The Journal of Supercomputing (2025) 81:237
https://doi.org/10.1007/s11227-024-06734-1

An intelligent virtual machine allocation optimization model for energy-efficient and reliable cloud environment

Smruti Rekha Swain[1] · Anshu Parashar[1] · Ashutosh Kumar Singh[1] · Chung Nan Lee[2]

Accepted: 15 November 2024

Abstract

The exponential growth of cloud computing has brought increased attention to energy efficiency in data centers. However, fluctuating resource demands and fixed virtual machine (VM) sizes lead to excessive energy consumption, inefficient resource utilization, and load imbalances. While dynamic VM consolidation mitigates these issues by reducing the number of active physical machines (PM), frequent consolidation can compromise system reliability, as VMs may be assigned to unreliable PMs. An effective resource management strategy is therefore essential for balancing energy efficiency and reliability in cloud data centers. This paper presents a novel resource prediction-based VM allocation approach that significantly reduces energy consumption while enhancing system reliability. The core innovation lies in optimizing a feed-forward neural network using the self-adaptive differential evolution algorithm, which integrates multi-dimensional learning and global exploration. Unlike traditional gradient descent algorithms, this method searches for the global best solution, offering more accurate and robust predictions of future resource usage. These proactive resource estimations enable fault-tolerant and reliable VM management, preventing system failures and improving overall performance. Evaluated with the Google cluster dataset, the proposed model outperforms existing methods, delivering remarkable reductions in power consumption (up to 44.81%) and the number of active PMs (up to 64.73%). Additionally, the system's reliability improves by 65.25%, demonstrating the effectiveness of the approach in achieving energy-efficient and fault-tolerant cloud data center management.

Keywords Cloud computing · Virtual machine · Physical machine · Energy consumption · System reliability · Neural network

Anshu Parashar, Ashutosh Kumar Singh, and Chung Nan Lee contributed equally to this work.

Extended author information available on the last page of the article

Published online: 04 December 2024

1 Introduction

Cloud computing (CC) has become a vital distributed computing model, providing users with on-demand, scalable, and cost-effective computing resources [1]. Its flexibility, scalability, and low-latency characteristics make it a preferred platform for managing large-scale data processing. However, as the demand for cloud services surges, large-scale virtualized cloud data centers are experiencing a significant increase in energy consumption, resulting in high operational costs and environmental impact [2, 3]. Notably, inefficient resource utilization further exacerbates energy waste [4]. In addition to energy efficiency, ensuring reliable performance and maintaining quality of service (QoS) parameters, such as response time and system availability, are essential requirements, formalized through service-level agreements (SLAs) between service providers and users [5, 6].

Managing energy efficiency and system reliability in cloud data centers is a challenging bi-objective problem. While virtual machine consolidation has emerged as an effective approach to reducing power consumption, it introduces significant trade-offs that require careful consideration. The following challenges highlight the complexity of this issue:

- *Energy-Performance Trade-offs*: Reducing the number of active physical machines (PMs) through VM consolidation reduces energy consumption but may degrade system performance and lead to higher SLA violations due to increased VM migrations [7].
- *VM Consolidation and Reliability*: High consolidation frequencies increase the load on certain PMs, leading to greater risks of hardware failure, which negatively impacts overall system reliability [8].
- *Power State Transitions*: Switching between power states, such as dynamic voltage frequency scaling (DVFS), to save energy can waste power and reduce server reliability during frequent transitions [9].
- *Complexity of Resource Allocation*: Predicting future resource usage to optimize VM placement requires sophisticated algorithms that account for both energy efficiency and reliability while minimizing overhead and resource wastage [10].

The failure to address these challenges together can lead to several detrimental impacts:

- *Increased Operational Costs*: Excessive power consumption due to inefficient resource utilization directly increases operational costs for cloud providers.
- *Degraded QoS*: Frequent VM migrations and server overloads can increase response times and downtime, leading to SLA violations and reduced customer satisfaction.
- *Reduced System Reliability*: Overloading PMs and frequent power state transitions can compromise the reliability of the data center infrastructure, increasing the likelihood of hardware failures and system downtime.

- *Environmental Impact*: Excessive energy consumption contributes to environmental degradation due to increased carbon emissions from data center operations.

Existing solutions in cloud computing [2], including VM consolidation [11], dynamic voltage frequency scaling (DVFS), and predictive resource allocation [10], have made strides in addressing energy efficiency and reliability [8], yet each presents notable limitations. VM consolidation, while effective in reducing the number of active physical machines (PMs) to save energy, often results in increased VM migrations, which can degrade performance and violate SLAs during high-demand periods. DVFS dynamically adjusts power states to conserve energy, but frequent state transitions can increase power consumption and lead to hardware wear, thus compromising long-term reliability. Predictive resource allocation methods, utilizing machine learning and heuristic approaches, aim to improve VM placement by forecasting resource needs; however, many lack adaptability for rapid demand shifts or do not fully integrate reliability considerations, leading to potential overloads and system instability. These limitations underscore the need for a more cohesive framework that can effectively balance both energy efficiency and system reliability under dynamic conditions. Addressing these issues is essential to achieve sustainable and reliable cloud operations:

- *Energy Efficiency*: It is critical to develop VM placement strategies that reduce the number of active PMs and minimize unnecessary power state transitions, without sacrificing system performance [9].
- *Reliability Improvement*: Ensuring reliable VM allocation by minimizing excessive migrations and avoiding the overloading of PMs is vital for maintaining system integrity and meeting SLA requirements [7].
- *Balanced Optimization*: A proactive approach that optimizes both energy efficiency and system reliability, while forecasting resource usage, is crucial for the future of cloud data center management [12].

This study proposes an energy-efficient and reliable virtual machine placement (EER-VMP) model, which aims to address these challenges. The EER-VMP incorporates a neural network-based resource prediction unit to forecast future resource usage of VMs. This proactive approach allows for optimal VM placement, minimizing the number of active PMs and reducing energy consumption while enhancing system reliability by preventing unnecessary VM migrations and avoiding overloaded PMs. The proposed model is evaluated using the Google cluster dataset (GCD) and demonstrates significant improvements in both energy efficiency and system reliability. The key contribution is threefold:

- A novel energy-efficient and reliable virtual machine placement (EER-VMP) model meticulously designed to optimize VM assignments, ensuring placement on energy-efficient and reliable physical machines.
- The proposed EER-VMP model introduces an advanced neural network-based resource predictor, optimized through the self-adaptive differential evolution

(SADE) algorithm, significantly enhancing resource usage predictions' accuracy. By leveraging SADE's adaptive optimization capabilities, the neural network achieves a higher level of precision in forecasting future resource demands. This accuracy allows for more feasible and targeted VM placements, ensuring that resources are allocated based on precise demand predictions.

- Conducted a comprehensive evaluation of the proposed model using the Google cluster dataset, demonstrating superior performance compared to state-of-the-art techniques in critical metrics such as power consumption, overall system reliability, and reducing active PMs.

The remainder of this paper is organized as follows: Literature study on VMP discussed in Sect. 2. Section 3 presents the overview of the proposed EER-VMP model and different resource capacity constraints. A detailed description of the resource prediction unit and SADE learning algorithm is discussed in Sect. 4. Section 5 explains the functional design and complexity analysis of the proposed EER-VMP model. The performance evaluation is presented in Sect. 6, followed by the conclusion of the proposed work in Sect. 7. A list of notations with their descriptive meaning used in this work is stated in Table 1.

2 Related work

Cloud computing has sparked significant research into efficient virtual machine placement (VMP) strategies to optimize energy consumption and resource utilization within data centers [13]. In work by Saxena et al. [14], a secure and multi-objective virtual machine placement (SM-VMP) framework is proposed, combining non-dominated sorting-based genetic algorithms (NSGA-II) and whale optimization algorithms (WOA) to enhance resource utilization while minimizing power consumption

Table 1 Symbols with their meanings

S	Physical machines	S_i^{Mem}	Memory capacity of ith PM
V	Virtual machines	S_i^{CPU}	CPU capacity of ith PM
r	Cloud users	V_j^{CPU}	CPU requested by jth VM
PW	Power consumption	p	Total number of PMs
$\text{Re}_{\text{system}}$	Overall system reliability	q	Total number of VMs
$\text{PW}_i^{\max}$	Maximum power	PW_i	Power consumption at server i
$\text{PW}_i^{\min}$	Minimum power	RU_{dc}^R	Total resource utilization of data center
$\text{PW}_i^{\text{idle}}$	Idle state power	z_{ji}	Mapping of jth VM on ith PM
PW_{dc}^R	Total power of data center	S_i^{RAM}	RAM capacity of ith PM
M	Total number of users	V_j^{RAM}	RAM requested by jth VM
β	Sensitivity factor	V_j^{Mem}	Memory requested by jth VM
λ_{ji}	Hazard rate of jth VM on ith PM		

and communication costs. Sharma et al. [15] introduced a hybrid genetic algorithm and particle swarm optimization (HGAPSO) for multi-objective VM placement to reduce energy consumption and minimize resource wastage and SLA violations. This method employs a bit value to denote the presence of VMs but lacks adaptability in heterogeneous environments. Singh et al. [16] proposed the secure and energy-aware load-balancing (SEA-LB) framework, utilizing a genetic algorithm (GA) to enhance resource usage while addressing power consumption and minimizing conflicting PMs. However, premature convergence remains a significant limitation of this approach. Tseng et al. [17] developed a prediction-based VM placement method that forecasts future workloads using historical data and the GA concept, focusing on optimal allocations based on predictions. Meanwhile, Han et al. [18] presented a security-aware multi-objective optimization (SMOOP) technique, emphasizing resource utilization, security, and network traffic but relying heavily on weighted sums and modified GA approaches. Basset et al. [19] proposed a bandwidth-efficient VMP technique, integrating an improved levy-based WOA with a best-fit approach for static resource allocation inhomogeneous configurations. The work in [20] offers a detailed survey and comparative analysis of machine learning-based workload prediction models, categorizing them into five types based on design and methodology. Each model is reviewed and benchmarked against three cloud workload datasets to evaluate effectiveness through key performance indicators. Wang et al. [21] present a resource scheduling framework for optimizing energy consumption and QoS in cloud data centers. The framework minimizes energy use while ensuring QoS by using an energy consumption model and optimization function. It leverages Lyapunov stability theory to address energy efficiency in response to task arrival fluctuations. Conversely, Saxena et al. [22] suggested a resource management framework to improve data center efficiency by balancing loads, minimizing the number of active servers, and reducing VM migrations. They developed an online resource prediction system to prevent SLA violations, while multi-objective VM placement and migration algorithms were designed to reduce network traffic and power consumption. Peake et al. [23] introduced an advanced virtual machine placement (VMP) algorithm based on parallel ant colony optimization (PACO), which harnessed parallelization and modern processor technologies to enhance cloud data center efficiency. This approach matched or surpassed the solution quality of other nature-inspired methods and achieved a remarkable speed-up of up to 2002x compared to recent serial algorithms. Hiremath et al. [5] addressed the interoperability challenge in cloud computing by proposing an energy-efficient data migration approach using a hybrid optimized deep learning framework. The Taylor Lion-based poor and rich optimization (Taylor Lion-based PRO) enhances load balancing and migration efficiency, achieving notable performance metrics for load, resource capacity, and energy consumption, thereby improving the overall effectiveness of data migration in heterogeneous cloud environments. Reliability considerations are also addressed in various studies. Xuejie et al. [24] introduced a hybrid method-based reliability evaluation (HMRE) model utilizing mean time to failure (MTTF) and continuous-time Markov chain (CTMC) metrics. However, it primarily focused on software failures. Sharma et al. [25] developed a failure-aware and power-efficient VM placement technique, employing exponential smoothing for fault tolerance but requiring

further refinement in dynamic environments. Zhou et al. [26] proposed the cloud service reliability enhancement (CSRE) mechanism to improve resource utilization via service checkpoints but did not adequately address VM workload variability. Azimzadeh and Biabani [27] introduced a multi-objective resource scheduling (MORS) technique, balancing system reliability and execution time for high-performance computing (HPC) workloads, but often fell short in providing optimal global solutions for multi-target problems [28]. A more recent multi-objective VMP technique by Sayadnavard et al. [29] utilized the Artificial Bee Colony-based Epsilon-dominance level (ϵ-MOABC) mechanism, effectively addressing dynamic VM consolidation issues. However, it did not account for resource overcommitment and bandwidth constraints. Despite these advancements, several gaps remain in the existing research:

- Many approaches focus on either energy efficiency or reliability, failing to address both concurrently.
- Several models suffer from premature convergence or adaptability issues in heterogeneous environments, limiting their practical application.
- The existing methods often overlook critical factors like resource overcommitment and dynamic workload variations, leading to suboptimal performance.

The proposed energy-efficient and reliable virtual machine placement (EER-VMP) model addresses these gaps by integrating a predictive neural network optimized through the self-adaptive differential evolution algorithm. This novel approach allows for proactive resource usage estimation, enabling dynamic VM allocation that balances energy efficiency and reliability. By considering both power consumption and system performance, the EER-VMP model enhances overall data center efficiency while mitigating risks associated with SLA violations and resource wastage.

3 EER-VMP model

A data center infrastructure comprises p number of servers, or PMs represented as $\{S_1, S_2, \cdots, S_p\} \in S$, q number of VMs described as $\{V_1, V_2, \cdots, V_q\} \in V$ and r number of users denoted as $\{U_1, U_2, \cdots, U_r\} \in U$ is considered in the proposed model as illustrated in Fig. 1. Both PMs S and VMs V have heterogeneous configurations. Each user is associated with an application and has purchased different types of VMs to execute their applications on energy-efficient and reliable PMs. The VM hypervisor layer allows assignment of VMs $\{V_{2y}, V_{m2}, \cdots, V_{mz}\} \in V$on server S_1 by developing an isolation layer among them. Similarly the VMs $\{V_{m1}, V_{11}, \cdots, V_{1x}\} \in V$ and $\{V_{12}, V_{21}, \cdots, V_{22}\} \in V$ are deployed on servers S_2 and S_p respectively. Every VM is associated with a specific guest operating system (OS) with a user-specified resource capacity for executing applications. Every server incorporates a pool of resources such as computing, storage, and bandwidth; a reliability checker and power analyzer are also accommodated within each server to analyze the respective PMs' reliability score and power consumption. The proposed approach is intended to assign VMs

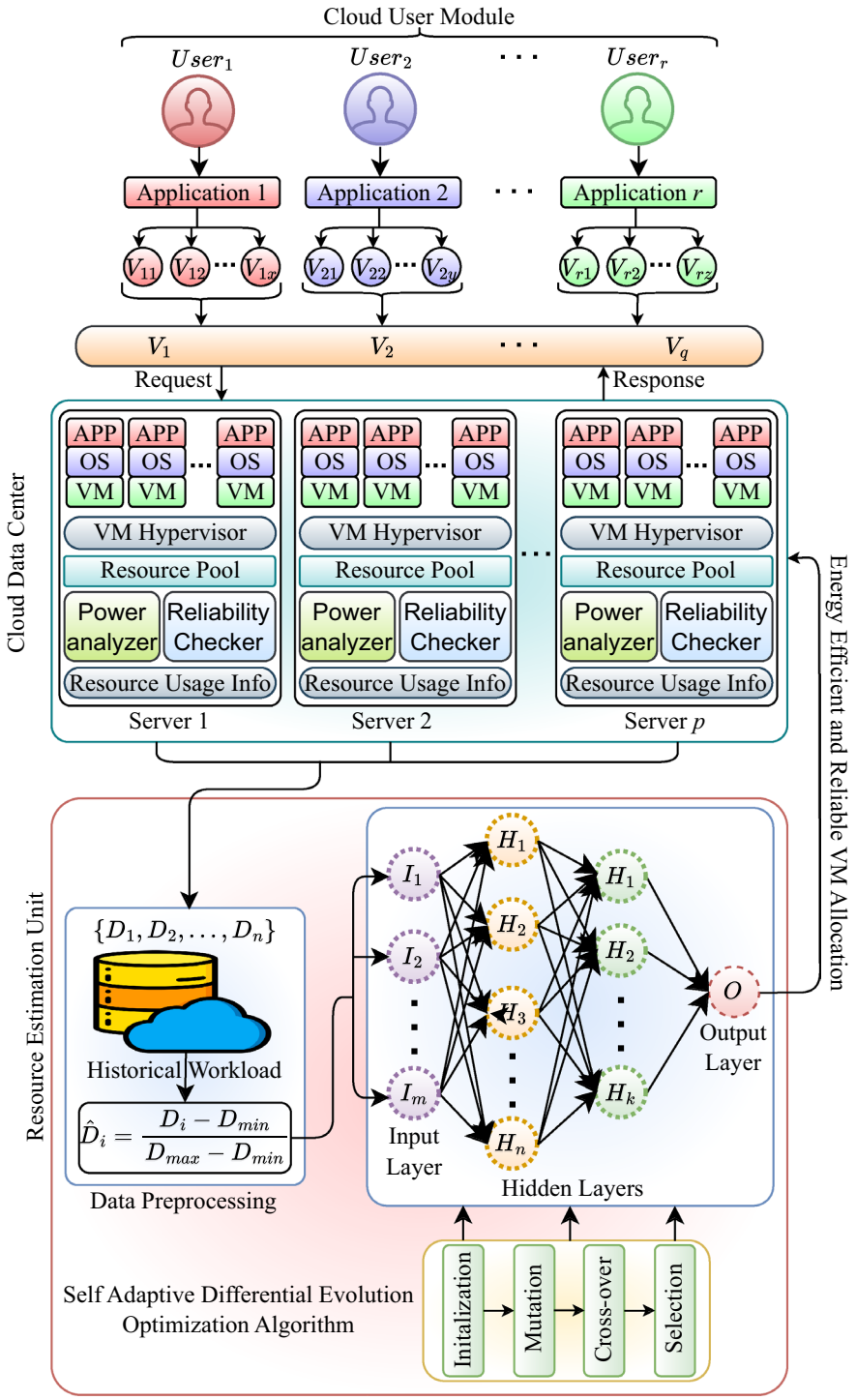

Fig. 1 EER-VMP model

in a reliable and energy-efficient PM. Hence, the overall reliability of the system increases. The following objectives are designed for VM placement:

3.1 System reliability

Reliability is a performance metric that measures a system's ability to run the system correctly in a given situation for a given time interval. Six attributes are used to calculate cloud system reliability: accuracy, availability, defective transaction, failure recovery, service latency, and fault tolerance. Let the reliability of a particular attribute be denoted by r_a, where a is $1 \leq a \leq 6$. The proposed mechanism quantified system reliability, mainly considering resource failures, i.e., power usage, which is the hard-wired metric of computing resources. VMs placed on a server node will fail only if the server fails. λ_{ji} denotes the failure rate or hazard rate of V_j running on the server S_i having resource utilization RU_i. λ_{ji} computed by applying in Eq. (1).

$$\lambda_{ji} = \lambda_{\max_j} \times RU_i^{\beta} \tag{1}$$

where $\lambda_{\max_j}$ is the server failure rate S_i at its highest utilization level. The sensitivity factor β (> 0) indicates the sensitivity of the failure rate toward utilization. When $\beta = 1$, it means that there is a linear relationship exists between λ_{ji} and RU_i. All the VMs on servers S_i share the same hazard rate. Maximum $\lambda_{\max_i}$ of a server S_i can be calculated as the inverse of the mean time between failures (MTBF) and given in Eq. (2).

$$\lambda_{\max_i} = 1/\mathrm{MTBF}_i \tag{2}$$

At any utilization level, the failure rate is constant. Using the hazard rate λ_{ji}, the probability with which a VM, V_j will complete the execution of all the running jobs is the same as the probability with which the most extended task, $l_{\max_j}$ running on V_j will finish the execution before the occurrence of a failure. It is stated as the reliability of a VM. As mentioned earlier, while a server fails, all the VMs assigned to the server also fail. The server and VMs share a linear relationship among them. Hence, the reliability of a server S_i is computed as the product of the reliability of all the VMs over it, as stated in the following Eq. (3).

$$\mathrm{Re}_{\mathrm{system}} = \prod_{j=1}^{q} \exp^{-\lambda_{ij}} \times l_{\max_j} \tag{3}$$

3.2 Power consumption

Since the main objective of this work is power management, it is also one of the evaluation parameters described in this section. Data center power consumption is because of the memory, storage, CPU, and the number of active network components in a running state. The high energy consumption of these components leads to high operating costs. This work considers all PMs mostly based on the built-in DVFS power-saving

strategy. A CPU has two different states: busy and idle. Power consumption depends on processing applications and CPU utilization in a busy state. While in sleep, the CPU performs minimal work in minimal clock cycles. Some components of the CPU are not in a working state. So, PW_i and PW_{dc} are the power consumption of the ith PM and the total power consumption of the data center as stated in Eqs. (4) and (5), respectively.

$$PW_i = \left(\left[PW_i^{max} - PW_i^{min} \right] \times RU_i + PW_i^{idle} \right) \tag{4}$$

$$PW_{dc} = \int_{t1}^{t2} \left(\sum_{i=1}^{n} PW_i \right) dt \tag{5}$$

where $RU_i \in [0,1]$ represents the resource utilization of ith PM.

3.3 Constraints

The primary constraints must be met during VM placement and are listed in equations (6)–(8).

$$C1 : \sum_{j=1}^{q} V_j^{CPU} \times z_{ji} \leq S_i^{CPU} \quad \forall i \in 1, 2, \ldots, p \tag{6}$$

$$C2 : \sum_{j=1}^{q} V_j^{RAM} \times z_{ji} \leq S_i^{RAM} \quad \forall i \in 1, 2, \ldots, p \tag{7}$$

$$C3 : \sum_{j=1}^{q} V_j^{Mem} \times z_{ji} \leq S_i^{Mem} \quad \forall i \in 1, 2, \ldots, p \tag{8}$$

Where z_{ji} is a binary variable, and its value is one if jth VM is placed on ith PM. Otherwise, it will have a value of zero. V_j^{RAM}, V_j^{CPU}, and V_j^{Mem} are RAM, CPU capacity (in MIPS), and secondary memory requirement of jth VM. Similarly, S_i^{RAM}, S_i^{CPU}, S_i^{Mem} are RAM, CPU capacity (in MIPS), and secondary memory requirement of ith PM. The proposed model gives an energy-efficient and reliable VM placement method with in-time execution of user applications. The formulation of the problem is given in Eqs. (9)–(10).

$$\chi = \sum_{i=j=1}^{p,q} z_{ji} \tag{9}$$

$$\begin{aligned} &\text{Minimize } f(P_{dc}) \\ &\text{Maximize} f(\text{Re}_{\text{system}}) \\ &\text{subject to } \{C1, C2, C3\} \end{aligned} \tag{10}$$

4 Resource estimation unit

The evolutionary neural network (NN) module forecasts the future resource usage of VMs. The machine learning capability and computational ability of NN can enhance the accuracy of the prediction unit. The output of the prediction unit and the actual data are used for effective resource management within the data center. A NN is represented as m-n-p, where m, n, and p denote the number of neurons in the input, hidden, and output layers. ω represents the NN weights generated randomly in the range of [0,1]. The NN collects historical resource usage information that is normalized to create and provide m input values such as $\{D_1, D_2, \ldots, D_m\}$ into the input layer. The training of NN starts with the randomly generated N different neural networks, each of size $L = (m+1) \times n + (n \times p) = n(m+p+1) \implies n(m+2)$ as $p = 1$. An additional input is added to the input neuron as a bias value. The prediction technique consists of training, testing, and predicting. During the learning stage, historical data are preprocessed by subsequent attribute extraction and normalization. The entire data are then divided into training and test samples. The most appropriate patterns are extracted from training samples, and the prediction unit finds correlations between them. Data validation is applied to optimize the model by calculating the model accuracy using the error function root mean squared error (RMSE). Once the desired accuracy is achieved, the prediction unit enters the testing phase. Otherwise, the model will be retrained. During the test phase, test patterns are utilized to calculate the trained model accuracy and produce the final predictive model. Finally, the prediction unit is deployed for cloud workload prediction during the prediction phase.

4.1 Data preprocessing

The preprocessing of data extracts resource requests from different *V*s and aggregates them over a fixed time interval. Equation (11) is used to normalize the input data in the range (0,1) using a data aggregation process.

$$\hat{D}_i = \frac{D_i - D_{\min}}{D_{\max} - D_{\min}} \tag{11}$$

$D_{\max}$ and $D_{\min}$ are the maximum and minimum values obtained from the dataset. $\hat{D}$ represents the normalized data, which is a set of all normalized data values$\{\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_m\}$. This normalized value is treated as input into the input layer of NN. These normalized one-dimensional values are arranged as a multi-dimensional input and output matrix represented as X_{input} and Y_{output}, respectively, as mentioned in the following Eq. (12).

$$X_{input} = \begin{bmatrix} X_1 & X_2 & \cdots & X_m \\ X_2 & X_3 & \cdots & X_{m+1} \\ \vdots & \vdots & \cdots & \vdots \\ X_n & X_{n+1} & \cdots & X_{m+n-1} \end{bmatrix} Y_{output} = \begin{bmatrix} X_{m+1} \\ X_{m+2} \\ \vdots \\ X_{m+n} \end{bmatrix} \tag{12}$$

The prediction unit takes X_{input} as the input vector and extracts the appropriate pattern. This model analyzes previous m workload values to predict the workload (Y_{output}) at $m+1$th time instance at the data center.

4.2 Evolutionary neural network architectural optimization

The functionality of the resource prediction unit is shown in Fig. 1. Neurons in one layer are associated with each neuron in the next layer with the help of weights w_1, w_2, ..., w_n. The neurons at input layer receive input as actual-valued data $\hat{D}_1$, $\hat{D}_2$, ..., $\hat{D}_m$. The bias value is generated in the [0, 1] range, denoted as b. The input layer is linked to the hidden layer through neural weights and expressed as w_{11}^{IN}, w_{12}^{IN}, ..., w_{mn}^{IN}. The sigmoid activation function shown in Eq. (13) maps input to output in the range [0,1]. Then, the predicted values are scaled in the range [0,1] to compare with actual output values. Lastly, the accuracy and performance of the prediction unit are evaluated using the RMSE score, as stated in Eq. (14).

$$\text{sigmoid}(\theta) = \frac{1}{1+\exp^{\theta}} \tag{13}$$

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(Y_{\text{actual}} - Y_{\text{predicted}})^2} \tag{14}$$

where Y_{actual} and $Y_{\text{predicted}}$ are the actual and predicted workloads, respectively.

A *self-adaptive differential evolution* (SADE) learning mechanism is used for the dynamic and adaptive optimization of network weights. The SADE algorithm consists of four basic operations: initialization, mutation, recombination (or crossover), and selection.

4.2.1 Initialization

A set of N network vectors is generated randomly such that each vector represents a set of weights in terms of a vector of size $(m+1)\times n + (n\times p) = n(m+p+1) \implies n(m+2)$, where $p = 1$ for output layer and bias one is added at input layer. As mentioned in Table 4, the learning process involves initializing various parameters, such as the number of epochs, mutation rate, and crossover rate. The mutation-selection probability (*mps*) parameter is generated in each iteration to choose one of four optional mutation strategies. The operations such as mutation and crossover are performed to obtain offspring for the next subsequent iteration. Optimal solutions are selected based on their fitness value. After a fixed time interval called the 'learning period,' the mutation rate and crossover values are updated.

4.2.2 Mutation strategy

To explore the search space in multiple directions and generate new solutions with better fitness values, mutation and crossover operators are applied during the iterative optimization process. Four different mutation policies are selected for this work, i.e., DE/best/1 (15), DE/ current-to-best/1 (16), DE/ current-to-rand/1 (17), and DE/rand/1 (18). The selection of mutation policy plays a vital role in enhancing the quality of solutions. Each mutation strategy has its own characteristics that specify its specific usage. For example, DE/best/1 (15) and DE/current-to-best/1 (16) strategies use the best individual to generate the vectors that bring extensive exploitation during the evolutionary stages of the algorithm. DE/ current-to-rand/1 (Eq. 17) and DE/rand/1 (18) strategies are used to prevent premature convergence by allowing exploration [30].

$$u_i^j = X_{\text{best}}^j + \mu_i \times \left(X_{r1}^j - X_{r2}^j\right) \tag{15}$$

$$u_i^j = X_i^j + \mu_i \times \left(X_{\text{best}}^j - X_i^j\right) + \mu_i \times \left(X_{r1}^j - X_{r2}^j\right) \tag{16}$$

$$u_i^j = X_i^j + k_i \times \left(X_{r1}^j - X_i^j\right) + \mu_i \times \left(X_{r2}^j - X_{r3}^j\right) \tag{17}$$

$$u_i^j = X_{r3}^j + \mu_i \times \left(X_{r1}^j - X_{r2}^j\right) \tag{18}$$

where u_i^j and X_i^j are ith mutant and current vectors in jth iteration. $r1$, $r2$, and $r3$ are randomly generated mutually different numbers in the range [1, N]. X_{best}^j denotes the best solution. μ_i and k_i are controlling parameter for corresponding vector in ith iteration. Let γ_1, γ_2, γ_3, and γ_4 be the probabilities for selecting the DE/best/1, DE/current-to-best/1, DE/rand/1, and DE/current-to-rand/1 mutation techniques, respectively. Initially, $\gamma_1 = \gamma_2 = \gamma_3 = \gamma_4 = 0.25$, so every technique can get equal selection opportunity. The roulette wheel selection technique [31] is used for assigning probabilities among four different mutation strategies. The *mps* parameter of N random numbers is generated in the range [0,1]. For instance, if $mps_i \le \gamma_1$, then DE/current-to-best/1 is applied. Similarly, if $mps_i > \gamma_1$ and $mps_i \le \gamma_1 + \gamma_2$, then DE/best/1 is utilized and so on. The selection of the mutation scheme is denoted as Υ in Eq. (19).

$$\Upsilon = \begin{cases} \text{DE/ current-to- best/1}, & 0 < \text{mps}_i \le \gamma_1 \\ \text{DE/best/1}, & \gamma_1 < \text{mps}_i \le \gamma_1 + \gamma_2 \\ \text{DE/rand/1}, & \gamma_1 + \gamma_2 < \text{mps}_i \le \gamma_1 + \gamma_2 + \gamma_3 \\ \text{DE/current-to-rand/1}, & \text{(Otherwise)} \end{cases} \tag{19}$$

4.2.3 Crossover

Uniform crossover is utilized to mutant vector w_i^j, and target vector X_i^j to obtain offspring χ_i^j i.e., ith solution at jth iteration. Equation (20) shows the crossover

operation, where $\mathbb{R}$ is a random number generated in the range [0,1]. Crossover is applied successfully; if the value of $\mathcal{R}$ is less than the corresponding crossover rate (CR), then the exchanging of ith gene of parent chromosomes is achieved. Otherwise, the parent solution is passed to the next generation.

$$\chi_i^j = \begin{cases} \omega_i^j \ (\mathbb{R} \in (0,1)) \leq CR \\ \mathcal{X}_i^j \ (Otherwise) \end{cases} \tag{20}$$

This work randomly initializes the crossover rate in the range [0,1]. The mean value of crossover rate CR and standard deviation CR_s are 0.5 and 0.1, respectively. After a certain number of iterations, the crossover rate, termed the crossover learning period (lp^c), is updated. At each iteration, the number of solutions passed through the next generation is ζ_1, ζ_2, ζ_3, and ζ_4 that represents four different mutation techniques. Similarly, β_1, β_2, β_3, and β_4 are the solutions that fail to reach the next iteration. Γ_1, Γ_2, Γ_3, and Γ_4 are the selection probabilities for DE/rand/1, DE/best/1, DE/ current-to-best/1, and DE/ current-to-rand/1 schemes and stated in Eqs. (21–25).

$$\begin{aligned} \Delta =& 2(\zeta_1\zeta_2\zeta_3 + \zeta_1\zeta_2\zeta_4 + \zeta_1\zeta_3\zeta_4 + \zeta_2\zeta_3\zeta_4) \\ &+ \beta_1(\zeta_2 + \zeta_3 + \zeta_4) + \beta_2(\zeta_1 + \zeta_3 + \zeta_4) \\ &+ \beta_3(\zeta_1 + \zeta_2 + \zeta_4) + \beta_4(\zeta_1 + \zeta_2 + \zeta_3) \end{aligned} \tag{21}$$

$$\Gamma_1 = \frac{\zeta_1(\zeta_2 + \beta_2 + \zeta_3 + \beta_3 + \zeta_4 + \beta_4)}{Z} \tag{22}$$

$$\Gamma_2 = \frac{\zeta_2(\zeta_1 + \beta_1 + \zeta_3 + \beta_3 + \zeta_4 + \beta_4)}{Z} \tag{23}$$

$$\Gamma_3 = \frac{\zeta_3(\zeta_1 + \beta_1 + \zeta_2 + \beta_2 + \zeta_4 + \beta_4)}{Z} \tag{24}$$

$$\Gamma_4 = 1 - (\Gamma_1 + \Gamma_2 + \Gamma_3) \tag{25}$$

4.2.4 Selection

The selection operation is performed using the concept of survival of the fittest using Eq. (26) to select candidates to be passed on to the next generation, where Ψ_i^{j+1} is the candidate chosen in the next iteration, the solution obtained after the crossover is χ_i^j and the existing solution is Ψ_i^j.

$$\Psi_i^{j+1} = \begin{cases} \chi_i^j \ (\text{fitness}(\chi_i^j) \leq \left(\text{fitness}\left(\Psi_i^j\right)\right) \\ \Psi_i^j \ (\text{Otherwise}) \end{cases} \tag{26}$$

5 Operational design and complexity analysis

The functional design of the proposed EER-VMP model is depicted in Algorithm 3. It initializes the total number of PMs (p) and VMs (q). In steps 1–3, random VM allocation solutions are generated by calling Algorithm 1. It obtains Z number of VM-PM allocations randomly by utilizing random-fit (RF) and first-fit decreasing (FFD) algorithms having a time complexity $O\ (Z \times p \times q)$. Each allocation must follow the resource capacity constraints as stated in Eqs. (6)–(8). In steps 4–8, the resource usage of each VM is predicted by calling Algorithm 2, which provides training steps of resource predictor whose time complexity is $O\ (m \times n \times G \times W \times I)$, where m and n are the numbers of neurons present at the input and hidden layer of NN, G and W are the numbers of networks and network weight connections, and I denotes the total number of iterations. Steps 10–16 compare the predicted CPU and memory of each VM with the actual value of CPU and memory of available PM; if it is less or equal to any PM, then the allocation is successful; otherwise, there is a failure. Hence, the overall time complexity can be computed as $O\ (p^2 \times q^2 \times m \times n \times G \times W \times I)$.

Algorithm 1 Random VM allocation module

Input: $List_{PM}$, $List_{VM}$, p,q
Output: Random allocations of VMs

```
for k = 1, 2, ..., Z do
    for i = 1, 2, ..., p do
        for j = 1, 2, ..., q do
            while List_PM ≠ Empty do
                for each V in List_VM do
                    choose V_j from List_VM;
                    choose S_i from List_S randomly
                    for ℝ ∈ {CPU, RAM, Mem} do
                        if V_j^R ≤ S_i^R then
                            Assign V_j to S_i;
                            set z_ji=1;
                            Compute S_i^R=S_i^R - V_j^R;
                            Delete V_j from List_V;
                        else
                            V_j is not allocated;
                            set z_ji =0
                        end
                    end
                end
            end
        end
    end
end
for i = 1, 2, ..., Z do
    Use FFD algorithm for remaining unassigned Vs;
end
```

Algorithm 2 VM resource prediction module

Input: $\mathbb{CR}$=0.5, $\mathbb{CR}_s$=0.1, maximum number of population (M), $\gamma_1 = \gamma_2 = \gamma_3 = \gamma_4 = 0.25$
Output: Predicted resource usage of each VM
Initialize neural network with m, n, p neurons at input, hidden and output layer, w bias, and Γ solutions ;
Evaluate the fitness of each solution (Ψ) using fitness function (RMSE) as stated in Eq. (14);
for $i \leq I$ **do**
  Produce mps of N random number $\in$ [0, 1];
  for $j \leq N$ **do**
    Randomly create $r_1 \neq r_2 \neq r_3 \neq r_4 \neq i \in$ [0, 1];
    if $0 < mps_i \leq \gamma_1$ **then**
      Apply $DE/best/1$ mutation$List_V$;
      else if $mps_i \leq \gamma_1 + \gamma_2$ **then**
        Apply $DE/current-to-best/1$ mutation
        else if $mps_i \leq \gamma_1 + \gamma_2 + \gamma_3$ **then**
          Apply $DE/rand/1$ mutation
          else
            Apply $DE/current-to-rand/1$ mutation
          end
        end
      end
    end
  end
end
Apply uniform crossover ;
Evaluate the fitness value of the newly obtained solution by using RMSE ;
Perform selection operation for the new generation;
Update ζ_1, ζ_2, ζ_3, ζ_4, β_1, β_2, β_3, β_4 ;
Update Γ_1, Γ_2, Γ_3, Γ_4;

Algorithm 3 Reliable VM allocation module

Input: $List_{PM}$, $List_{VM}$, p,q
Output: Reliable VM allocations
for $i = 1, 2, \ldots, p$ **do**
  for $j = 1, 2, \ldots, q$ **do**
    Random VM allocation()
  end
end
for *every* j^{th} *VM on* V *and* $\{S_1, S_2, \ldots, S_P\} \in S$ **do**
  $\{V_i^{pre.C}, V_i^{pre.M}\} \longleftarrow$ VM Resource prediction();
  if $(V_j^{pre.C} \leq S_i^C)$ *and* $(V_j^{pre.M} \leq S_i^M)$ **then**
    Allocate V_j to S_i ;
    else
      Allocation Unsuccessful ;
    end
  end
end
Compute Re using using Eq. (3) and PW using Eq. (5) ;

6 Performance evaluation

6.1 Experimental setup

The experiments were conducted on a dedicated server machine, featuring 2 Intel®Xeon®Silver 4114 CPUs with 40 core processors, clocked at 2.20 GHz, and running Ubuntu 16.04 LTS with 128 GB of RAM. The data center environment simulated in the experiments consists of three types of physical machines (PMs) and four types of virtual machines (VMs). PMs and VMs are configured in Tables 2 and 3. The resource features like power consumption ($P_{\max}$, $P_{\min}$), MIPS, RAM, and memory are taken from real server IBM and Dell configuration where S1 is 'ProLiantM110G5XEON3075,' S2 is 'IBMX3250Xeonx3480,' and S3 is 'IBM3550Xeonx5675,' respectively. The VM configurations were inspired by Amazon EC2 instances [32]. The experiment utilizes the Google cluster dataset (GCD) [33], which contains detailed resource usage data, including CPU, memory, and disk I/O requests for 672,300 tasks executed on 12,500 PMs over 29 days. CPU and memory

Table 2 PM configuration

PM type	PE	MIPS	RAM (GB)	Storage (GB)	$P_{\max}$	$P_{\min}/P_{\text{idle}}$
S_1	2	2660	4	160	135	93.7
S_2	4	3067	8	250	113	42.3
S_3	12	3067	16	500	222	58.4

Table 3 VM configuration

VM type	PE	MIPS	RAM (GB)	Secondary storage (GB)
V_{Small}	1	500	0.5	40
V_{Medium}	2	1000	1	60
V_{Large}	3	1500	2	80
$V_{\text{Extra Large}}$	4	2000	3	100

utilization for each VM were collected at 5-min intervals over 24 h. This setup does not rely on a cloud simulation framework but instead uses a physical server environment to simulate the data center operations, providing real-world resource allocation data for analysis. Table 4 initializes all the parameters involved in the learning process of NN. Following are the performance parameters computed in our work:

- Accuracy of actual versus predicted workload.
- Power consumption (PW)
- Reliability of the system (Re)
- Number of active PMs (APMs)

6.2 Simulation results

Several experiments were performed based on distinct combinations of PMs and VMs to analyze the performance of the proposed EER-VMP model. The dataset does not specify the number of cloud users, a random number of users was assumed, each requesting different types and quantities of VMs. Multiple performance metrics were assessed in the simulation, and the results are presented in Table 5. It is observed that the power consumption (PW) has increased linearly with data center size. The reliability metric for each workload is more than 64%, which varies between 64.11 and 65.25%. Table 5 shows that the number of active PMs (APM) increases with the increased size of the data center, and the time of execution is linear to the number of requested *VM*s. Figures 2 and 3 show the effectiveness of the resource prediction module, where predicted CPU and memory usage almost overlap with actual CPU and memory usage for the considered dataset. The reason for

Table 4 Experimental setup parameters and their values

Input layer neurons (m)	7–28	Hidden layer neurons (n)	4–20
Output layer neurons (l)	1	Total number of iteration (I)	100
Size of training data	75%	Number of population	100
Mutation learning period	10	Cross over rate	0.9
Mutation rate	0.8	learning rate	0.001
Batch size	32	Momentum	0.9

Table 5 Performance metrics for EER-VMP

VMs	PMs	User	PW (KW)	RE(%)	APMs	Exec. time (ms)
50	30	20	3.29E+3	64.11	12	2.59
100	60	40	7.05E+3	64.52	31	3.12
200	120	80	1.31E+4	64.17	41	17.69
400	240	160	2.18E+4	64.29	80	20.29
600	360	240	3.27E+4	64.33	100	107.59
800	480	320	4.16E+4	65.25	109	123.17
1000	600	400	4.98E+4	64.23	219	165.74
1200	720	480	5.12E+4	65.33	269	191.29
1600	960	640	5.18E+4	65.19	311	247.41

Table 6 Optimization of neural network (SADE vs. SGD)

Epochs	Self-adaptive differential evolution		Stochastic gradient descent	
	MAE	PW (KW)	MAE	PW (KW)
10th	0.0062	7.05E+3	0.0361	8.05E+3
20th	0.0054	1.31E+4	0.0137	1.96E+4
40th	0.0039	2.18E+4	0.0133	3.21E+4
60th	0.0016	3.27E+4	0.0299	4.06E+4
80th	0.0011	4.16E+4	0.0194	5.66E+4
100th	0.0014	4.98E+4	0.0503	7.17E+4

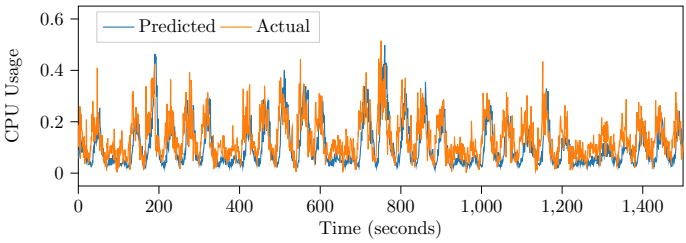

Fig. 2 Predicted versus actual CPU workload

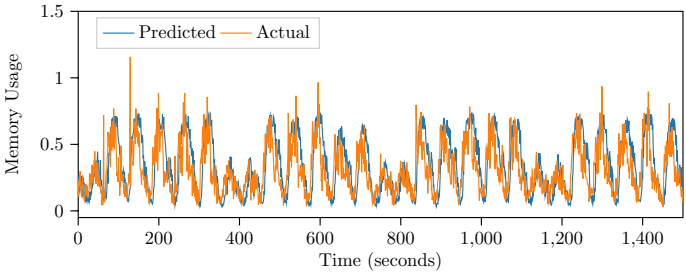

Fig. 3 Predicted versus actual memory workload

such accuracy is that the proposed NN-based resource predictor periodically learns and retrains in response to historical workload. In addition, the application of the SADE technique explores the search space in multiple directions. It works with the N solution, which enables efficient learning of patterns and correlations from historical data. The comparison between optimization algorithms such as self-adaptive differential evolution (SADE) and stochastic gradient descent (SGD) across various epochs, as shown in Table 6, demonstrates that SADE consistently outperforms SGD in both accuracy and energy efficiency. SADE achieves significantly lower mean absolute error (MAE), with values steadily decreasing over time and reaching a minimum of 0.0011 at the 80th epoch, while SGD exhibits higher MAE values, peaking at 0.0503 at the 100th epoch. In terms of energy consumption (PW), SADE shows more efficient usage, with a gradual increase in energy consumption over the epochs. In contrast, SGD's power consumption increases more rapidly throughout the epochs. Overall, SADE proves to be more effective in minimizing both error and power usage compared to SGD.

6.3 Comparative analysis

The existing techniques selected for the comparison are WOGA [14], HGAPSO [15], NSGA-II [16], GA [17], SMOOP [18], WOA [19], random-fit [28], best-fit [34], First-Fit [35], and OP-MLB [22].

Figure 4 compares power consumption (PW) across the proposed EER-VMP model and existing approaches. The proposed model consistently shows the lowest power consumption, outperforming all other methods, especially in larger data centers. While PW increases with data center size for all approaches, the proposed model maintains a significant energy efficiency. At a data center size of 200 VMs, PW values converge across all approaches, indicating that the efficiency gap narrows for medium-sized data centers. However, the EER-VMP model demonstrates superior performance in reducing PW as the data center grows, highlighting its scalability and energy efficiency, which is crucial for sustainable cloud computing. Figure 5 compares the number of active physical machines (APMs) used during VM placement, showing that the EER-VMP model significantly reduces the number of active

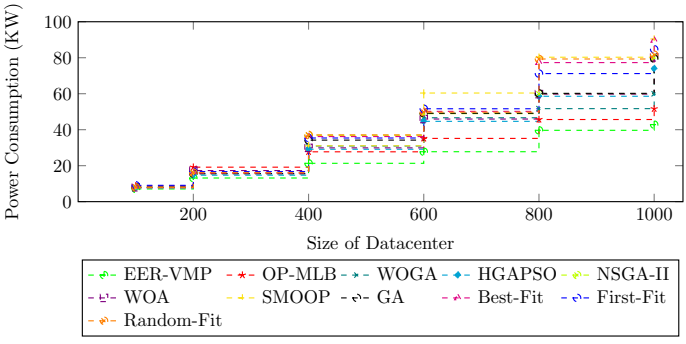

Fig. 4 Power consumption

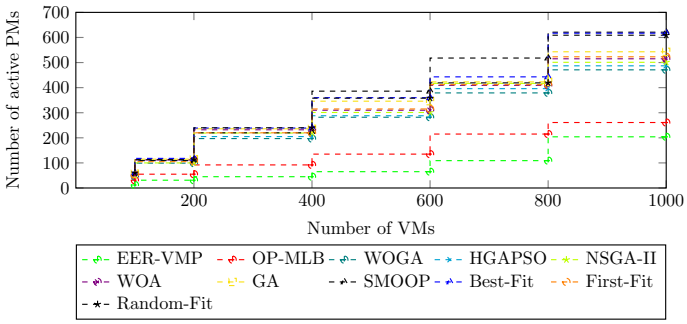

Fig. 5 Number of active PMs

PMs compared to existing approaches. This reduction highlights the model's efficiency in resource allocation, leading to optimized resource usage, and lower energy consumption. The proposed model demonstrates superior proficiency in minimizing APMs, making it a more effective solution for large-scale cloud environments where efficient resource management is essential for sustainability and cost-effectiveness. Figure 6 illustrates the overall system reliability before and after the implementation of resource prediction. The proposed model significantly improves system reliability, achieving an impressive value of up to 65.25%, which is crucial in minimizing the risk of system failures and reducing the need for VM migrations. This enhancement in reliability is key to maintaining stable and efficient cloud operations.

In dynamic virtual machine placement (VMP), VM configurations are determined at runtime, allowing for the dynamic reallocation and deallocation of VMs based on resource utilization. This process involves moving VMs from overutilized or underutilized PMs to the most efficient ones, ensuring optimal resource distribution. To compare the performance of the EER-VMP model against existing algorithms for dynamic VM placement, the GCD dataset [33] is utilized. The CPU and memory utilization percentages of VMs are extracted from the GCD dataset and recorded at 5-min intervals. In this scenario, 20% of users are randomly selected from the total data center size. Key performance metrics, such as energy consumption (PW), active physical machines (APM), and operational execution time, are analyzed within this dynamic environment, with comparative results presented in Table 7.

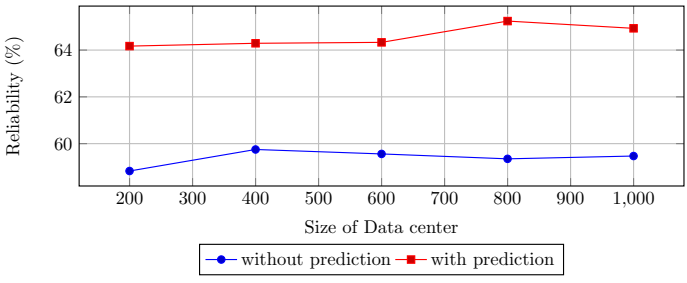

Fig. 6 Reliability

Table 7 Proposed versus state-of-the-art comparison for dynamic VMP

Approaches	VMs	Users	PW (*KW*)	APMs	Execution time
EER-VMP	100	20	7.55E+3	31	3.12
	200	40	1.41E+4	41	17.69
	400	80	2.83E+4	80	20.28
	600	120	3.57E+4	100	107.59
	800	160	4.26E+4	109	123.17
	1000	200	5.58E+4	219	145.74
OP-MLB [22]	100	20	8.80E+03	39	6.39
	200	40	1.96E+04	54	11.76
	400	80	3.17E+04	111	28.16
	600	120	4.11E+04	142	62.59
	800	160	5.59E+04	179	138.15
	1000	200	7.92E+04	233	216.81
WOGA [14]	100	20	7.69E+3	46	5.73
	200	40	1.54E+4	99	32.61
	400	80	3.09E+4	197	32.61
	600	120	4.66E+4	282	196.78
	800	160	5.17E+4	379	108.61
	1000	200	5.99E+4	471	196.78
HGAPSO [15]	100	20	7.67E+3	51	16.37
	200	40	1.46E+4	110	77.15
	400	80	2.91E+4	205	154.62
	600	120	4.46E+4	288	234.41
	800	160	5.85E+4	396	678.58
	1000	200	7.39E+4	487	1138.30
NSGA-II [16]	100	20	7.91E+3	49	8.12
	200	40	1.59E+4	100	83.61
	400	80	3.68E+4	219	78.92
	600	120	4.89E+4	301	256.89
	800	160	5.99E+4	417	453.56
	1000	200	7.99E+4	501	679.16
GA [17]	100	20	8.07E+3	57	7.46
	200	40	1.55E+4	115	81.72
	400	80	3.42E+4	236	134.65
	600	120	4.91E+4	346	258.04
	800	160	6.01E+4	423	217.90
	1000	200	7.94E+4	543	421.34
SMOOP [18]	100	20	7.77E+3	58	4.92
	200	40	1.56E+4	109	82.35
	400	80	3.10E+4	220	25.63
	600	120	6.03E+4	386	317.08
	800	160	8.02E+4	518	121.81
	1000	200	9.04E+4	621	164.50

Table 7 (continued)

Approaches	VMs	Users	PW (*KW*)	APMs	Execution time
WOA [19]	100	20	7.99E+3	51	5.81
	200	40	1.72E+4	112	90.76
	400	80	2.99E+4	232	37.43
	600	120	4.59E+4	310	241.13
	800	160	5.97E+4	409	129.55
	1000	200	8.02E+4	515	162.20
Random-fit [28]	100	20	7.97E+3	60	41.86
	200	40	1.60E+4	113	84.56
	400	80	3.71E+4	240	194.86
	600	120	4.97E+4	358	261.19
	800	160	7.92E+4	419	416.45
	1000	200	8.16E+4	609	428.95
Best-fit [34]	100	20	7.78E+3	60	3.67
	200	40	1.60E+4	113	84.56
	400	80	3.71E+4	240	25.89
	600	120	4.97E+4	358	261.19
	800	160	7.92E+4	419	89.23
	1000	200	8.16E+4	609	148.7
First-fit [35]	100	20	9.07E+3	52	1.32
	200	40	1.66E+4	105	87.66
	400	80	3.53E+4	220	9.17
	600	120	5.16E+4	315	271.17
	800	160	7.11E+4	412	38.29
	1000	200	8.47E+4	523	98.19

7 Conclusion

In this work, a prediction-based energy-efficient and reliable VM placement model is developed. The proposed work aims to reduce power consumption and improve overall system reliability to manage cloud data centers effectively. The NN-based resource prediction unit is utilized to predict the upcoming resource usage of each VM; then, the SADE algorithm is used to optimize the neural network weight. The performance evaluation of the proposed EER-VMP model indicates that it reduces power consumption (PW) by minimizing the number of active PMs (APMs), and a significant improvement is achieved in the overall system reliability of the cloud data center. Experiments are conducted on real cloud workload traces, and the simulation supports all the obtained results. The comparison with existing techniques reveals that the proposed EER-VMP technique can significantly reduce PW and the number of APMs and improve the overall reliability of the cloud data center.

Author contributions Smruti Rekha Swain was involved in conceptualization, methodology, design of the work, software, analysis, writing—original draft, visualization, investigation, and implementation. Anshu Parashar and Ashutosh Kumar Singh took part in conceptualization, visualization, research gap finding, and writing—review and editing. Chung Nan Lee participated in conceptualization, visualization, research gap finding, and writing—review.

Data availability No datasets were generated or analyzed during the current study.

Declarations

Competing interests The authors declare no competing interests.

References

1. Mell P, Grance T et al (2011) The nist definition of cloud computing
2. Lin W, Luo X, Li C, Liang J, Wu G, Li K (2023) An energy-efficient tuning method for cloud servers combining DVFS and parameter optimization. IEEE Trans Cloud Comput
3. Gupta A, Namasudra S, Kumar P (2024) A secure VM live migration technique in a cloud computing environment using blowfish and blockchain technology. J Supercomput 1–24
4. Swain SR, Saxena D, Kumar J, Singh AK, Lee C-N (2024) An intelligent straggler traffic management framework for sustainable cloud environments. IEEE Trans Sustain Comput
5. Hiremath TC, Rekha K (2023) Energy efficient data migration concerning interoperability using optimized deep learning in container-based heterogeneous cloud computing. Adv Eng Softw 183:103496
6. Yang W, Zhao M, Li J, Zhang X (2024) Energy-efficient DAG scheduling with DVFS for cloud data centers. J Supercomput 80(10):14799–14823
7. Kaur G, Bala A, Chana I (2019) An intelligent regressive ensemble approach for predicting resource usage in cloud computing. J Parallel Distrib Comput 123:1–12
8. Sayadnavard MH, Toroghi Haghighat A, Rahmani AM (2019) A reliable energy-aware approach for dynamic virtual machine consolidation in cloud data centers. J Supercomput 75(4):2126–2147
9. Shang L, Peh L-S, Jha NK (2003) Dynamic voltage scaling with links for power optimization of interconnection networks. In: The Ninth International Symposium on High-Performance Computer Architecture. HPCA-9 2003. Proceedings. IEEE, pp 91–102
10. Nawrocki P, Grzywacz M, Sniezynski B (2021) Adaptive resource planning for cloud-based services using machine learning. J Parallel Distrib Comput 152:88–97
11. Swain SR, Saxena D, Kumar J, Singh AK, Lee C-N (2024) An intelligent straggler traffic management framework for sustainable cloud environments. IEEE Trans Sustain Comput 1–13. https://doi.org/10.1109/TSUSC.2024.3393357
12. Rostami S, Broumandnia A, Khademzadeh A (2024) An energy-efficient task scheduling method for heterogeneous cloud computing systems using capuchin search and inverted ant colony optimization algorithm. J Supercomput 80(6):7812–7848
13. Kaviarasan R, Harikrishna P, Arulmurugan A (2022) Load balancing in cloud environment using enhanced migration and adjustment operator based monarch butterfly optimization. Adv Eng Softw 169:103128
14. Saxena D, Gupta I, Kumar J, Singh AK, Wen X (2021) A secure and multi-objective virtual machine placement framework for cloud data center. IEEE Syst J
15. Sharma NK, Reddy GRM (2016) Multi-objective energy efficient virtual machines allocation at the cloud data center. IEEE Trans Serv Comput 12(1):158–171
16. Singh AK, Kumar J (2019) Secure and energy aware load balancing framework for cloud data centre networks. Electron Lett 55(9):540–541
17. Tseng F-H, Wang X, Chou L-D, Chao H-C, Leung VC (2017) Dynamic resource prediction and allocation for cloud data center using the multi-objective genetic algorithm. IEEE Syst J 12(2):1688–1699

18. Han J, Zang W, Chen S, Yu M (2017) Reducing security risks of clouds through virtual machine placement. In: IFIP Annual Conference on Data and Applications Security and Privacy. Springer, pp 275–292
19. Abdel-Basset M, Abdle-Fatah L, Sangaiah AK (2019) An improved lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment. Clust Comput 22(4):8319–8334
20. Saxena D, Kumar J, Singh AK, Schmid S (2023) Performance analysis of machine learning centered workload prediction models for cloud. IEEE Trans Parallel Distrib Syst 34(4):1313–1330
21. Wang B, Liu F, Lin W, Ma Z, Xu D (2021) Energy-efficient collaborative optimization for VM scheduling in cloud computing. Comput Netw 201:108565
22. Saxena D, Singh AK, Buyya R (2021) OP-MLB: An online VM prediction based multi-objective load balancing framework for resource management at cloud datacenter. IEEE Trans Cloud Comput
23. Peake J, Amos M, Costen N, Masala G, Lloyd H (2022) PACO-VMP: parallel ant colony optimization for virtual machine placement. Futur Gener Comput Syst 129:174–186
24. Xuejie Z, Zhijian W, Feng X (2013) Reliability evaluation of cloud computing systems using hybrid methods. Intell Autom Soft Comput 19(2):165–174
25. Sharma Y, Javadi B, Si W, Sun D (2016) Reliability and energy efficiency in cloud computing systems: survey and taxonomy. J Netw Comput Appl 74:66–85
26. Zhou A, Wang S, Zheng Z, Hsu C-H, Lyu MR, Yang F (2014) On cloud service reliability enhancement with optimal resource usage. IEEE Trans Cloud Comput 4(4):452–466
27. Azimzadeh F, Biabani F (2017) Multi-objective job scheduling algorithm in cloud computing based on reliability and time. In: 2017 3th International Conference on Web Research (ICWR). IEEE, pp 96–101
28. Jangiti S, Sri Ram E, Shankar Sriram V (2019) Aggregated rank in first-fit-decreasing for green cloud computing, pp 545–555
29. Sayadnavard MH, Haghighat AT, Rahmani AM (2021) A multi-objective approach for energy-efficient and reliable dynamic VM consolidation in cloud data centers. Eng Sci Technol Int J
30. Iorio AW, Li X (2004) Solving rotated multi-objective optimization problems using differential evolution. In: Australasian Joint Conference on Artificial Intelligence. Springer, pp 861–872
31. Zhang L, Chang H, Xu R (2012) Equal-width partitioning roulette wheel selection in genetic algorithm. In: 2012 Conference on Technologies and Applications of Artificial Intelligence. IEEE, pp 62–67
32. Amazon (1999) Amazon EC2 instances. https://aws.amazon.com/ec2/instance-types/. Accessed 19 Jan 2022 (Online)
33. Reiss C, Wilkes J, Hellerstein JL (2019) Google cluster-usage traces: format+ schema. Google Inc., White Paper 1
34. Shirvastava S, Dubey R, Shrivastava M (2017) Best fit based VM allocation for cloud resource allocation. Int J Comput Appl 158(9)
35. Jung G, Hiltunen MA, Joshi KR, Schlichting RD, Pu C (2010) Mistral: dynamically managing power, performance, and adaptation cost in cloud infrastructures. In: 2010 IEEE 30th International Conference on Distributed Computing Systems. IEEE, pp 62–73

Authors and Affiliations

Smruti Rekha Swain[1] · Anshu Parashar[1] · Ashutosh Kumar Singh[1] · Chung Nan Lee[2]

✉ Smruti Rekha Swain
smruti_62000063@nitkkr.ac.in

Anshu Parashar
anshuparashar@nitkkr.ac.in

Ashutosh Kumar Singh
ashutosh@nitkkr.ac.in

Chung Nan Lee
cnlee@mail.cse.nsysu.edu.tw

[1] Department of Computer Applications, National Institute of Technology, Kurukshetra, Haryana 136119, India

[2] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 804201, Kaohsiung, Taiwan