An Energy-Efficient VM Selection Using Updated Dragonfly Algorithm in Cloud Computing

Ajay Prashar* and Jawahar Thakur

Department of Computer Science, Himachal Pradesh University, Shimla-171005, Himachal Pradesh, India
Email: ajayprashar93@gmail.com (A.P.); jawahar.hpu@gmail.com (J.T.)
*Corresponding author
Manuscript received September 20, 2023; revised October 19, 2023; accepted February 7, 2024; published July 11, 2024

***Abstract*—Cloud computing is popular among industries, academia, and government to supply reliable and scalable computational power. High-speed networks in cloud data centers connect Virtual machines with Physical Machines. Virtualization assists the cloud service providers to manage resources effectively but unoptimized and inefficient services degrade the performance of the system. The scheduling architecture of cloud computing includes Physical Machines (PMs), Virtual Machines (VMs) and the allocation and migration policy of the VMs over the PMs. The overutilized PMs get few migrations and this paper introduces novel behavior of VM selection from overutilized PMs using Swarm intelligence. The evaluation of the proposed algorithm architecture is compared with another state-of-the-art optimization algorithm from the same series. The evaluation has been done on the base of Quality of Service (QoS) parameters, such as Service Level Agreement (SLA) violation, and energy consumption against various load variation scenarios to support elasticity. The proposed algorithm has outperformed other techniques by considerable margin in terms of QoS, and the details are presented in the results section. The simulation results demonstrate that the proposed technique exhibits 6.3% and 6.7% enhancement in terms of reduced energy consumption compared to both Cuckoo Search (CS) and general Dragonfly (DF) techniques, and 3% decrease in SLA violations in comparison to current methods. Additionally, the results reveal an 11% enhancement in VM migration compared to existing approaches.**

***Keywords*—cloud computing, Virtual Machine (VM) placement, migration, dragonfly, Cuckoo Search (CS)**

I. Introduction

Cloud computing is the most intriguing concept for today's businesses in the IT sector. The exchange of information is widespread globally over the internet [1]. There is a tremendous amount of data that must be retained and constantly transmitted using the Internet in the cloud center. Cloud computing offers a wide range of services in different prospects such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [2]. More specifically, the popular cloud services attract the attention of huge companies such as Google, Microsoft, IBM, and Wipro that make centralized data centers across the world. Cloud centers consist of physical hosts in thousands that consume a significant amount of energy. Therefore, companies keep on building more and more data centers while the cloud resources of current data centers are still not fully utilized [3]. However, cost-effective services require the delivery of efficient and affordable utilization of Virtual Machines (VM) [4]. For the energy-efficient data center, virtualization is the key technology to provide interoperable and flexible services.

Utilization of machine learning in combination with Swarm Intelligence (SI) has been observed as a solution to the VM selection policy from over-utilized Physical Machines (PMs). The collective behavior of decentralized, self-organizing systems, which draws inspiration from the coordination seen in natural swarms, is referred to as swarm intelligence. It makes use of the interactions of simple agents to solve issues in a complex and adaptable way. This method is frequently used to simulate the flexibility and efficiency observed in social insect colonies or bird flocks in domains including robotics, optimization, and artificial intelligence. Algorithms such as particle swarm optimization and ant colony optimization can be included in swarm intelligence models. Parvizi and Rezvani [5], Ghasemi and Toroghi Haghighat [6] demonstrated the usage of meta-heuristics, also referred to as SI for the selection of the VM selection policy along with the utilization of machine learning [7].

The generalized allocation process and placement scheme of VM are illustrated in Fig. 1 [8]. In this paper, we focused on VM allocation and migration process by assuring that minimum power is consumed by the machines with minimum violation of service level. Virtualization is a key technology to make the Data Centre (DC) energy efficient while balancing the load [9]. Therefore, VMs can be migrated, deleted, and created among the host machines depending upon the usage of power. VM management that is energy efficient is extended to task scheduling, consolidation of workload, request batching, selection of mobile service, choosing the remote or local clouds, etc.

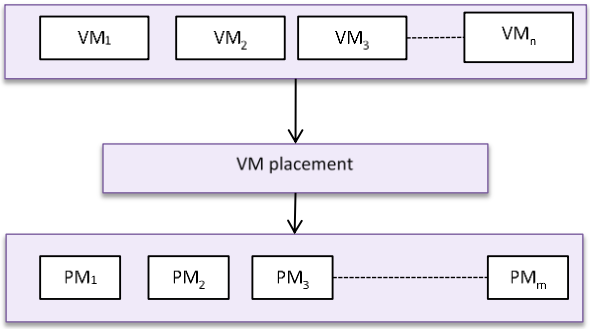

Fig. 1. Network of host machine and virtual machine [8].

Commercial applications for virtualization require more computational resources than the complimentary resources that lead to migration of VM in the cloud computing platform. Moreover, complementary resources cannot extend beyond a limit and loads must be handled effectively without any violations of service level and Quality of Service (QoS). In such a case, two types of migrations take place namely the

migration from overloaded hosts and the migration from under loaded hosts [10–12]. The migration of virtual machines occurs across both over and underload servers to achieve load balancing, transferring workloads from overwhelmed servers to servers with available resources, all while ensuring the requirements of QoS-sensitive applications are met. An under-loaded server is one that is not efficiently utilizing its available resources. In other words, it has excess CPU, memory, or other resources that are not being fully utilized. Under-loaded servers are inefficient and can lead to wasted resources and higher operational costs. Addressing server overloads and under-loads is critical for maintaining the efficiency, performance, and cost-effectiveness of a cloud data center. Live VM migration, server consolidation, dynamic resource allocation, and load balancing are all valuable tools and techniques to manage and optimize server resource usage in Dynamic Voltage and Frequency Scaling (DVFS)-enabled cloud data centers [13]. Different migration algorithms were developed to determine the utilization status of VMs and migrate the underutilized VMs to the less utilized ones when there are sufficient resources. The existing studies focus on a selection of VM using metaheuristic and energy-efficient techniques such as learning automata and the Analytical Hierarchy Process (AHP). However, such techniques are limited to utilizing the power in the case of heavily loaded machines. The current studies focus on migrating the VMs considering the utilization of resources of VM, CPU utilization, target systems, QoS, and VM requirements for target systems. Therefore, the main contribution of this work is as follows:

- An improved meta-heuristic-based algorithm architecture for the selection of the VMs.
- A comparison architecture for the proposed meta-heuristic architecture with other state-of-the-art algorithms of the same series.
- An integrated training and classification architecture for the rank generation of the VMs based on their migration mechanism.
- Comparative analysis of the proposed model with the existing techniques has been presented in terms of different quality of service parameters.

The remainder of the paper is arranged as follows: Section II discusses the current techniques for efficient migration of VM. the research methodology in which optimization models for VM placement and migration are implemented and their features are analyzed in Section III. The results and discussion are represented in Section IV and finally conclusion is presented in Section V.

II. Research Background

The VM allocation problem is defined using the notations. Consider a cloud DC having $p \times q$ racks and each contains hosts (h). The number of total hosts (T_H) is computed using the Eq. (1) [14].

$$T_H = p \times q \times h \tag{1}$$

The cloud infrastructure consists of users $U = \{u1, u2, u3 \ldots, u_n\}$ where n is the total number of users. The users submit their requests on the cloud data center either by themselves or via brokers. The cloud data center has P number of PMs and V number of VMs and the PMs and the VMs can be represented as a set PMs = $\{p_1, p_2, p_3, \ldots, P\}$ and VMs=$\{v_1, v_2, \ldots, V\}$. As the VMs are associated with the PM in order to execute the request generated from the users, the VMs use the resources of the PMs. Higher job volume will increase the load on the system and hence the CPU will be more utilized as the power. Considering the literature studies, CPU utilization is the main power consumption factor of the cloud [15, 16]. For this, there is a need to determine the direct relation between the usage of power and the performance of the CPU of the cloud host. The determination of accumulative usage of resources of allocated VMs to a specific host machine is determined by computing the CPU power utilized by the host in Million Instructions per Second (MIPS). The CPU consumed by the host is computed by dividing the total CPU capacity of the associated VMs by the CPU utilization of the host machine is given as follows:

$$CPU_u = \sum_{j=1}^{V_k} \frac{v_{j_{cpu}}}{P_{i_{cpu}}} \tag{2}$$

where k is the total number of VMs associated with i^{th} PM.

There is a degradation of the performance of the VM due to outages of resources during the execution that will cause Service Level Agreement Violations (SLA-V) and downtime of the VM. Thus, the total degradation of VM performance is computed as per the work. Further, the migration of the VM and overutilization time of the host is considered to compute the migration of the VM in a given time slot. The VMs are migrated from one PM to another based on the conditions shown in Eq. (3) as follows:

$$f(CPU) = \begin{cases} -1, & p_{i_{cpu}} \leq \min\limits_{threshold} CPU_u \ \forall\, v \\ 1, & p_{i_{cpu}} \geq \max\limits_{threshold} CPU_u \ \forall\, v \end{cases} \tag{3}$$

The function returns –1 which denotes that all the VMs from the current PM p_i will be migrated as the PM is not utilizing it. Here, 30% CPU utilization is referred to as the minimum threshold and 70% as the maximum threshold.

In such a scenario, no algorithmic architecture is required to select the VMs as all the VMs have to be migrated. In the second scenario, the function returns 1 when the CPU utilization of the current PM p_i exceed the maximum utilization limit of the CPU. In this scenario, the PM is considered to be overloaded and some of the VMs are selected in order to bring the overutilized PM to a normalized PM category. The total consumed power, referred to as Power Consumption (PC) is the sum of idle PC and execution PC defined in Eq. (4) as follows:

$$PC = PC_{idle_{v_j}} + PC_{ex_{v_j}} \tag{4}$$

where $PC_{ex_{v_j}}$ is the execution cost at v_j VM for any PM, $PC_{idle_{v_j}}$ is the idle cost of vm_j the execution cost can be computed using Eq. (5) as follows.

$$PC_{ex_{v_j}} = CPU_{u_{v_j}} \times puc_{v_j \ \forall PM_i} \tag{5}$$

where $CPU_{u_{v_j}}$ is the CPU utilization of v_j VM and "*puc*" is the per unit cost of execution under PM_i.

When a user is associated with the cloud server, the cloud

server agrees to the service terms which are generally referred to as Service Level Agreement (SLA). In the case of mathematical computing, the violations made to SLA are called SLA-V and are calculated based on any computation QoS parameter. Here in the case of the proposed work, the SLA-V is computed based on power consumption as shown in Eq. (6).

$$(SLA-V) = \begin{cases} 0, & PC_i \leq th \\ x, & x = \frac{PC_i - th}{th} \end{cases} \quad (6)$$

where "*th*" is the threshold of power consumption of each pm "*i*".

An appropriate placement and VM migration are done through an effective utilization of resources, and to reduce the consumption of energy. The challenges and various issues noticed in VM migration are analyzed considering the different QoS parameters. The main challenges are generally relying on continuity of network connection, and migration of data considering the storage and memory aspects. Several studies were conducted considering these aspects and some of these are discussed to determine the research gap. Ruan *et al.* [17] suggested a method for determining the host machine's optimal operating utilization levels. To make an idea workable given that performance and power statistics must be measured on actual platforms, a method called "PPR Gear" considers the different sampling levels of utilization and calculates Performance-to-Power Ratios (PPR). The authors also provide a framework for allocating and migrating virtual machines that make use of the PPR for different host types. The framework can guarantee that host computers operate at the most power-efficient level by striking the ideal balance between host utilization and energy consumption. Wei *et al.* [18] presented an exact algorithm to deal with the bin packing problem of idle and working machines. The experiment was performed considering the small, medium, and large-scale data instances from DCs. The authors developed the best-fit algorithms that were used to combine the fit rules considering the computation time with regards to PM and VM. The algorithm performance was computed for different instances and total energy consumption with different variants provided a suitable maximum number of resources in order to fulfil service level agreements [19]. An essential technology in cloud computing is virtualization. Creating several VM instances, helps cloud providers manage data center resources effectively, which improves resource use. In order to meet acceptable Service Level Agreement (SLA) criteria, this study introduces a novel machine-learning-based method for dynamically integrating Virtual Machines (VMs) based on adaptive forecasts of utilization thresholds. Ahmadi *et al.* [20] presented a flexible approach for addressing the challenge of VM selection in the cloud computing environment. They utilized a hierarchical-based process for decision-making. The simulation analysis performed using 1000 VMs resulted in a reduction of energy consumption by 23% with 49% reduced VM migrations. This results in reduced better overall performance. Khan and Santhosh [21] proposed a hybrid optimization approach in this research effort to manage the migration of VMs in a cloud environment. The suggested hybrid optimization model was created by combining the Cuckoo Search (CS) and Particle Swarm Optimization (PSO) algorithms. This research's main goal is to cut down on energy use, calculation time, and migration expenses. Another goal of this research project is to maximize resource use. The effectiveness of the hybrid simulation study is confirmed through simulation analysis and compared with traditional algorithms in terms of performance metrics to justify the research objective. From the results, it is acquainted that energy consumption using the proposed technique is 0.470 watts with a load of about 0.0025. However, the migration performance is still limited which may have achieved using the multi-optimization technique. A detailed comparative analysis of various existing VM allocation techniques is tabulated in Table 1.

Table 1. Comparison of existing VM allocation techniques

Reference	Technique	Evaluation tool	Improved Metrics	Workload and Data Center
Abdessamia *et al.* [22]	Virtual Machine Placement using the Gravitational Search Optimization algorithm	Simulation using the MATLAB tool	Energy consumption, and no. of active server	Artificial and Synthetic (heterogeneous servers)
Abdel-Basset *et al.* [23]	VM migration using the Wolf optimization algorithm with levy flight	Simulation using the CloudSim	CPU utilization, and no. of physical server	PlanetLab and Synthetic (cloud user-customized VMs)
Parvizi and Rezvani [5]	Metaheuristic approach for VM placement	Simulation using the CloudSim toolkit	CPU utilization, resource loss, energy consumption, and execution time.	PlanetLab and Synthetic (cloud user-customized VMs)
Rasouli *et al.* [24]	VM placement using the Learning Automata approach	Simulation using the CloudSim toolkit	SLA violation, energy consumption, and number of VM migration	PlanetLab and Synthetic (cloud user-customized VMs)
Ghasemi and Haghighat [6]	Load balancing and migration using Machine Learning	Simulation using the CloudSim toolkit	Migration cost for VM, execution time, and number of shut down server	PlanetLab and Synthetic (cloud user-customized VMs)
Azizi and Li [25]	Heuristic-based VM migration technique	Simulation using C++	Energy consumption, wastage of resources, CPU utilization, and number of active servers	Artificial and Synthetic, Real world
Wei *et al.* [18]	Exact method-based VM placement	Simulation using Gurobi solver and Python	Computation time, energy consumption, number of active servers, and utilization of resources	Artificial and Real-world (Google Datacenter)
Aboamama and Hamouda [26]	Genetic Algorithm-based VM placement	Simulation using MATLAB	Energy consumption, Wastage of resources, and elapsed run time	Artificial and real-world Data Using Travelling Salesman Problem
Tarahomi *et al.* [14]	Micro genetic approach	Simulation using the CloudSim toolkit	SLA violation, energy consumption, number of server shutdown	PlanetLab using the Real workload
Shirvani *et al.* [27]	Energy-Efficient VM Placement	MOD-JAYA	Power Consumption, Resource Wastage	Cloud Data Centers, Multi-Objective Optimization

From the above literature, it is clear that there is a need for an energy-efficient VM placement technique using the optimization approach as most of the researchers are inefficient in providing the optimized VM placement technique. However, some of the techniques are still limited which may have been achieved using the multi-optimization technique. Some of the researchers were limited to applying the Machine Learning technique for efficient placement. Therefore, carefully understanding the literature work, it is clear that there is a need for optimization techniques to optimize the VM placement Equipped with Machine Learning techniques.

III. Energy Efficient Placement and Migration of VM

The energy consumption is dominated during the assignment of VM to PM in VMMP cloud DC. Whenever a computational request is assigned to the data centre then deployment of the request has been done for specific configurations considering the different computational resources such as CPU utilization, execution time, and memory size. The execution process is done after assigning the VM to the PM. The simultaneous execution process of VM to PM consumes an enormous amount of energy. Additionally, the numbers of PMs are arranged in a cluster or group and the PMs in the cluster are turned off until the assignment of VM in the group. However, frequent shutdowns will incur serious damage to the system by consuming a lot of energy, and idle machines also consume energy even VM is not hosted by the server. Thus, efficient computational resources are provided to the customer by maintaining the Makespan time, energy consumption, and service cost during the development of schedules.

The proposed dragonfly-based optimization model is based on the selection and migration of VMs for efficient placement. Optimization models such as firefly, Cuckoo search, ant colony optimization, and many more such models are developed in the cloud for energy-efficient placement. But, still, the presented works are limited to addressing the respective features for VM placement and selection for migration. This research incorporates different optimization techniques such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Cuckoo Search (CS), and Dragonfly (Df).

The implementation has been done using these techniques and performance is compared further to determine the best results. The main problem during the VM placement is given as:

- In the procedure of resource allocation, two different cases are considered. In the first case, the PM is considered overloaded due to the number of user requests more than available slots. In such a scenario, additional computational resources can reduce the overload but it is going to increase the Cost of Investment (CoI) and will reduce the Return on Investment (RoI).
- In the second case, the VM requests are less but PM still available to provide services. Therefore, a state of idle PM exists that also consumes the power to satisfy the requirements of other users as shown in Fig. 2.

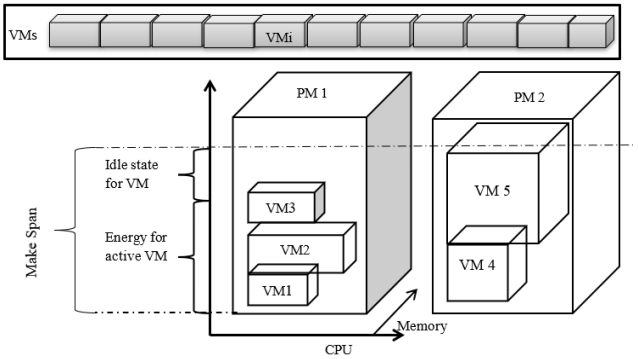

Fig. 2. VM migration and placement.

To avoid the above two problems, VM migration is employed considering the different aspects such as migration cost, energy consumption, allocation of resources, and computational utilization of resources. The cloud parameters considered in this research are illustrated in Table 2 and include a set of PMs, VMs, Memory, and requirement of bandwidth for PM and VM. These parameters are essential for performing the simulation analysis, and affect the overall performance. The constraints considered for placement of VMs (VM) to PMs (PM), and Quality of Service (QoS) objectives: Energy consumption, SLA violation, and Makespan are considered. In the VMPA scheduling, a set of PMs are hosting the set of VMs constrained by CPU and utilization of memory. For a feasible placement, the CPU demand of the VM (CPU^{VM}) under the CPU capacity which is provided by PM (CPU^{PM}) ensuring that ($CPU^{VM} \leq CPU^{PM}$). Similarly, the memory demand is utilized by PM (M^{PM}) and meet the demand for respective VMs (M^{VM}) ensuring that ($M^{PM} \geq M^{VM}$). PM hosts the set of VMs considering the CPU, memory and time, and the placement is considered a 3D-bin packing problem. In two different levels, active PM and VM consume enormous energy. The PM will consume energy in the range of $[E_{PM}^{-}, E_{PM}^{+}]$, that relies on the energy-efficient $[\widetilde{E_{PM}}]$ of the hosted VM (Vm).

Table 2. Simulation parameters

S. No.	Parameter	Value
1.	Number of DC's	5
2.	Total number of PM's	10–100
3.	Total number of tasks	25–57
4.	Total number of VMs	100–1000
5.	Memory (VM)	2 Gb
6.	Memory (Host)	4 Gb
7.	CPU capacity of Host (MIPS)	1000–3000
8.	CPU capacity of VM (MIPS)	250–1000
9.	Bandwidth of VM	100 Mbit/sec
10.	Bandwidth of PM	1 Gbits/sec
11.	Gradient Value	6.48
12.	Workload coefficient	0.1×–0.4×

A. Proposed VM Selection Approach Using Dragonfly (DF) Algorithm

In 2016, Mirjalili [28] developed the dragonfly algorithm which is one of the most effective algorithm architectures from the meta-heuristic series. The DF algorithm is based on either the efficient hunting procedure of the dragonflies or the aspects of the migration of the dragonflies from one end to another or one field to another. DF is an interesting nature-inspired algorithm devised to solve complex optimization problems. Dragonflies are small flies that are carnivorous and eat a large number of bees, ants, butterflies, and mosquitoes.

There are around 2800 different species of dragonflies, and their lifecycle is different. They consist of two stages, one is adult and the other one is a nymph. The working mechanism of the dragonflies is based on static and dynamic behaviors the former is based on the feeding mechanism and later is a migratory mechanism. In case of the static behavior, the group formation of the dragonflies is limited to a small group size whereas in the case of dynamic behavior, the swarming size becomes high. Furthermore, the static and dynamic behavior constitutes the exploration and exploitation phase respectively. In the exploitation phase, dragonflies in the swarm fly over long distances in one direction and distract from harmful flies (enemy). In the exploration phase, there is a small group that flies back and forth over a small area to attract prey for food. The features related to Dragonfly in different contexts are as follow:

- Dragonfly topology is a scalable and fault-tolerant network design used in high-performance computing. It excels in efficient routing, low-latency communication, and supporting numerous compute nodes, crucial for data exchange in computing environments.
- DragonFlyBSD is an open-source Unix-like OS with features like the HAMMER file system for advanced data management. It uses lightweight kernel threads and supports Symmetric Multiprocessing (SMP), making it versatile for desktop and server applications.
- They have compound eyes, transparent wings, and agile flight patterns. These predators play a vital ecological role by hunting other insects.
- In the realm of Unmanned Aerial Vehicles (UAVs), “Dragonfly” denotes agile small drones used for surveillance, reconnaissance, and data collection. Their versatility, maneuverability, and applicability to civilian and military tasks are their defining features.
- Dragonfly topology is designed to optimize resource allocation, which includes efficiently distributing computational tasks across the available CPUs while keeping power consumption in check. This makes it a valuable choice for high-performance computing environments where balancing computational power with energy efficiency is essential.

The proposed work utilizes CPU utilization and power consumption as major deciding and evaluation parameters for the dragonflies. The five different principles in the case of dragonflies have been utilized.

Separation (S): It represents the avoidance from the neighbors to avoid collision. It is mathematically modelled as given below:

$$S_u = -\sum_{v-1}^{F} P - P_v \tag{7}$$

where P is the position of the firefly and P_v is the position of the neighboring individual and F is the number of flies in the neighbor.

Alignment (A): It is the speed of the dragonfly through which it propagates towards its global best solution. In other words, the speed of the fly matches with neighborhood flies swarming in the same group. The alignment is given as follows:

$$A_u = \frac{\sum_{v-1}^{F} V_v}{F} \tag{8}$$

where V_v is the velocity of the v^{th} fly in the group.

Cohesion (C): It is the tendency of individual dragon flies towards the center of the global best position in the centre of the group.

$$C_u = \frac{\sum_{v-1}^{F} P_v}{F} - P \tag{9}$$

Attraction: The flies attracted toward the food are mathematically modelled as:

$$F_{S_u} = P_{F_S} - P \tag{10}$$

where F_{S_u} is the food source of the fly and P_{F_S} is the position of food source.

Distraction (D): The distraction from the enemies is mathematically modelled as:

$$E_u = P_E + P \tag{11}$$

where E_u denotes the enemy position of u^{th} individual, and P_E is the position of the enemy.

The workflow for the proposed work can be demonstrated using the following workflow diagram as shown in Fig. 3. When dealing with discrete state problems, such as those found in many cloud computing and resource allocation scenarios (e.g., allocating VMs to physical servers), it's important to adapt the algorithm or use a different approach that is more suitable for discrete problems. k-means is used to divide the allocations into 3 states as normal, overloaded and under loaded.

The proposed DF algorithm works on levy flights and co-relation in which for each levy flight, a dragon is either awarded with a positive reward or a negative reward. There are a total of 15 steps in the proposed DF algorithm that are illustrated as follows in terms of algorithmic architecture.

Algorithm 1. Proposed Dragonfly Algorithm

```
Input: HL, where HL is the overload host list
 At is the allocation table
Output: V ML is the VM list to be migrated
[dg, dc] = kmeans(At, 3)
// divide the allocation table into 3 states as normal,
overloaded and underloaded.
Where dg is the dragon group and dc is the dragon
centroid [global food]
For(i = 1 : Len((H1))
    vms = Find (AL, HLs[i]) // Find all vms of concerned host
    drg = vms // consider vms as dragons
    For(j = 1 : Len(drg)
       Dgj = dg[j]; find the state of current dargon
       Lf = 10; //create a reward matrix that holds reward
       value for each fight
       For(k = 1 : if)
          Ep = 30; exp = 60; where ep and exp is swarm size
          percentage of exploitation and exploration.
          A = Cos(At[vms], C)/ Eucl(At[drp], C); //Define
          alignment as the ratio of cosine similarity to Euclidean
          distance of all VM's parameters to the global food
          parameter defined as cohesion
          S = Cos(At[drp], C)/Eucl(At[drp], C); // Define
          separation as the ratio of cosine similarity to
          Euclidean distance of the group parameters to the
          global food parameters defined as cohesion.
          [f, fv] = DragonFitness(A, S, C); where f is Boolean
          value for the fit unfit[1,0]and fv is the fitness value
          from the fitness function.
          If(f == 1) //Assign reward for the flight.
```

```
            R[k] = 100 − fv; //Assign reward for the flight
        Endif
            Rm = Mean(R) // Compute mean of rewards
        If (Rm ≤ 60)
            Vml.append(drg[j]); //This VM is selected for the
            migration
        Endif
        Define DragonFitness[A, S]
        F, fv = 0
        If(fv = (A − S)/A) ≤. 30f; Endif
        Return f, fv
        End DragonFitness
    EndFor
  EndFor
EndFor
Return vml
```

The different features of the VMPA are formulated using the proposed dragonfly algorithm is given as follows:

- The constraints of PMs such as utilization of CPU and Memory capacity are considered for the placement of VM to PM, $CPU^{VM} \leq CPU^{PM}$) and ($M^{PM} \leq M^{VM}$) for all feasible VM placements x_{PmVm}.
- It is entirely pre-emptive which indicates that only one VM can be hosted by PM at any time, $\sum_{Vm} x_{PmVm} \leq 1$ at any time t for a PM (Pm).
- When the placement imitates, then it is acquainted that the process must be ON before all VMs placed. The Make span time is the earliest completion of VMs for PMs, $\sum_{Vm} T_{Vm}^{Vm} x_{PmVm}$.
- In the proposed work scenario, two different types of energy are considered in the processing energy during the placement of VMs to PM. These are denoted by processing energy (A_{energy}), and the idle energy (I_{energy}) in which PM is in an active state but does not host the VM. Thus, total energy consumption is the sum of active energy and idle energy ($T_{energy} = A_{energy} + I_{energy}$).
- The processing energy A_{energy} depends upon the energy efficiency ($\tilde{E}_{PmVm}$ and execution time of VM (T_{Vm}^{Vm}).
- The energy efficiency for the placement of VM to PM is given as follows, where minimum and maximum energy consumed by PM for one hour represented as $[E_{Pm}^{-}, E_{Pm}^{+}]$.
- PM in idle state also consumes energy and is determined by idle time and minimum energy consumed per hour.

$$\tilde{E}_{PmVm} = E_{Pm}^{-} + (E_{Pm}^{+} - E_{Pm}^{-}) \cdot e^{\frac{CPU_{VM}}{CPU_{PM}}} \quad (12)$$

- Idle power consumption is computed by considering the m^{th} VM in an idle state and PM also in an idle state.

$$E_{VM_i}^{idle} = \begin{cases} Power_{server_l}^{idle} & \exists \frac{z}{U_{VM_i}^{z}} = 100\% \\ \frac{\sum_z \alpha_z . U_{VM_i}^{z}}{\sum_z \alpha_z} \cdot Power_{server_l}^{idle} & otherwise \end{cases} \quad (13)$$

where α_z is the weight assigned to resource z and $U_{VM_i}^{z}$ is the utilization of resource z by i^{th} VM. Here, ∃ before z denotes the existence of a value for the variable z such that the condition that follows is true. It means that for at least one value of z that meets the given criterion, the equation is applicable.

This indicates that idle power consumed by the VM is equivalent to the idle power consumed by the servers if there is 100% utilization of the VM.

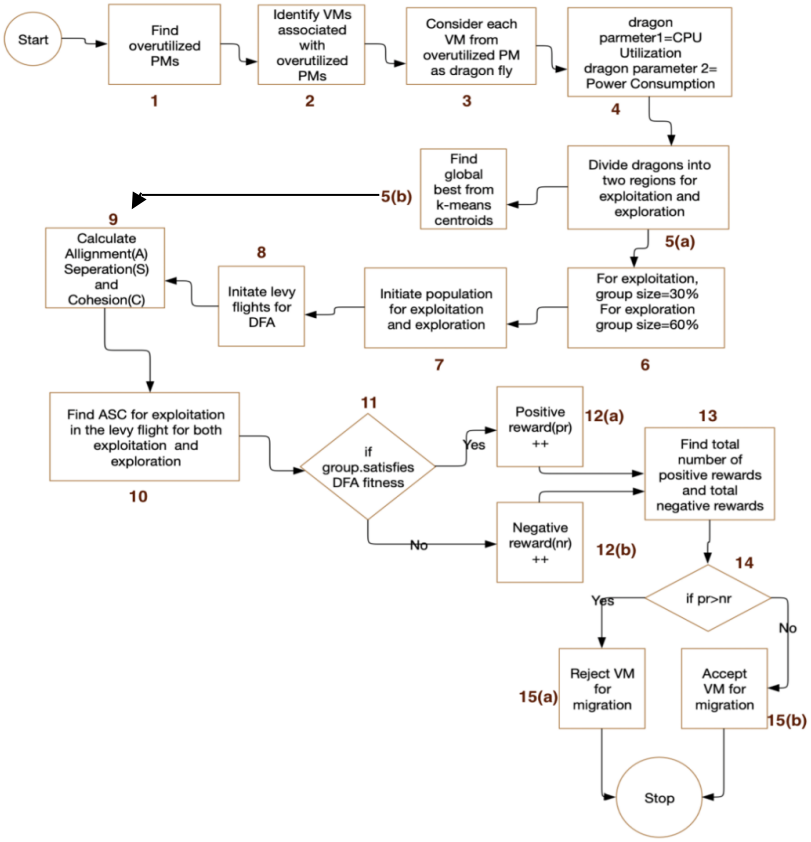

Fig. 3. The proposed workflow for the DF algorithm.

The proposed work uses modification in the dragonfly fitness function to achieve the claimed outcomes. The proposed dragonfly algorithm is implemented to compute the efficiency of the PM to host the VM. The efficient utilization of PM to avoid the wastage of power. In order to comprehend the proposed algorithm, the proposed optimization algorithm is also compared with other state-of-the-art algorithms as given in the later sections. The parametric settings for the DF algorithm are illustrated in Table 3 [29].

Table 3. Parameter settings for the dragonfly algorithm

Parameter	Value
Number of Simulation Rounds	100
Number of search agents	5
Search Domain	[0 1]
Dimension	Number of features acquired in the data
Number of runs	15

B. Optimization Using the Cuckoo Search (CS)

CS is a nature-inspired technique that consists of several eggs used to represent the PM. For four different PMs such as 'R', 'S', 'U', and 'W', a set of VMs are represented by (1–12). R = {1,2,3,4}, S = {5,6,7,8}, U = {9,10,0,0}, W = {11,12,0,0}. Therefore, nest is denoted as {1,2,3,4}, {5,6,7,8}, {9,10,0,0}, {11,12,0,0}. CS is a metaheuristic algorithm which is designed considering the behavior of the Cuckoos. The nature of the Cuckoo is to lay eggs in other nests considering the amazing abilities such as laying eggs having strong nests or other eggs chosen as their eggs. Parasitic Cuckoos roamed to find nests where other Cuckoos lay eggs in no time and accuracy is high for laying eggs. In such a scenario, the other Cuckoo will remove the eggs from the nest and this reduces the probability of legitimate eggs. In some cases, Cuckoos find that the egg in the nest is foreign and therefore, abandon the nest and search somewhere for a new nest [30]. In brief, Cuckoo at the final stage destroys the original nest that it has intruded and occupied by pretending that it belongs to the nest of the original mother bird and thus, does irreparable harm by hatching the eggs early, and therefore causing their demise. The evaluation procedure consists of considering the three operators:

a) Levy fight

New solutions are produced using the levy flight. The external Cuckoo gets more food when host chicks call.

b) Existing nests are replaced with new solutions

In this process, the probability of a new solution is computed by randomly selecting a new value for each solution.

c) Selection of VM list

A comparison has been made with the old value, if the new one has better quality the updated solution is considered as the final one and the other one is ignored. The algorithm for the proposed CS is given as follows:

Algorithm 2. Cuckoo Search optimization

1. **Input**: **A set of VMs**, **and set of PMs**
2. **Initialize the operators of CS** – Simulation Round
   − No. of Eggs (€) // No. of VMs
   − Variables// No. of PMs
3. Compute the size of the VM $S_i \leftarrow$ Size (VM)
4. Fitness Function F (fit)

f(fit)

$$= \begin{cases} \text{True; if superior quality eggs are maintained } (f_r > f_{th}) \\ \text{False; otherwise} \end{cases}$$

Where, f_r is the random change in the position of the egg and f_{th} is the threshold value for host birds.

5. For each simulation round
6. $f_r = \sum_{r=1}^{n} \text{VM energy} = P_n$
7. $f_{th} = \frac{\sum_c^n \text{power (VMs)}}{P_c}$
8. $VM_{allocation} = CSO(P_n, N_{var}, f(fit))$
9. End
10. Return: An optimized VM list is created for allocation
11. End

C. Optimization Using the Ant Colony Optimization (ACO)

ACO is a metaheuristic algorithm adapted to solve the VM placement problem in the cloud environment [31]. As developing a new swarm series algorithm will take a lot of other researchers from other fields, the researchers change the behavioral architecture of the swarm algorithm. ACO technique is based on the foraging behavior of the ants to compute the best path. The ants when foraging release, the pheromone on the path where they move. As it has been illustrated earlier the authors adopt behavioral change in the algorithm architecture, the basic ACO work is customized in several researches [32]. In the selection procedure, the ants are considered as VM and PC has been observed as a vital parameter for migration in most of the studied cases. The probability of migration in the case of ACO is given by Eq. (14) as follows.

$$P_{r,c} = \begin{cases} 0, \text{ if } host_c \text{ have not enough resources for } VM_r \\ \frac{\iota_{r,c}^{\alpha} \cdot \eta_{r,c}^{\beta}}{\sum_{k=1}^{n} \iota_{r,c}^{\alpha} \cdot \eta_{r,c}^{\beta}}, \text{ otherwise} \end{cases} \quad (14)$$

where, $\iota_{r,c}^{\alpha}$ is the pheromone released by the ants on the path from allocation of VM_r *to* PM_c that means ant prefers PM_c to place VM_r in the previous iteration rounds if $\iota_{r,c}^{\alpha}$ is larger and $\eta_{r,c}$ is the visibility level. It signifies the tendency for allocation from VM_r *to* PM_c considering the ant perspective itself. The suitability concept is introduced for allocation and $\eta_{r,c}$ is computed as follows:

$$\eta_{r,c} = \frac{1}{fit_{r,c}} \quad (15)$$

$$fit_{r,c} = \left(Res_{CPU} - \overline{Res}_{r,c}\right)^2 + \left(Res_{memory} - \overline{Res}_{r,c}\right)^2 + \left(Res_{Bandwidth} - \overline{Res}_{r,c}\right)^2 \quad (16)$$

$$\overline{Res}_{r,c} = \frac{Res_{CPU} + Res_{memory} + Res_{Bandwidth}}{3} \quad (17)$$

where, Res_{CPU} is the CPU utilization, Res_{memory} is the memory utilization, and $Res_{Bandwidth}$ is the utilization of bandwidth for the remaining resources in PM. The idle state and active state both account for the same. The larger value of $\eta_{r,c}$ shows that there is a greater tendency of ants to allocate VM_r *to* PM_c. α and β represent the pheromone and the importance of visibility. The algorithm based on ACO is given as follows:

Algorithm 3. Ant Colony optimization (ACO)

1. **Initiate the Pheromone level**
2. **Input n number of ants**
3. for each simulation round (I) do

4. for each ant do
5. for each VM do
6. Place V_r to H_c as per the PM selection policy
7. End for
8. End for
9. Update the best position as per global pheromone policy.
10. End for
11. Output the migration and allocation
12. Repeat the steps until the best position is obtained.

VM is optimized using the different optimization techniques and results are further evaluated to determine the best optimization technique.

IV. Result and Discussion

Extensive simulation experiments have been conducted to evaluate the performance of the designed secure cloud model. The choice of simulator is driven by project specific requirements. The simulation was performed on MATLAB in addition to the Intel core i3 processor operated at 2.30 GHz oscillator frequency, 64-bit OS, and 4 GB RAM. During the experiment, we had two participants one is a cloud user and the other is a cloud server. The cloud user acts like a data owner and behaves as an authorized user. On the other side, a cloud server acts like a Cloud Service Provider. The performance of the designed model has been evaluated as described below.

A. Performance Analysis

Scalability is the system's capacity to manage growing workloads or resource demands efficiently. A system's scalability tells how well it can scale up or down to meet changing requirements without compromising performance, energy efficiency, or satisfying SLAs—by evaluating how it operates under various configurations (varying VMs and PMs). The proposed work is therefore evaluated based on the following evaluation parameters.

- Energy Consumption: It is calculated as the total consumed power in order to perform utility to the VM allocation and migration that maintains the SLA.
- SLAV: It is the ratio of the consumed power to desired power consumption and violation of service level.
- Number of Migrations: It is the total number of migrations of the VMs in the rack of PM.

Fig. 4 and Table 4 show the comparison of different techniques to determine the energy consumed by the VM and PM in the data centers. The workload is accessed and the average energy consumed using the dragonfly technique is 11.89 kWh while energy consumed using the CS, ACO, PSO and DF techniques is 12.7 kWh, 12.78 kWh, 12.86 kWh and 12.76 kWh, respectively. The analysis results prove that the proposed dragonfly performs better in comparison to other techniques. Therefore, VM placement using the dragonfly technique consumes less energy and there is an improvement of about 6.3%, 14%, 21%, and 6.7% from CS, ACO, PSO and DF techniques, respectively.

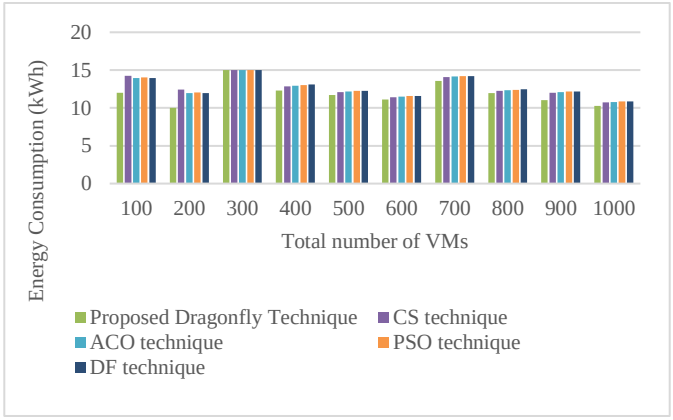

Fig. 4. Comparative analysis for energy consumption (kWh).

Table 4. Comparative analysis for energy consumption (kWh) using different optimization techniques

Total VM	Total PM	Proposed dragonfly technique (kWh)	CS technique (kWh)	ACO technique (kWh)	PSO technique (kWh)	DF technique (kWh)
100	20	12.02000	14.23300	13.97000	14.03000	13.95000
200	40	10.02570	12.43870	11.97570	12.03570	11.95570
300	60	14.98700	15.00930	15.01160	15.02390	15.02490
400	80	12.28700	12.86370	12.94040	13.01710	13.09380
500	100	11.71213	12.09883	12.17553	12.25223	12.25893
600	120	11.12130	11.41800	11.49470	11.57140	11.57810
700	140	13.56667	14.06634	14.14304	14.21974	14.22041
800	160	11.94667	12.24634	12.32304	12.39974	12.47674
900	180	11.02667	12.00337	12.08007	12.15677	12.16847
1000	200	10.26667	10.71534	10.79204	10.86874	10.87544

Fig. 5 and Table 5 shows the comparison of different techniques for SLA violation due to negotiation in contracts between the user and service providers in the data centers. The average SLA violation using the proposed dragonfly is 0.138 while the violation of service level using the CS, ACO, PSO and DF technique is 0.146, 0.155, 0.163, and 0.160, respectively.

The analysis results prove that the dragonfly technique performs better in comparison to other techniques. Therefore, the proposed dragonfly technique better manages the violation in service level and it is improved by 5.7%, 11.2%, 15.6%, and 13.7% from CS, ACO, PSO, and DF techniques respectively. This improvement is due to the use of energy-efficient techniques and heuristic search mechanisms by the dragonflies.

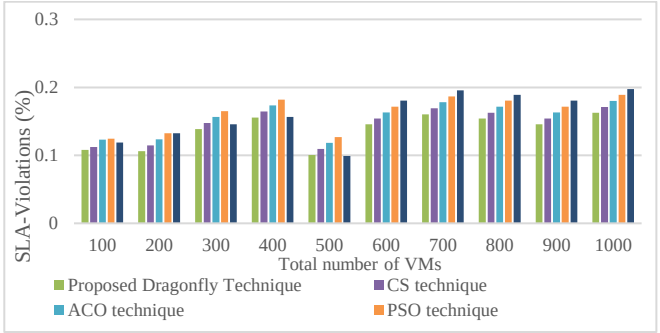

Fig. 5. Comparative analysis for SLA violation.

Table 5. Comparative analysis for SLA violation using different optimization techniques

Total VM	Total PM	Proposed dragonfly technique (kWh)	CS technique (kWh)	ACO technique (kWh)	PSO technique (kWh)	DF technique (kWh)
100	20	0.10790	0.11233	0.12322	0.124544	0.11883
200	40	0.10604	0.11482	0.12360	0.13238	0.13246
300	60	0.13880	0.14758	0.15636	0.16514	0.14555
400	80	0.15577	0.16455	0.17333	0.18211	0.156566
500	100	0.10059	0.10937	0.11815	0.12693	0.098999
600	120	0.14545	0.15423	0.16301	0.17179	0.18057
700	140	0.16045	0.16923	0.17801	0.18679	0.19557
800	160	0.15405	0.16283	0.17161	0.18039	0.18917
900	180	0.14545	0.15423	0.16301	0.17179	0.18057
1000	200	0.16245	0.17123	0.18001	0.18879	0.19757

Table 6 shows the comparison of different techniques for the Number of VM migrations in the data centers. The average migrations using the proposed dragonfly is 7.8 while the average number of migrations using the CS, ACO, PSO, and DF techniques is 8.4, 9.3, 8.8, and 8.7, respectively. The improvement is seen in the dragonfly technique in comparison to other techniques as shown in Fig. 6. Therefore, the dragonfly technique has lessened the number of VM migrations and it is improved by 7.1%, 9.6%, 11.3%, and 10.3% from CS, ACO, PSO, and DF technique, respectively.

Table 6. Comparative analysis for number of migrations using different optimization techniques

Total VM	Total PM	Proposed dragonfly technique	CS technique	ACO technique	PSO technique	DF technique
100	20	1	1	2	2	1
200	40	2	3	4	3	3
300	60	6	8	8	8	7
400	80	9	9	10	9	8
500	100	3	5	4	4	5
600	120	8	9	9	11	10
700	140	15	13	14	12	16
800	160	1	2	3	3	4
900	180	15	14	16	15	14
1000	200	18	20	23	21	19

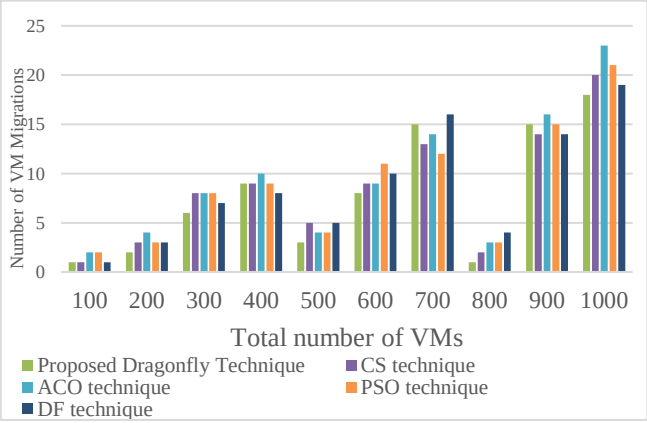

Fig. 6. Comparative analysis for the number of VM migrations.

B. Comparative Analysis

The present article based on different optimization techniques is compared with the existing techniques to validate the results. The results using the proposed dragonfly are better as compared to CS, ACO, PSO and DF techniques. The existing techniques proposed by Talwani *et al.* [19] use the K-Nearest Neighbor (KNN) for VM migration and allocation in the data center. The proposed work is also compared with Khan and Santhosh [21], in which a hybrid model using the CS and PSO technique is proposed for VM migration. Further, it is compared with Huang *et al.* [2] in which a VM allocation strategy was proposed by analyzing the demands of user requirements.

Fig. 7 shows the comparison of the proposed dragonfly technique with the existing technique for 100 numbers of VMs. The average value for energy consumption using the proposed dragonfly technique is 12.02 kWh while energy consumed using dynamic cloud architecture proposed by Talwani *et al.* [19] and Huang *et al.* [2] is 14.31 kWh and 15.3 kWh respectively. Thus, the proposed technique is improved by 16% from Talwani *et al.* [19] and 21.4% from Huang *et al.* [2]. The improvement in the proposed technique is due to the energy-efficient technique employed with a dragonfly that consumes less energy in comparison to existing techniques.

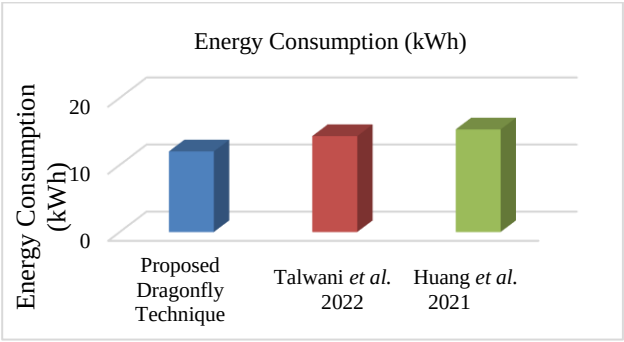

Fig. 7. Comparison with existing techniques for energy consumption.

Fig. 8 shows the comparison of the proposed dragonfly technique with the existing technique for 100 numbers of VMs. The average value for SLA violation using the dragonfly technique is 0.107 while the technique proposed by Talwani *et al.* [19] and Huang *et al.* [2] shows SLA of about 0.148 and 0.5 respectively. Thus, the proposed technique is improved by about 3% from Talwani *et al.* [19] and 7.4% from Huang *et al.* [2]. The improvement in the proposed technique is due to energy energy-efficient technique employed with the dragonfly technique which consumes less energy in comparison to existing techniques.

Fig. 9 shows the comparison of the proposed dragonfly technique with the existing technique for 100 numbers of VMs. The average value for migrations using the proposed dragonfly technique is 8 while 8.5 migrations are shown by

Khan and Santhosh [21] and 9 shown by Talwani *et al.* [19]. Thus, the proposed technique is improved by 11.1% from Talwani *et al.* [19] and 5.8% from Khan and Santhosh [21].

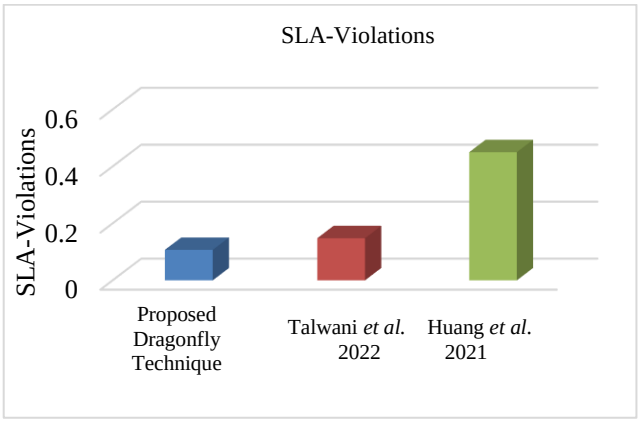

Fig. 8. Comparison with existing techniques for SLA violation.

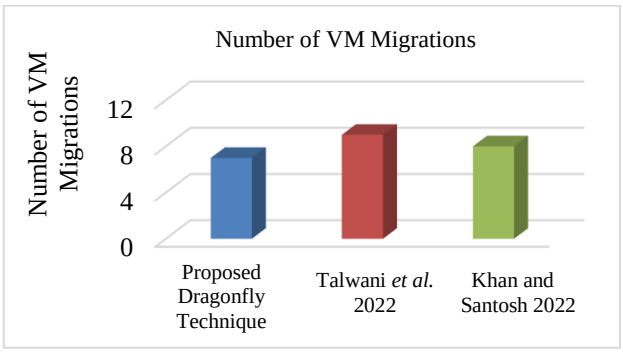

Fig. 9. Comparison with existing techniques for the number of VM migration.

The improvement in the proposed technique is due to energy energy-efficient technique employed with the dragonfly technique which consumes less energy in comparison to existing techniques.

Fig. 10 shows the comparison of dragonfly technique with the existing technique for 1000 numbers of VMs. The average CPU utilized using the proposed dragonfly technique is 0.2977% while 0.30647% utilized using the technique proposed by Huang *et al.* [2]. The technique proposed using the CS, ACO, PSO, and DF have shown an average CPU utilization of 0.3012% using work of Talwani *et al.* [19] and, 0.2997% using work of Khan and Santhosh [21]. Thus, overall utilization for different number of test groups, proposed dragonfly technique attained the best performance with maximum CPU utilization.

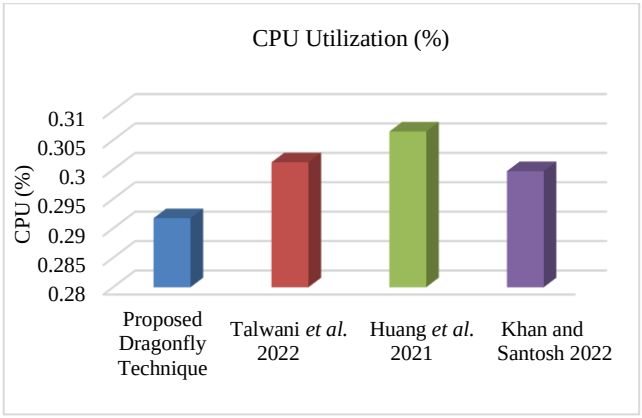

Fig. 10. Comparison with existing technique for CPU utilization.

V. Conclusion

This paper presents the VM allocation and Migration problem considering the different optimization techniques. The different optimization techniques such as dragonfly, CS, ACO, and PSO are implemented for the allocation of VM with minimum wastage of resources. The proposed technique is optimized in accordance with CPU utilization, and resource requirements such as power and agreement of service levels. The different techniques are evaluated in terms of energy consumption, number of migrations, and SLA violation. The analysis results shows that proposed dragonfly technique perform better and further validated using the existing technique. The simulation results are shown that proposed technique is improved by 6.3% and 6.7% in terms of energy consumption from CS and general DF technique and 3% for SLA violation in comparison to current techniques. Furthermore, 11% improvement is seen in VM migration in comparison to existing techniques. In future, multiclass meta-heuristic and machine learning techniques are employed for better extraction of features. The limitations of the study's findings are that they may not be universally applicable to all real-world cloud computing scenarios. This is because the proposed technique's performance was evaluated under specific conditions, which constrain its ability to address the full spectrum of operational challenges in diverse environments.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

AP conducted the research and analyzed the data; JT supervised the research work. Both authors had approved the final version of the article.

Acknowledgment

The authors wish to express their sincere gratitude to everyone who contributed to this research. We are particularly thankful to our seniors for their unwavering motivation and support throughout the duration of this work.

We would also like to extend our appreciation to our colleagues and peers for their valuable insights and feedback, which greatly improved the quality of this research

References

[1] H. Chen, Y. Wen, and Y. Wang, "An energy-efficient method of resource allocation based on request prediction in multiple cloud data centers," *Concurrency and Computation: Practice and Experience*, e7636, 2023. doi: 10.1002/CPE.7636

[2] Y. Huang, H. Xu, H. Gao, X. Ma, and W. Hussain, "SSUR: An approach to optimizing virtual machine allocation strategy based on user requirements for cloud data center," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 2, pp. 670–681, 2021.

[3] F. Dewangan, A. Y. Abdelaziz, and M. Biswal, "Load forecasting models in smart grid using smart meter information: A Review," *Energies*, vol. 16, no. 3, 1404, Jan. 2023. doi: 10.3390/EN16031404

[4] A. R. Madireddy and K. Ravindranath, "Dynamic virtual machine relocation system for energy-efficient resource management in the cloud," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 3, e7520, Feb. 2023. doi: 10.1002/CPE.7520

[5] E. Parvizi and M. H. Rezvani, "Utilization-aware energy-efficient virtual machine placement in cloud networks using NSGA-III meta-heuristic approach," *Cluster Computing*, vol. 23, no. 4, pp. 2945–2967, 2020.

[6] A. Ghasemi and A. T. Haghighat, "A multi-objective load balancing algorithm for virtual machine placement in cloud data centers based on machine learning," *Computing*, vol. 102, no. 9, pp. 2049–2072, 2020.
[7] A. Kaur *et al.*, "Algorithmic approach to virtual machine migration in cloud computing with updated SESA algorithm," *Sensors*, vol. 23, no. 13, 6117, Jul. 2023. doi: 10.3390/S23136117
[8] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *Journal of Network and Computer Applications*, vol. 66, pp. 106–127, May 2016. doi: 10.1016/J.JNCA.2016.01.011
[9] P. A. Malla and S. Sheikh, "Analysis of QoS aware energy-efficient resource provisioning techniques in cloud computing," *International Journal of Communication Systems*, vol. 36, no. 1, e5359, Jan. 2023. doi: 10.1002/DAC.5359
[10] J. Ahmadi, A. T. Haghighat, A. M. Rahmani, and R. Ravanmehr, "Confidence interval-based overload avoidance algorithm for virtual machine placement," *Software: Practice and Experience*, vol. 52, no. 10, pp. 2288–2311, Oct. 2022. doi: 10.1002/SPE.3127
[11] R. Zolfaghari, A. Sahafi, A. M. Rahmani, and R. Rezaei, "An energy-aware virtual machines consolidation method for cloud computing: simulation and verification," *Software: Practice and Experience*, vol. 52, no. 1, pp. 194–235, 2022.
[12] A. Yousefipour, A. M. Rahmani, and M. Jahanshahi, "Energy and cost-aware virtual machine consolidation in cloud computing," *Software: Practice and Experience*, vol. 48, no. 10, pp. 1758–1774, 2018.
[13] M. H. Shirvani, A. M. Rahmani, and A. Sahafi, "A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter: taxonomy and challenges," *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 3, pp. 267–286, 2020.
[14] M. Tarahomi, M. Izadi, and M. Ghobaei-Arani, "An efficient power-aware VM allocation mechanism in cloud data centers: A micro genetic-based approach," *Cluster Computing*, vol. 24, no. 2, pp. 919–934, 2021.
[15] M. Ghobaei-Arani, A. A. Rahmanian, M. Shamsi, and A. Rasouli-Kenari, "A learning-based approach for virtual machine placement in cloud data centers," *International Journal of Communication Systems*, vol. 31, no. 8, e3537, May 2018. doi: 10.1002/DAC.3537
[16] M. Ghobaei-Arani, M. Shamsi, and A. A. Rahmanian, "An efficient approach for improving virtual machine placement in cloud computing environment," *Journal of Experimental & Theoretical Artificial Intelligence* , vol. 29, no. 6, pp. 1149–1171, Nov. 2017. doi: 10.1080/0952813X.2017.1310308
[17] X. Ruan, H. Chen, Y. Tian, and S. Yin, "Virtual machine allocation and migration based on performance-to-power ratio in energy-efficient clouds," *Future Generation Computer Systems*, vol. 100, pp. 380–394, 2019.
[18] C. Wei, Z.-H. Hu, and Y.-G. Wang, "Exact algorithms for energy-efficient virtual machine placement in data centers," *Future Generation Computer Systems*, vol. 106, pp. 77–91, 2020.
[19] S. Talwani *et al.*, "Machine-learning-based approach for virtual machine allocation and migration," *Electronics*, vol. 11, no. 19, 3249, 2022.
[20] J. Ahmadi, A. T. Haghighat, A. M. Rahmani, and R. Ravanmehr, "A flexible approach for virtual machine selection in cloud data centers with AHP," *Software: Practice and Experience*, vol. 52, no. 5, pp. 1216–1241, May 2022. doi: 10.1002/SPE.3062
[21] M. S. A. Khan and R. Santhosh, "Hybrid optimization algorithm for VM migration in cloud computing," *Computers and Electrical Engineering*, vol. 102, 108152, Sep. 2022. doi: 10.1016/J.COMPELECENG.2022.108152
[22] F. Abdessamia, W. Z. Zhang, and Y. C. Tian, "Energy-efficiency virtual machine placement based on binary gravitational search algorithm," *Cluster Computing*, vol. 23, no. 3, pp. 1577–1588, Sep. 2020. doi: 10.1007/S10586-019-03021-0/METRICS
[23] M. Abdel-Basset, L. Abdle-Fatah, and A. K. Sangaiah, "An improved Lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment," *Cluster Computing*, vol. 22, no. 4, pp. 8319–8334, 2019.
[24] N. Rasouli, R. Razavi, and H. R. Faragardi, "EPBLA: Energy-efficient consolidation of virtual machines using learning automata in cloud data centers," *Cluster Computing*, vol. 23, no. 4, pp. 3013–3027, 2020.
[25] S. Azizi, M. Zandsalimi, and D. Li, "An energy-efficient algorithm for virtual machine placement optimization in cloud data centers," *Cluster Computing*, vol. 23, no. 4, pp. 3421–3434, 2020.
[26] A. S. Abohamama and E. Hamouda, "A hybrid energy-aware virtual machine placement algorithm for cloud environments," *Expert Systems with Applications*, vol. 150, 113306, 2020.
[27] M. H. Shirvani, "An energy-efficient topology-aware virtual machine placement in cloud datacenters: A multi-objective discrete JAYA optimization," *Sustainable Computing: Informatics and Systems*, vol. 38, 100856, 2023.
[28] Y. Meraihi, A. Ramdane-Cherif, D. Acheli, and M. Mahseur, "Dragonfly algorithm: A comprehensive review and applications," *Neural Computing and Applications*, vol. 32, pp. 16625–16646, 2020.
[29] M. Alshinwan *et al.*, "Dragonfly algorithm: A comprehensive survey of its results, variants, and applications," *Multimedia Tools and Applications*, vol. 80, no. 10, pp. 14979–15016, Apr. 2021. doi: 10.1007/S11042-020-10255-3/METRICS
[30] M. A. N. Saif, S. K. Niranjan, B. A. H. Murshed *et al.*, "Multi-objective Cuckoo Search Optimization Algorithm for optimal resource allocation in cloud environment," in *Proc. 2022 3rd International Conference for Emerging Technology (INCET)*, 2022, pp. 1–7.
[31] M. K. Hossain, M. Rahman, A. Hossain, S. Y. Rahman, and M. M. Islam, "Active idle virtual machine migration algorithm-a new ant colony optimization approach to consolidate virtual machines and ensure green cloud computing," in *Proc. ETCCE 2020, International Conference on Emerging Technology in Computing, Communication and Electronics*, Dec. 2020. doi: 10.1109/ETCCE51779.2020.9350915
[32] T. P. Shabeera, S. D. M. Kumar, S. M. Salam, and K. M. Krishnan, "Optimizing VM allocation and data placement for data-intensive applications in cloud using ACO metaheuristic algorithm," *Engineering Science and Technology, an International Journal*, vol. 20, no. 2, pp. 616–628, 2017.