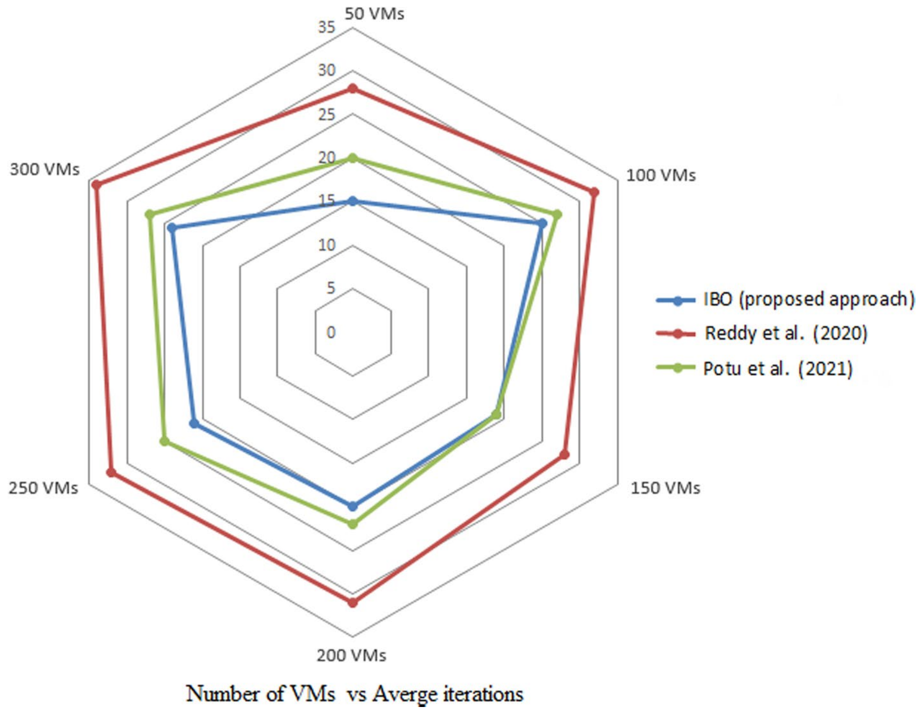

Fig. 6 Global optimum convergence analysis with respect to the average number of iterations

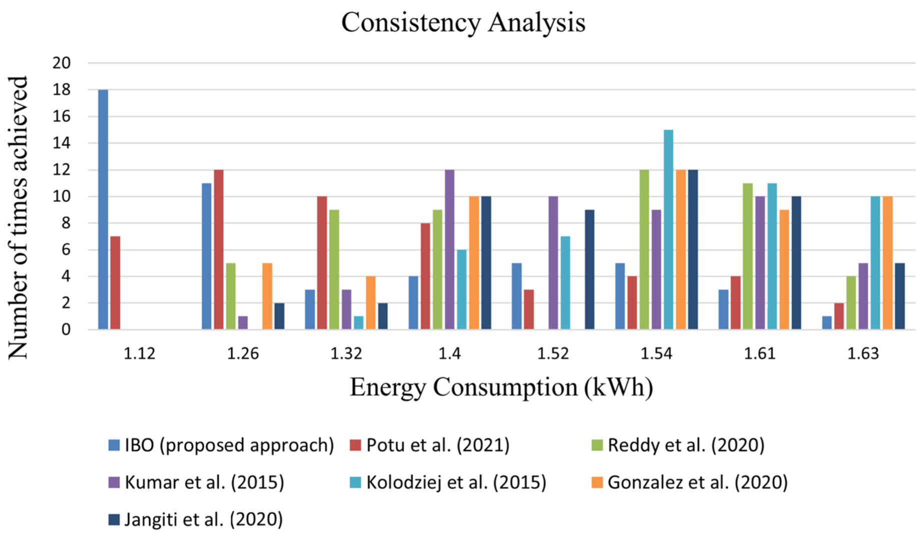

Fig. 7 Consistency analysis

mean values of all approaches have a distinguished difference with 58 degrees of freedom at 0.05% level of significance. These results with t-value and p-value are presented in Table 9. The obtained values indicate strong statistical significance, as all t-values are positive and all p-values are less than 0.00001. This suggests that our proposed method demonstrates a statistically significant improvement over other methods.

Table 7 Scalability analysis of IBO

Scalability analysis of the IBO algorithm for varying data center sizes						
Number of hosts	500	1000	2000	3000	4000	5000
Number of VMs	1000	1000	1000	1000	1000	1000
Energy consumption	5.49 kWh	5.51 kWh	5.51 kWh	5.50 kWh	5.55 kWh	5.51 kWh
Number of VM migrations	11865	11696	12034	13147	13690	9706

Table 8 Comparative analysis of energy consumption with increasing workloads for 200 Hosts

Number of VMs	**500 (kWh)**	**1000 (kWh)**	**2000 (kWh)**	**3000 (kWh)**	**4000 (kWh)**	**5000 (kWh)**
IBO	2.74	4.89	8.95	13.23	17.15	21.42
Potu et al. (2021)	2.97	4.99	9.25	14.02	18.27	22.31
Reddy et al. (2020b)	2.93	5.27	9.60	14.56	18.45	22.83
Kumar and Raza (2015)	3.11	5.62	9.70	14.75	18.98	23.49
Gonzalez and Tang (2020)	3.30	5.76	10.34	15.63	20.47	25.42
Kołodziej et al. (2015)	4.18	8.33	13.69	20.25	26.89	33.22
Jangiti et al. (2020)	7.26	9.52	16.35	23.65	28.43	38.12

Discussion and conclusions

Our work focuses on optimizing power consumption and resource utilization in a data center with heterogeneous server configurations. Most of the existing algorithms for solving the VM placement problem have slow convergence rates, weakness in local search, and run the risk of getting stuck into local optima in the case of high-dimensional search spaces. These approaches involve many parameters, and the results will depend on how one tunes them. Further, these algorithms fail to balance between exploration and exploitation. To address these limitations, we proposed a novel imitation-based optimization algorithm with fewer parameters to improve

Table 9 T-test statistical analysis results for 100 hosts and 500 VMs

	IBO (proposed)	**Potu et al. (2021)**	**Reddy et al. (2020b)**	**Kumar and Raza (2015)**	**Gonzalez and Tang (2020)**	**Kołodziej et al. (2015)**	**Jangiti et al. (2020)**
IBO (proposed)	*						
Potu et al. (2021)	t: 8.5939 $p < 0.00001$	*					
Reddy et al. (2020b)	t: 9.1311 $p < 0.00001$	t: 6.7983 $p < 0.00001$	*				
Kumar and Raza (2015)	t: 8.5939 $p < 0.00001$	t: 4.3707 $p < 0.00001$	t: 2.4909 $p < 0.00001$	*			
Gonzalez and Tang (2020)	t:29.1319 $p < 0.00001$	t: 20.6052 $p < 0.00001$	t: 16.6166 $p < 0.00001$	t: 6.7983 $p < 0.00001$	*		
Kołodziej et al. (2015)	t: 72.2772 $p < 0.00001$	t: 64.2367 $p < 0.00001$	t: 59.7736 $p < 0.00001$	t: 35.0676 $p < 0.00001$	t: 40.0430 $p < 0.00001$	*	
Jangiti et al. (2020)	t: 507.5297 $p < 0.00001$	t: 341.6171 $p < 0.00001$	t: 271.5684 $p < 0.00001$	t: 83.3146 $p < 0.00001$	t: 141.4511 $p < 0.00001$	t: 59.5184 $p < 0.00001$	*

*indicates Not Applicable/Not Required

the resource allocation strategy. The proposed IBO employs fewer servers for the VM placement and has greater exploration and exploitation capabilities than state-of-the-art approaches. The IBO algorithm explores the search space rapidly thanks to the proposed imitation phase as part of Algorithm 2, where each particle imitates a random leader and reaches the global optimum. In most swarm-based approaches, particles try to reach the location of a single global best particle. This sometimes leads to falling into local optima. To avoid this, the proposed approach considers multiple best particles as Leaders and randomly chooses one to imitate, which significantly improves the exploitation capability. Furthermore, we introduce randomness in every iteration to ensure we can obtain a better solution than the one obtained after imitation. This guarantees exploration and escapes from local optima.

The detailed analysis showed that the proposed IBO approach significantly reduces energy consumption when compared to EPSO, Hybrid, PSO, ILP, GA, and HBF. We found that there is a reduction of up to 43.8% in energy consumption with increasing workloads. The proposed IBO algorithm reduces the energy consumption at an average of 7%, 10%, 11%, 28%, 17%, and 35% compared to Hybrid, EPSO, PSO, GA, ILP, and HBF, respectively. This leads to monthly energy savings of up to 501kWh. The monthly cost savings will be €201.4 euro, and CO_2 Savings will be 460.92 lbs CO_2 /month. The proposed algorithm consistently delivers optimal solutions with superior exploration and exploitation capabilities. Further, we observe fewer VM migrations and SLA violations in all cases compared to other approaches. On the negative side, Mahalanobis distance does not properly identify the groups if there are many outliers. This decreases the algorithm exploration capability in a few cases.

In our future work, we plan to include other factors, such as cooling, along with advanced machine learning methods to predict resource utilization in a real cloud environment to make resource allocation energy-efficient. Further, we will use reinforcement learning algorithms for real-time feedback and changing conditions. This will help in the proactive placement and dynamic adjustment of VMs. The algorithms should consider the use of renewable energy sources and reduce the carbon footprint of data centers. CloudSim offers six distinct power models, namely, Linear, Cubic, Spec Benchmarks, Square, and Square root. However, most of these models are based on CPU utilization, static, and maximum power of the servers. With better power models considering other relevant factors, one might obtain better overall energy savings.

Acknowledgements
The work presented is supported by the 'NADIKI' project funded by the German Federal Ministry for the Environment, Nature Conservation, Nuclear Safety and Consumer Protection (67KI32001).

Author contributions
DV proposed the IBO algorithm and analyzed the data. MA made great contributions to manuscript preparation and constructive discussion. SG helped with the constructive discussion. All authors read and approved the final manuscript.

Funding
The author(s) received no financial support for the research and/or publication of this article.

Data availability
The data that has been used in this paper is publicly available. We will share the research code based on any reasonable request.

Declarations

Ethics approval and consent to participate
Not Applicable

Consent for publication
Not Applicable

Competing interests
The authors declare that they have no known competing financial interests.

Received: 27 December 2023 Accepted: 24 July 2024
Published online: 15 October 2024

References

Abdul Razaak MP, Ansari GA (2022) A review on virtual machine placement with ACO in cloud computing. In: Proceedings of International Conference on Innovative Computing and Communications: Proceedings of ICICC 2022, Springer 2, pp. 87-98
Abdullahi M, Asri NM, Dishing SI, Abdulhamid SM (2022) An adaptive symbiotic organisms search for constrained task scheduling in cloud computing. J Ambient Intell Humaniz Comput 14:1–12
Abualigah L, Alkhrabsheh M (2022) Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing. J Supercomput 78:740–765
Ajmera K, Tewari TK (2023) Energy-efficient virtual machine scheduling in IaaS cloud environment using energy-aware green-particle swarm optimization. Int J Inf Technol 15:1–9
Al-Dulaimy A, Itani W, Zantout R, Zekri A (2018) Type-aware virtual machine management for energy efficient cloud data centers. Sustain Comput Inf Syst 19:185–203
Andrae AS, Edler T (2015) On global electricity usage of communication technology: trends to 2030. Challenges 6(1):117–157
Assudani PJ, Balakrishnan P (2022) A novel bio-inspired approach for VM load balancing and efficient resource management in cloud. Int J Ad Hoc Ubiquit Comput 40(1–3):214–224
Azizi S, Mohammad S, Jemal A, Rajkumar B (2020) A greedy randomized algorithm for virtual machine placement in cloud data centers. IEEE Syst J 15(2):2561–2582
Babar M, Karamti H, Alzamzami O, Khan A, Nawaz M (2022) A bacterial foraging based smart offloading for IoT sensors in edge computing. Comput Electr Eng 102:108123
Beloglazov A, Buyya R (2012) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr Comput Pract Exp 24:1397–1420
Beloglazov A, Rajkumar B (2012) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr Comput Pract Exp 24(13):81398–1420
Beloglazov A, Jemal A, Rajkumar B (2012) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Futur Gener Comput Syst 28(5):755–768
Brian S, CFA, Derek R, Brendan B, Carly C, Alberto D, Gandol (2024) AI/Data Centers' Global Power Surge and the Sustainability Impact. The Goldman Sachs Group Inc., pp 4–15
Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41:23–50
Chen G, Jie Q, Zhizhong Z, Zhi S (2019) Multi-objective optimal power flow based on hybrid firefly-bat algorithm and constraints-prior object-fuzzy sorting strategy. IEEE Access 7:139726–139745
Chen G, Qian J, Zhang Z, Sun Z (2019) Multi-objective optimal power flow based on hybrid firefly-bat algorithm and constraints prior object-fuzzy sorting strategy. IEEE Access 7(1):139726–139745
Chen R, Chen X, Yang C (2022) Using a task dependency job-scheduling method to make energy savings in a cloud computing environment. J Supercomput 78(3):4550–4573
Corradi A, Fanelli M, Foschini L (2014) VM consolidation: a real case based on OpenStack cloud. Futur Gener Comput Syst 32:118–127
da Silva RA, da Fonseca NL (2016) Topology-aware virtual machine placement in data centers. J Grid Comput 14:75–90
Dashti SE, Rahmani AM (2016) Dynamic VMs placement for energy efficiency by PSO in cloud computing. J Exp Theo Artif Intell 28(1–2):97–112
De Maesschalck R, Jouan-Rimbaud D, Massart DL (2000) The Mahalanobis distance. Chemom Intell Lab Syst 50:1–18
Farzai S, Shirvani MH, Rabbani M (2020) Multi-objective communication-aware optimization for virtual machine placement in cloud data centers. Sustain Comput Inf Syst 28:100374
Fu X, Zhou C (2015) Virtual machine selection and placement for dynamic consolidation in cloud computing environment. Front Comp Sci 9:322–330
Ghetas M (2021) A multi-objective monarch butterfly algorithm for virtual machine placement in cloud computing. Neural Comput Appl. 1–15
Gomathi B, SaravanaBalaji B, Krishna Kumar V, Abouhawwash M, Aljahdali S, Masud M, Kuchuk N (2020) Multi-objective optimization of energy aware virtual machine placement in cloud data center. Intell Automat Soft Comput 33(3):1771–1785
Gonzalez C, Tang B (2020) FT-VMP: fault-tolerant virtual machine placement in cloud data centers. In: 2020 29th International Conference on Computer Communications and Networks (ICCCN), IEEE. 1-9
Gupta MK, Jain A, Amgoth T (2018) Power and resource-aware virtual machine placement for IAAS cloud. Sustain Comput Inf Syst 19:52–60

Hao Y, Cao J, Ma T, Ji S (2019) Adaptive energy-aware scheduling method in a meteorological cloud. Futur Gener Comput Syst 101:1142–1157

He H, Zhao Y, Pang S (2020) Stochastic modeling and performance analysis of energy-aware cloud data center based on dynamic scalable stochastic Petri net. Comput Inf 39:28–50

Hobaei-Arani M, Shamsi M, Rahmanian AA (2017) An efficient approach for improving virtual machine placement in cloud computing environment. J Exp Theo Artif Intell 29(6):81149–1171

Homsi S, Liu S, Chaparro-Baquero GA, Bai O, Ren S, Quan G (2016) Workload consolidation for cloud data centers with guaranteed QoS using request reneging. IEEE Trans Parallel Distrib Syst 28:2103–2116

Ibrahim A, Noshy M, Ali HA, Badawy M (2020) PAPSO: a power-aware VM placement technique based on particle swarm optimization. IEEE Access 8:81747–81764

Jangiti S et al (2020) Hybrid best-fit heuristic for energy efficient virtual machine placement in cloud data centers. EAI Endorsed Transactions on Energy Web 7

Jatoth C, Gangadharan G, Fiore U (2019) Optimal fitness aware cloud service composition using modified invasive weed optimization. Swarm Evol Comput 44:1073–1091

Javadi-Moghaddam S-M, Zahra D (2023) Virtual machine placement in cloud using artificial bee colony and imperialist competitive algorithm. Int J Electr Comput Eng 13(4):4743–4751

Keller G, Tighe M, Lutfiyya H, Bauer M (2012) An analysis of first fit heuristics for the virtual machine relocation problem. In: Proceedings of 2012 workshop on systems virtualiztion management, IEEE pp 406–413

Kimmons B (2015) Comparing the Efficiency of Heterogeneous and Homogeneous Data Center Workloads. Master's Thesis, Georgia Southern University 1249

Kołodziej J, Khan SU, Wang L, Zomaya AY (2015) Energy efficient genetic-based schedulers in computational grids. Concurr Comput Pract Exp 27:809–829

Kumar KP, Ragunathan T, Vasumathi D (2020) Virtual machine consolidation using modified lion optimization algorithm to improve energy efficiency in cloud computing environment. Int J Adv Res Eng Technol 11(12):1593–1608

Kumar D, Raza Z (2015) A PSO based VM resource scheduling model for cloud computing. In: Proceedings of IEEE International Conference on Computational Intelligence & Communication Technology (CICT), IEEE. 213–219

Liang B, Dong X, Zhang X (2019) A heuristic virtual machine scheduling algorithm in cloud data center. In: 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), IEEE. pp. 180–184

Mahalanobis PC (1936) On the generalized distance in statistics. In: Proceedings of the statistical laboratory, National Institute of Science of India

Mangalampalli S, Swain SK, Mangalampalli VK (2022) Multi objective task scheduling in cloud computing using cat swarm optimization algorithm. Arab J Sci Eng 2:1821–1830

Mann ZÁ (2016) Multicore-aware virtual machine placement in cloud data centers. IEEE Trans Comput 65:3357–3369

Mann ZA, Máté S (2017) Which is the best algorithm for virtual machine placement optimization? Concurr Comput Pract Exp 29(10):e4083

McLachlan GJ (1999) Mahalanobis distance. Resonance 4:20–26

Nabavi SS, Gill SS, Xu M, Masdari M, Garraghan P (2022) TRACTOR: traffic-aware and power-efficient virtual machine placement in edge-cloud data centers using artificial bee colony optimization. Int J Commun Syst 35(1):e4747

Najafizadegan N, Nazemi E, Khajehvand V (2021) An autonomous model for self-optimizing virtual machine selection by learning automata in cloud environment. Softw Pract Exp 51:1352–1386

Ndayikengurukiye A, Ez-Zahout A, Fouzia O (2024) Optimizing virtual machines placement in a heterogeneous cloud data center system. Int J Comput Netw Appl 11(1):1–12

Palmieri F, Fiore U, Ricciardi S, Castiglione A (2016) GRASP-based resource re-optimization for effective big data access in federated clouds. Futur Gener Comput Syst 54:168–179

Pandey HM, Ankit C, Deepti M (2012) A comparative review of approaches to prevent premature convergence in GA. Appl Soft Comput 24:1047–1077

Parida BR, Amiya K, Bibudhendu R, Chhabi RP, Hitesh P, Rajkumar M, Buyya (2024) SSEPC cloud: Carbon footprint aware power efficient virtual machine placement in cloud milieu. Computer Science and Information Systems 00:18–18

Potu N, Jatoth C, Parvataneni P (2021) Optimizing resource scheduling based on extended particle swarm optimization in fog computing environments. Concurr Comput Pract Exp 33:e6163

Pradhan A, Bisoy SK (2020) A novel load balancing technique for cloud computing platform based on PSO. J King Saud Univ Comput Inf Sci

Rao GSVRK, Gangadharan GR, Vemula DR (2021) System and method for determining optimal solution in a swarm of solutions using swarm intelligence. US Patent 11144832

Reddy VD, Gangadharan G, Rao GSV (2019) Energy-aware virtual machine allocation and selection in cloud data centers. Soft Comput 23:1917–1932

Reddy VD, Gangadharan G, Rao G, Aiello M (2020) Energy-efficient resource allocation in data centers using a hybrid evolutionary algorithm. Machine learning for intelligent decision science. Springer, Berlin, pp 71–92

Reddy KHK, Luhach AK, Pradhan B, Dash JK, Roy DS (2020a) A genetic algorithm for energy efficient fog layer resource management in context-aware smart cities. Sustain Cities Soc 63:102428

Sohrabi MK, Ghods V, Fard SYZ (2018) A novel virtual machine selection policy for virtual machine consolidation. In: 2018 6th International Symposium on Computational and Business Intelligence, IEEE. pp. 28–32

Vijaya C, Srinivasan P (2024) Multi-objective meta-heuristic technique for energy efficient virtual machine placement in cloud data centers. Informatica 48(6):1–18

Vilaplana J, Mateo J, Teixidó I, Solsona F, Giné F, Roig C (2015) An SLA and power-saving scheduling consolidation strategy for shared and heterogeneous clouds. J Supercomput 71:1817–1832

Wu CM, Chang RS, Chan HY (2014) A green energy-efficient scheduling algorithm using the DVFS technique for cloud data centers. Futur Gener Comput Syst 37:141–147

Xu Z, Lei Z, Shuaikui T, Mengyang H, Sijin Y, Yu S, Ling M (2020) Energy-driven virtual network embedding algorithm based on enhanced bacterial foraging optimization. IEEE Access 8:76069–76081

Yadav R, Zhang W, Li K, Liu C, Shafiq M, Karn NK (2020) An adaptive heuristic for managing energy consumption and overloaded hosts in a cloud data center. Wireless Netw 26:1905–1919

Zhao H, Wang J, Liu F, Wang Q, Zhang W, Zheng Q (2018) Power-aware and performance-guaranteed virtual machine placement in the cloud. IEEE Trans Parallel Distrib Syst 29:1385–1400

Zhao H, Nanzhi F, Jianhua L, Guobin Z, Jing W, Quan W, Bo W (2023) VM performance-aware virtual machine migration method based on ant colony optimization in cloud environment. J Parallel Distrib Comput 176:17–27

Zimmerman Donald W (1997) Teacher's corner: a note on interpretation of the paired-samples t test. J Educ Behav Statis 22(3):349–360