Cat-Squirrel Optimization Algorithm for VM Migration in a Cloud Computing Platform

Ashok Kumar C., PSNA College of Engineering and Technology, India*
Sivakumar P., PSG College of Technology, India

ABSTRACT

This paper introduces an approach for the VM migration based on an optimization algorithm named CS in cloud. Selecting the provider is carried out with the usage of multiple constraints, like delay, bandwidth, cost, and load. Subsequently, the effective searching criteria are computed for finding the optimal service on the basis of fitness constraints. The searching criteria are formulated as optimization problems, which are tackled using CS. The proposed CS is designed by integrating CSO with the SSA such that the fitness function is evaluated for the optimal VM migration by considering several parameters, such as delay, cost, bandwidth, and load. Thus, the cloud manager will perform the migration of VM in cloud based on the proposed CS-based VM migration approach. The performance of the CS-based VM migration is evaluated in terms of delay, cost, and load. The proposed CS-based VM migration method achieves the minimal delay of 0.146, minimal cost of 0.052, and the minimal load of 0.182.

KEYWORDS

Cat Swarm Optimization, Cloud Computing, Migration Factor, Squirrel Search Algorithm, Virtual Machine Migration

1. INTRODUCTION

Recently, cloud data centers have been paid considerable attention in industrial as well as academic communities. Multiple Virtual Machines (VMs) are allowed for re-locating on a single Physical Machine (PM) based on resource virtualization. Nowadays, the leading enterprises, such as Amazon and Microsoft, employed data centers for providing applications, like scientific computations, storage of large data, and hosting several web services. To fix the security vulnerability, few of the data centers undergo maintenance process, and therefore the continuous services are not provided to VMs at the particular time slots (Wang, et al., 2017; Karthikeyan, et al., 2018). With the rapid growth of cloud services in large and small scale industries, the data center sizes increasing continuously. Thus, the data centers need several cooling devices to keep the data center at a particular temperature, resulting in increased Carbon di oxide (CO₂) emission and energy consumption (He, et al., 2019; Han, et al., 2019). The physical resources cost of the data center is 15%, and the cost of energy consumption...
is 45% (Narantuya, et al., 2018; Zakarya, 2018). Therefore, the factors mentioned above are very important for identifying energy-based resource allocation at the cloud data center to make a green data center (Kansal & Chana, 2016; Sharma & Reddy, 2016).

Cloud computing is implemented based on the distributed services, and it provides the virtualized resources like parallel semantic computing, distributed semantic models construction, semantic ambiguity, etc. The semantic web is employed to convert the World Wide Web into a structured intelligent web system and provide services to the users. The semantic web helps to express the domain’s knowledge and organize the Metadata in cloud computing. Moreover, it helps solve the big data distribution in cloud computing and encourages data sharing (Meshram, et al., 2016). With the expeditious growth of the cloud for computation, storage and networking, live migration of VMs has given rise to the need for effectively managing data centers (Singh, & Gupta, 2016; Patel, et al., 2019). VM is widely utilized for data center management. In addition, virtualization-based solutions are adopted for generating VMs based on users’ requests for computing resources, storage space, and the network bandwidth (Medina & García, 2014; Osanaiye, et al., 2017). When multiple VMs are available in a single server, the virtualization may enhance the underloaded server’s utilization, leading to less power consumption at the fewer servers (Satpathy, et al., 2018). However, the energy-efficient-based resource management for the virtualization data centers becomes an attractive research area (Zhang, et al., 2018; Han, et al., 2019). The migration of VMs from one data center to another has made it possible to conveniently maintain data centers without affecting much of the performance of VMs. In live migration, data from a physical machine (PM) is copied to destination PM in another data center while the VM continuously runs on the former PM. Once the data is copied, the VM is continued on the new PM. It ensures negligible downtime and thus achieves high performance. During the migration of VMs, clock synchronization becomes paramount. The clocks must be synchronized and at higher precision (Zhang, et al., 2019 ; Patel, et al., 2019).

The VMs allocation in the cloud is categorized into three: i) Static VM allocation approach is introduced for allocating the VMs on PMs at the particular time interval. After VMs expiry time, this framework is re-executed to destroys expired VMs. ii) Semi-Dynamic VMs allocation technique is executed at a specific time; iii) Dynamic-based VM allocation deals based on future predictions with on-demand resources allocation of workload. Thus, the techniques mentioned above have some demerits and merits, like the amount of energy consumption, resource utilization, and Service level agreement (SLA) violation (Sivagami & Easwarakumar, 2019; Sharma & Reddy, 2016). However, the energy-efficient VM management is partitioned into dynamic or static over time. As the real resource is essentially dynamic, the static approaches attain low resource utilization (Li, et al., 2016; Paulraj, et al., 2018). When the resource demands of VMs are dynamic, insufficient resources are provided to the VMs (Rodrigues, et al., 2017). Hence, the migration and allocation policies of VMs are adaptive by jointly considering resource shortage and energy consumption over time. When the dynamic resource is considered, the existing VM management approaches are centralized but suffer from scalability and robustness issues. Furthermore, the dynamic approaches (Beloglazov & Buyya, 2010; Chen & Shen, 2014; Bobroff, et al., 2007) are heuristics-based, but the theoretical performance was not achieved better (Zhang, et al., 2019 ; Han, et al., 2019).

The primary intention of this research is to develop an approach for VM migration in a cloud computing platform by proposing an optimization algorithm. At first, the cloud system is designed, and the migration agent repeatedly monitors the resource utilization and memory in the cloud. Then, the loads are balanced by migrating the VMs to handle the tasks. Here, a VM migration strategy is established using the proposed Cat squirrel (CS). Then, the other parameters, such as delay, cost, bandwidth, and load, are computed. In this case, the VM movement from PM to another PM is the optimization issue. It is performed based on developed CS, which combines the Cat swarm optimization (CSO) and Squirrel search algorithm (SSA).

The main contributions of the research paper towards the VM migration in the cloud computing environment are illustrated below:
• Computing the multi-objective model for choosing the suitable VM for migration using the parameters such as delay, cost, bandwidth, and load.

• Proposing CS algorithm by integrating CSO in SSA to select optimal VM to handle the cloud task effectively. CSO Algorithm is very effective because it has fast convergence and can spot complex and different problems in various areas. Similarly, SSA Algorithm has high robustness and stability. The conventional methods have the challenges like high delay, which increases the communication cost and energy consumption. The proposed Cs algorithm helps to migrate the VM with minimal delay, hence the cost and energy consumption is reduced and the performance of the system is further enhanced.

The research organization is arranged in the following manner: Section 2 reviews the existing VM migration techniques with challenges that remain the motivation for the research; section 3 elaborates the developed CS algorithm for the VM migration. The results of the developed model are discussed in section 4 and conclude the paper in section 5.

2. MOTIVATION

This section presents the literature survey of several methods utilized for VM migration in the cloud, and the challenges of existing works are discussed.

2.1 Literature Survey

Several methods related to VM migration are described and analyzed: Han, et al., 2019 developed dynamic VM management for solving large-scale Markov decision process (MDP) in the data centers. In addition, the method suffers from a dimensionality curse. Hence the MDP-driven dynamic VM management is termed Mad VM. The method failed to consider multiple dynamic resources, like network, memory, and Central processing unit (CPU). Zakarya, 2018 developed an approach for VM allocation in which the effective hosts were employed initially to minimize energy consumption. In this, Extended Energy-Aware Cost Recovery Approach is used for the VM migration. In addition, the migration scheme was introduced for migrating the VMs only to save energy. The method provides optimal energy efficiency but does not predict the runtime of VM. Sharma & Reddy, 2016 presented Euclidean distance-enabled multi-objective resources allocation for migrating the VMs. Additionally, the hybrid approach, named Genetic Algorithm and Particle Swarm Optimization (HGAPSO), was introduced to save energy consumption and reduce resource wastage at the cloud data center. However, the method does not achieve better performance. Wu, et al., 2016 modeled an improved grouping genetic algorithm (IGGA) in the heterogeneous cloud datacenters. Here, the consolidation score was defined for enabling overall evaluation with two objectives of saved power and migration costs. It effectively reduced the energy consumed in cloud data centers but failed to consider other algorithms for improving the system performance.

Patel, et al., 2019 developed enhanced Data center Time Protocol (DTP) and wireless Precision Time Protocol (PTP)-enabled clock synchronization approaches for achieving maximal precision at inter and intra-cloud data center networks. The method achieved high precision clock synchronization for on-demand live VM migrations. However, the overall performance enhancement for the inter-data center is very less. Ruan, et al., 2019 developed an approach for computing the levels of optimized working utilization for the host computers. Hence, the performance-to-power ratio (PPR) was introduced to measure the power data in a real-time environment. The method failed to use host utilization to conserve overall energy consumption. Soltanshahi, et al., 2019 developed the Krill herd approach for allocating the energy-aware VM in the cloud. Here, the energy efficiency was found better but does not consider deep learning approaches to improve system performance. Wang, et al., 2017 modeled an approach for migrating VMs to PMs in the data centers. Initially, the joint optimization issue with migration cost and the delay was formulated. Then, the heuristic approach was introduced...
for solving the general problem. At last, less migration cost was considered for migrating the VMs with minimal waiting delay. However, more VMs were needed to be served in the system to minimize the delay. (Pande, et al., 2021) devised a VM migration in the vehicular cloud using resource management. In this, resource utilization-aware VM migration (RU-VMM) is used to distribute loads with reduced energy consumption. They achieved better performance by avoiding unnecessary migrations. The drawback of the system is, it failed to consider the bandwidth and the delay in the network. (Prakash, et al., 2021) devised a VM migration using resource utilization. Osmotic Hybrid artificial Bee and Ant Colony with Future Utilization Prediction with Multipath Traffic Routing (OH-BAC-FUP-MTR) is used for congestion avoidance and efficient transmission. They achieved reduced SLA violations. The drawback of the system is, it is not considered the workload of the VM. (Mapetu, et al., 2020) developed Pearson Correlation algorithm in cloud computing using load balancing to reduce energy consumption. It has better SLA violations and low time complexity. It is not applicable for real time applications. (Dubey, et al., 2020) developed VM Placement Policy for the data center in cloud environment using the Efficient of Service Best Fit Decreasing (EBFD). In this, the execution time is reduced, and the VM placement failure rate is also decreased. However, it does not consider deep learning approaches to improve the overall system performance. The literature review of the existing method is included in table 1.

2.2. Problem Statement

A VM is a virtual environment that works like a computer within a computer. It runs on an isolated partition of its host computer with its resources of CPU power, memory, an operating system (e.g. Windows, Linux, macOS), and other resources. An efficient VM will provide better availability with fast convergence, minimum delay, and fast convergence. Another parameter that influences the VMs is their bandwidth limit. The bandwidth capacity limitation leads to less number of VMs migrating at the same time. Further, load balancing in VMs is an important parameter to improve the overall performance of the system. Then, costs may include data storage and transfer, migration interruptions, migration time, and the number of migrations. Hence, the cost may be minimized by using minimum load and minimum delay with proper resource allocation. Hence, there is a need for a technique to perform the VM migration effectively. The newly devised CS-based VM migration effectively migrates the VM by considering the load, bandwidth, cost, and load conditions.

3. CLOUD SETUP

This section illustrates the VM migration approach based on the CS optimization algorithm in a cloud computing platform. Figure 1 depicts the system model of VM migration in the cloud. In the last few decades, cloud computing has been paid great attention in computer science. The cloud computing model provides a flexible and simplest way to manage and retrieve files and data. However, the cloud model comprises several PMs to tackle user requests, and the PM collects the VMs to process the tasks dynamically. Thus, the VM is available in the cloud produced dynamically to reduce the bottleneck and the visualization problem. The user utilized the services as a task, and every task is subjected to VM based on the round-robin. In this case, the VM is controlled by PM; the cloud has a load balancer that checks the PM load. When the PM load value is higher than the threshold value, the migration of VM is done.

3.1 Algorithmic Procedure of VM Migration Algorithm

Initially, the number of PMs and VMs is initialized, and the maximum migration cost is maximum. The task is assigned in a round-robin pattern for efficient migration. Then it checks the load; if it exceeds the threshold, the migration is performed optimally using the fitness value. The steps continue until it reaches the optimal solution based on fitness value. The algorithmic procedure followed by the developed VM migration model is given below:
Table 1 Literature review of existing methods

| Author       | Methodology                                                                 | Advantages                                      | Limitations/Challenges                                                                 | Performance metrics                                                                 |
|--------------|-----------------------------------------------------------------------------|-------------------------------------------------|---------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| Han, et al., 2019 | Developed dynamic VM management for solving large-scale Markov decision process (MDP) in the data centers | Achieved better power consumption.              | Failed to consider multiple dynamic resources, like network, memory, and Central processing unit (CPU). Besides, the method suffers from a dimensionality curse. | MadVM has reduced power consumption by 23%, 47%, and 58.7% compared with CompVM, CloudScale, and Tetris techniques. |
| Zakarya, 2018 | Developed an approach for VM allocation in which the effective hosts using Extended Energy-Aware Cost Recovery Approach is used for the VM migration | The method provides optimal energy efficiency | Failed to predict the runtime of VM                                                  | Using 20 071 200 VMs, the best approach has 18.8% of VMs that are migratable, of which 99.1% recover their migration cost. |
| Sharma & Reddy, 2016 | The hybrid approach, named Genetic Algorithm and Particle Swarm Optimization (HGAPSO), was introduced to save energy consumption and reduce resource wastage at the cloud data center | Achieved better energy consumption and efficiently utilized the resources. | The optimal solution with multi-objectives creates a lot of challenges | The CPU utilization=8, RAM utilization=19, and storage utilization=3; using 100VMs and 60PMs for the heterogeneous network. |
| Wu, et al., 2016 | Modeled an improved grouping genetic algorithm (IGGA) in the heterogeneous cloud datacenters | It effectively reduced the energy consumed in cloud data centers | Failed to consider other algorithms for improving the system performance.            | The performance is evaluated based on saved power and energy consumption. Migration cost=0.08 ± 0.002, and saved power=854 ± 2.1 for 250VMs. |
| Patel, et al., 2019 | Developed enhanced Data center Time Protocol (DTP) and wireless Precision Time Protocol (PTP)-enabled clock synchronization approaches for achieving maximal precision at inter and intra-cloud data center networks | The method achieved high precision clock synchronization for on-demand live VM migrations | The overall performance enhancement for the inter-data center is very less          | Optimal precision is bounded by 0.80648 seconds for 62 connected nodes in Three-tier - VL2 network |
| Ruan, et al., 2019 | Developed an approach for computing the levels of optimized working utilization for the host computers | The performance-to-power ratio (PPR) was introduced to measure the power data in a real-time environment | The method failed to use host utilization to conserve overall energy consumption.     | The performance is evaluated in terms of migration and shutdown.                   |
| Soltanshahi, et al., 2019 | Developed the Krill herd approach for allocating the energy-aware VM migration in the cloud. | They achieved better energy efficiency.         | Failed to consider deep learning approaches to improve system performance.          | Energy consumption=49.25.                                                          |
| Wang, et al., 2017 | Modeled an approach for migrating VMs to PMs in the data centers. Initially, the joint optimization issue with migration cost and the delay was formulated. | The heuristic approach was used for solving the general problem. In addition, less migration cost was considered for migrating the VMs with minimal waiting delay. | More VMs were needed to be served in the system to minimize the delay.            | The performance is evaluated based on delay and migration costs.                   |
| Pande, et al., 2021 | Devised a VM migration in the vehicular cloud using resource management. In this, resource utilization-aware VM migration (RU-VMM) is used to distribute loads with reduced energy consumption. | They achieved better performance by avoiding unnecessary migrations | The drawback of the system is, it failed to consider the bandwidth and the delay in the network. | The performance is evaluated based on the number of final source vehicles, percentage of successful VM migrations, and percentage of dropped VM migrations. |

continued on following page
1. Initially, the cloud with $M$ VMs and $N$ PMs is initialized.
2. In this step, the cost of PM migration is initially set to high value, and therefore, the migration cost is equal to 1.
3. Consequently, the incoming tasks of VM are allocated using round-robin at the particular time duration.
4. After that, the load value of the VM is computed based on equation (2), and if the load value exceeds the value of the threshold, VM is migrated optimally using the proposed CS.
5. Find the other parameters, like resource availability, energy, and migration cost for the system.
6. Finally, the steps from (3) to (5) are repeated for every iteration, and the developed model is terminated at the end of the iteration when it satisfies the fitness value.

3.2 Cat Squirrel Optimization Algorithm for Virtual Machine Migration

The developed VM migration model based on the developed CS optimization in the cloud is described in this section. Figure 2 depicts the schematic view of the developed approach. First, the proposed CS algorithm is designed by integrating the CSO (Bahrami, et al., 2018) with the SSA (Jain, et al., 2019) for choosing the best VM. Here, the developed model is utilized to validate the status of load using each VM, and the task is reallocated to another VM when the load of the VM is greater than the specified threshold. After that, the developed model defines the novel fitness function with the help of various parameters, like delay, cost, bandwidth, and load. Thus, the developed model assigns the optimal tasks to the VM based on fitness measures.

3.2.1 Initialization Phase

Let us assume the cloud model with different VMs and PMs. The cloud model containing $Q$ number of PMs, represented as $Q = \{Q_1, Q_2, \ldots, Q_u, \ldots, Q_M\}; 1 \leq u \leq M$, and several VMs are available under each PM. In addition, the VM present in $u^{th}$ PM is indicated as, $P = \{P_1^u, P_2^u, \ldots, P_n^u \ldots P_N^u\}; 1 \leq n \leq N$, where, the $n^{th}$ VM in $u^{th}$ PM is denoted as $P_n^u$. Additionally, the individual user assigned the
requested task to every VM using round-robin and is represented as, $S = \{S_1, S_2, \ldots, S_v, \ldots, S_h\}$, where, the term $h$ indicates the total amount of tasks allocated to every VM user. The VM in the cloud consists of several parameters, like CPU utilization, memory, bandwidth, Million Instructions per Second (MIPS), delay, number of processing elements, and frequency scaling. In the cloud, the $n^{th}$ VM is present in $u^{th}$ PM, and is expressed as:

$$P_u^n = \{H_u^n, D_u^n, A_u^n, B_u^n, G_u^n, E_u^n, I_u^n\}$$

where, the term $H_u^n$ denotes the total amount of processing elements of $n^{th}$ VM in $u^{th}$ PM, and the $D_u^n$ refer to the number of CPUs used by $n^{th}$ VM in $u^{th}$ PM. The bandwidth is denoted as $A_u^n$, and $B_u^n$ refers to the memory of $n^{th}$ VM in $u^{th}$ PM. The total number of MIPS utilized by $n^{th}$ VM in $u^{th}$ PM is denoted as $G_u^n$, the symbol $E_u^n$ signifies the delay utilized by $n^{th}$ VM in $u^{th}$ PM, and the term $I_u^n$ indicates the frequency scaling of $n^{th}$ VM in $u^{th}$ PM. All the parameters employed during VM migration are acquiring the value ranges from 1 to 10.

### 3.2.2 Evaluation of Load

The load is estimated based on utilized resources for processing tasks by VM obtained from the user. The MIPS, CPU utilization, memory, network bandwidth, delay, the total number of processing elements, and frequency scaling are used to compute the load in the cloud platform. The load of VM is given by:

$$Load(K_n) = \sum_{g=1}^{3} \frac{R^d}{l}$$

where, the term $R^d$ indicates the resource utilization, and the total number of user tasks is denoted as $l$. The resource utilization is given by:

$$R^d = \frac{1}{F} \frac{H^F}{\max(H^F)} + \frac{D^F}{\max(D^F)} + \frac{A^F}{\max(A^F)} + \frac{B^F}{\max(B^F)} + \frac{G^F}{\max(G^F)} + \frac{E^F}{\max(E^F)} * \frac{1}{J}$$

where, the normalizing factor is denoted as $J$, and the term $H^F$ refers to the total amount of processing elements. The total amount of CPUs utilized is represented as $D^F$, and the bandwidth is represented as $A^F$. The MIPS is indicated as $G^F$, the delay is indicated as $E^F$, and the maximum value is represented as $\max()$. Then, the load of PM expression is given by:

$$Load(K_n) = \frac{1}{N} \sum_{n=1}^{M} (K_n)$$

### 3.2.3 Migration Factor

It is defined as the summation of delay and migration cost to the total number of normalizing factors. The formula for computing the migration cost is expressed as:
\[ Y = \frac{E + L}{N} \]  

where, the delay is denoted as \( E \), the migration cost is indicated as \( L \), and the normalizing factor is indicated as \( N \).

### 3.2.4 Delay

Delay is utilized to decide for waiting for their PM back, but the huge waiting delay is incurred and is intolerable for users, as the latency or delay is quite important for QoS of VMs, and the delay equation is expressed as:

\[
E = \sum_{q=1}^{k} \sum_{n=1}^{N} \left[ 1 - \sum_{j=1}^{N} y_{unj} \right] e_{u} + \sum_{Q_j \in Q^N} y_{unj} e_{uj} \]

Let us consider, \( Q^o = \{Q^o_1, Q^o_2, ..., Q^o_j \} \) be the set of PMs that are overloaded, and \( Q^N = \{Q^N_1, Q^N_2, ..., Q^N_j \} \) represent the set of unloaded PMs. The maintenance duration of \( u^h \) VM (constant) is denoted as \( e_u \), and the term \( e_{uj} \) refers to the delay during the migration of VM from \( u \) to \( j \). The variant \( y_{unj} \) satisfies the condition given below:

\[
y_{unj} = \begin{cases} 
1 & \text{if } P_{unj} \text{ is migrated to unloaded PM} \\
0 & \text{Otherwise} 
\end{cases} \]

### 3.2.5 Migration Cost

It is defined as the cost required to migrate the user requests from one VM to another VM. If the load of the system is higher then, the VM is migrated. The cost of the migration should be lower for an effective system. The migration cost is calculated as:

\[
L = \frac{1}{M} \sum_{u=1}^{M} \frac{j}{f + N} 
\]

where, \( f \) is the migration constant, \( j \) signifies the number of migrations of VMs and \( N \) refers to the number of VMs, and \( M \) is the number of PMs.

### 3.3 Proposed Cat Squirrel Optimization for VM Migration

In this section, the VM migration approach is elaborated using the proposed CS-based VM migration approach to migrate the VM in a cloud computing platform. The proposed CS is designed by combining CSO with SSA. SSA (Jain, et al., 2019) is the simple and powerful nature-inspired approach to tackle optimization issues. This approach uses the dynamic foraging strategy of the southern flying squirrels, and the way of locomotion is termed gliding. CSO (Bahrami, et al., 2018) is one of the Swarm Intelligence (SI)-enabled optimization approaches duly based on the cat’s behavior and is utilized to solve several optimization issues. In the CSO, the cat’s population is initially distributed and created randomly in M-dimensional solution space, and every cat indicating the solution. Then,
the population is partitioned into two subgroups. The first group is the seeking mode, which means resting and keeping an eye on surroundings. The second one is the tracking mode, where the cat starts moving around and chasing the prey. Consequently, the combination of two groups helps to obtain the global solution in M-dimensional search space. When the cats spend very little time tracing mode, the total amount of cats in tracing is small, and the Mixture Value (MR) is utilized to define the number with a small value. Once the cats are arranged into two modes, the new positions and the objective functions are obtained from which the optimal solution is saved in memory. Finally, the above steps are repeated until they reached the maximal solutions. Thus, the integration of CSO in SSA tunes the associated parameters for improving the convergence speed and algorithmic performance to obtain a globally optimum solution. Besides, the exploitation and the exploration phase trade off are balanced by incorporating the SSA with the CSO algorithm. Thus the global search process is enhanced. The solution encoding, fitness function, and proposed CS-based VM migration algorithm are explained below.

3.3.1 Solution Encoding

The proposed CS-based VM migration algorithm selected the VM that are suitable for migration. Let us assume that PM 1 consists of 3 VMs and PM 2 having 2 VMs. The incoming tasks are assigned to each VM using a round-robin fashion. If the VM exceeds the threshold value, then the VM is migrated to the underloaded VMs, and the optimal VM for the migration is assigned using an optimization algorithm. Figure 3 depicts the solution encoding of the developed model to identify the optimal VM for migration. Here, the solution represents the VM, denoted as \( W = \{W_1, W_2, \ldots, W_j\} \), where \( j \) denotes the total number of underloaded VMs. Hence, on the \( K \) underloaded VMs, the overloaded VMs that are to be migrated will be selected optimally using the optimization algorithm.

3.3.2 Fitness Evaluation

The fitness function is calculated for determining the best solution from the solution set. The fitness function of the CS-based VM migration is formulated in terms of three parameters, like load, migration factor, and bandwidth. The fitness is a maximization function, and thus the optimal VM is chosen for execution. The fitness function of the proposed model is expressed by:

\[
Y = \frac{\sum_{a=1}^{M} K_u + Y + A}{3}
\]

(9)

where, load of PM is denoted as \( K_u \), migration factor is denoted as \( Y \), and the bandwidth for service is indicated as \( A \). The expression of bandwidth is given below:

Figure 3. Solution encoding
\[ A = \sum_{u=1}^{M} \sum_{n=1}^{N} (BW_{wn} + BW^N) \]  

where, \( BW_{wn} \) refer to the bandwidth \( n^{th} \) VM in \( u^{th} \) PM, and \( BW^N \) signifies the bandwidth for migration, value to be fixed constant.

### 3.3.3 Algorithmic Procedure of the Proposed Cat Squirrel-Based VM Migration Approach

The algorithmic steps of the proposed model to select the best VM for migration is illustrated below:

1. **Initialization:** In the initial step, the optimization parameters, including the population cats, are initialized, which includes: \( \{Z_{vw}, 1 \leq v \leq o; 1 \leq w \leq y\} \), where, \( o \) refer to the population size, and the dimension is indicated as \( y \).

2. **Evaluation of fitness function:** The fitness for each solution is computed based on the fitness function depicted in equation (9). The fitness function is taken as a maximization function, and the solution producing the maximum fitness is considered as the best solution.

3. **Position update using Cat squirrel optimization:** The solution is classified into two groups: seeking mode and the tracing mode for solution update. In the seeking or resting mode, the cat is resting with keeping its eyes on the environment. Here, if the cat senses the danger, the cat decides to move cautiously and slowly. While resting, the cat observed to M-dimensional solution space for deciding its next move, seeking memory pool (SMP), seeking a range of the selected dimension (SRD), counts of dimension to change (CDC), and self-position consideration (SPC) are the four parameters utilized in this optimization algorithm. According to its velocity, the cat’s movement is very fast in tracing mode while chasing prey or any moving object. The standard equation of tracing mode is given by:

   \[ Z_{vw} (\tau + 1) = Z_{vw} (\tau) + h_{vw} \]  

   \[ Z_{vw} (\tau + 1) = Z_{vw} (\tau) + h_{vw} + m_{1}x_{1} (Z_{best,v} - Z_{vw} (\tau)) \]  

   \[ Z_{vw} (\tau + 1) = Z_{vw} (\tau) + h_{vw} + m_{1}x_{1}Z_{best,v} - m_{1}x_{1}Z_{vw} (\tau) \]  

   \[ Z_{vw} (\tau + 1) = Z_{vw} (\tau) (1 - m_{1}x_{1}) + h_{vw} + m_{1}x_{1}Z_{best,v} \]  

The equation (14) is modified using SSA, hence the standard equation of SSA is expressed as:

\[ Z_{vw} (\tau + 1) = Z_{vw} (\tau) + f_{a}T_{y} \times (Z_{best,v} - Z_{vw} (\tau))C_{1} \]
where, the gliding distance is denoted as \( f_d \), the random number is represented as \( C_1 \) ranging between 0 and 1, the term \( \tau \) denotes the current iteration. As \( FS_{vw}^\tau \) in SSA refer to the position of the flying squirrel that reached the hickory nut tree, which is the optimal solution. Hence, it can be considered as the best solution:

\[
FS_{vw}^\tau = Z_{best,w}
\] (16)

\[
Z_{vw} (\tau + 1) = Z_{vw} (\tau) + f_d T_y Z_{best,w} C_1 - f_d T_y Z_{vw} (\tau) C_1
\] (17)

\[
f_d T_y Z_{best,w} C_1 = Z_{vw} (\tau + 1) - Z_{vw} (\tau) + f_d T_y Z_{vw} (\tau) C_1
\] (18)

\[
Z_{best,w} = \frac{Z_{vw} (\tau + 1) - Z_{vw} (\tau) + f_d T_y Z_{vw} (\tau) C_1}{f_d T_y C_1}
\] (19)

Substituting equation (19) in equation (14), the solution becomes:

\[
Z_{vw} (\tau + 1) = Z_{vw} (\tau) \left(1 - m_1 x_1\right) + h_{vw} + m_1 x_1 \frac{Z_{vw} (\tau + 1) - Z_{vw} (\tau) + f_d T_y Z_{vw} (\tau) C_1}{f_d T_y C_1}
\] (20)

\[
Z_{vw} (\tau + 1) = Z_{vw} (\tau) \left(1 - m_1 x_1\right) + h_{vw} + m_1 x_1 \frac{Z_{vw} (\tau + 1) + f_d T_y Z_{vw} (\tau) C_1 - Z_{vw} (\tau)}{f_d T_y C_1}
\] (21)

\[
Z_{vw} (\tau + 1) - m_1 x_1 \frac{Z_{vw} (\tau + 1)}{f_d T_y C_1} = Z_{vw} (\tau) \left(1 - m_1 x_1\right) + h_{vw} + m_1 x_1 \frac{f_d T_y Z_{vw} (\tau) C_1 - Z_{vw} (\tau)}{f_d T_y C_1}
\] (22)

\[
Z_{vw} (\tau + 1) \left(1 - \frac{m_1 x_1}{f_d T_y C_1}\right) = Z_{vw} (\tau) \left(1 - m_1 x_1\right) + h_{vw} + m_1 x_1 \frac{f_d T_y Z_{vw} (\tau) C_1 - Z_{vw} (\tau)}{f_d T_y C_1}
\] (23)

\[
Z_{vw} (\tau + 1) \left(1 - \frac{f_d T_y C_1 - m_1 x_1}{f_d T_y C_1}\right) = Z_{vw} (\tau) \left(1 - m_1 x_1\right) + h_{vw} + m_1 x_1 \frac{f_d T_y Z_{vw} (\tau) C_1 - Z_{vw} (\tau)}{f_d T_y C_1}
\] (24)
\begin{align}
Z_{vw}(\tau + 1) &= \frac{f_d T_y C_1}{f_d T_y C_1 - m_1 x_1} \left[ Z_{vw}(\tau)(1 - m_1 x_1) + m_1 x_1 \frac{f_d T_y Z_{vw}(\tau) C_1 - Z_{vw}(\tau)}{f_d T_y C_1} + h_{vw} \right] \\
Z_{vw}(\tau + 1) &= \frac{f_d T_y C_1}{f_d T_y C_1 - m_1 x_1} \left[ Z_{vw}(\tau)(1 - m_1 x_1) + m_1 x_1 \frac{f_d T_y Z_{vw}(\tau) C_1 - m_1 x_1 Z_{vw}(\tau)}{f_d T_y C_1} + h_{vw} \right] \\
Z_{vw}(\tau + 1) &= \frac{f_d T_y C_1}{f_d T_y C_1 - m_1 x_1} \left[ Z_{vw}(\tau)(1 - m_1 x_1 + m_1 x_1) - m_1 x_1 Z_{vw}(\tau) \right] \\
Z_{vw}(\tau + 1) &= \frac{f_d T_y C_1}{f_d T_y C_1 - m_1 x_1} \left[ Z_{vw}(\tau)(1 - m_1 x_1) \right] + h_{vw} 
\end{align}

where, the term $h_{vw}$ is denoted as the velocity of $v^{th}$ cat in $w^{th}$ dimension, the term $Z_{best,w}$ represents the cat’s best solution, and the location of cat is denoted as $Z_{vw}(\tau)$. The term $m_1$ signifies the constant, and the term $x_1$ refers to the random number ranging from 0 to 1. The term $T_y$ indicates the gliding constant, which is utilized for balancing the exploitation and exploration phase. The value of $T_y$ is 1.9 that is achieved after the rigorous analysis.

4. **Checking the feasibility of the solution**: The feasibility of the solution is computed based on the fitness function. If the newly generated solution is better than the previous one, it is changed by the new solution.

5. **Termination**: The above steps are repeated until the termination condition reaches. Here, the maximum number of iteration is considered as the termination condition. Thus, the optimization algorithm discussed in this section aims at determining the optimal weights for VM migration. The pseudo-code of the developed CS-based VM migration is depicted in Algorithm 1, which demonstrates a step-wise description of the algorithm.

4. RESULTS AND DISCUSSION

The results of the proposed CS-based VM migration are discussed in this section based on the evaluation metrics.
The experimentation of the CS-based VM migration is performed on Windows 10 OS with 4GB RAM and Intel i3 core processor. The implementation of the proposed method was done using Python. Python is easy to implement, because it is easy to code and can run the program anywhere.

### Table 2. Algorithm 1: Pseudocode for the proposed CS-based VM migration

```
Input: Cat swarm population \( Z_{vw}, 1 \leq v \leq n; 1 \leq w \leq y \)

Output: Best solution

Procedure:
Begin
  Population initiation: \( Z_{vw}, 1 \leq v \leq n; 1 \leq w \leq y \), and read all the parameters
  While \( \tau < \text{max gen} \)
    Compute the fitness values for all cats and arranged them
    \( Z_{best,vw} \) is the cat best position
    For \( j = 1 : M \)
      If SPC=1
        Start seeking mode
      Else
        Update the tracing mode equation using (29)
      End if
    End for
  End while
  Check the feasibility of solutions
  Return the best solution
  \( \tau = \tau + 1 \)
End while
Optimal solution is obtained
End
```

### 4.1 Experimental Arrangement
The experimentation of the CS-based VM migration is performed on Windows 10 OS with 4GB RAM and Intel i3 core processor. The implementation of the proposed method was done using Python. Python is easy to implement, because it is easy to code and can run the program anywhere.
4.2 Evaluation Metrics
The proposed CS-based VM migration performance is employed for analyzing the methods using delay, cost, and load. The reason for choosing the performance metrics is because many of the existing methods used these metrics.

**Delay:** It is the waiting time of the VM for the migration. It is expressed in equation (6).

**Cost:** It is defined as the cost required to migrate the user requests from one VM to another VM. It is expressed in equation (8).

**Load:** The load is estimated based on utilized resources for processing tasks by VM obtained from the user. It is expressed in equation (4).

4.3 Simulation Setup
The cloud environment employed for simulation contains 10 PMs and 50 VMs, and the total incoming tasks vary as 100, 200, 300, and 400.

4.4 Competing Methods
The performance of the developed method is analyzed by comparing the developed model with existing methods, like the hybrid approach of Genetic Algorithm and Particle Swarm Optimization (HGAPSO), Improved Grouping Genetic Algorithm (IGGA) (Wu, et al., 2016), Heuristic algorithm (Wang, et al., 2017), PBPSO-DCO (Mapetu, et al., 2020), and RU-VMM(Pande, et al., 2021) respectively. The detailed description of the state of art techniques are given below:

**HGAPSO:** It is developed for the reduction of the energy consumption and to reduce the wastage of resource.

**IGGA:** It is designed to reduce the power and the migration cost.

**Heuristic algorithm:** It is modeled for the evaluation of the migration cost and delay.

**PBPSO-DCO:** It is developed for the energy consumption by considering the load balancing technique.

**RU-VMM:** It is designed for the reduction of the utilized energy.

4.5. Comparative Analysis
The comparative analysis of the proposed CS-based VM migration with the conventional methods in terms of delay, cost, and load parameters is evaluated. The analysis is performed by varying the iterations.

4.5.1. Analysis With Incoming Task=100
The analysis of methods in terms of delay, cost, and load by varying the iterations with incoming task=100 is shown in figure 4. The analysis of methods in terms of delay parameter is illustrated in figure 4a). When the iteration=90, the delay computed by existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration is 0.246, 0.226, 0.19, 0.183, 0.170 and 0.167. The analysis of methods in terms of cost is illustrated in figure 4b). For 90th iteration, the cost computed by existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration is 0.093, 0.077, 0.077, 0.070, 0.066, 0.060 and 0.052. The analysis of methods based on load is illustrated in figure 4c). For 50th iteration, the load computed by existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration are 0.108, 0.098, 0.080, 0.243, 0.221 and 0.075, respectively.
4.5.2. Analysis With Incoming Task=200

The analysis of methods in terms of delay, cost, and load by varying the iterations with incoming task=200 is deliberated in figure 5. The analysis of methods in terms of delay parameter is illustrated in figure 5a). When the iteration=90, the delay computed by existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration are 0.241, 0.219, 0.201, 0.173, 0.162 and 0.151. The analysis of methods in terms of cost parameter is illustrated in figure 5b). For 90th iteration, the cost computed by existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration are 0.109, 0.061, 0.058, 0.058, 0.058 and 0.057. The analysis of methods based on load parameter is illustrated in figure 5c). For 80th iteration, the load computed by existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration are 0.276, 0.272, 0.263, 0.263, 0.262 and 0.261, respectively.

4.5.3. Analysis With Incoming Task=300

The analysis of methods in terms of delay, cost, and load by varying the iterations with incoming task=300 is deliberated in figure 6. The analysis of methods based on delay parameter is illustrated in figure 6a). when the iteration=90, the delay computed by existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration are 0.235, 0.22, 0.188, 0.172, 0.163 and 0.152. The analysis of methods in terms of cost parameter is illustrated in figure 6b). For 90th iteration, the cost computed by existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration are 0.095, 0.082, 0.068, 0.064, 0.061 and 0.058. The analysis of methods in terms of load parameter is illustrated in figure 6c). For 50th iteration, the load computed by existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration are 0.354, 0.341, 0.309, 0.308, 0.304 and 0.302, respectively.
Figure 5. Comparative analysis by varying the number of iterations with incoming task=200 (a) Delay, (b) Cost, (c) Load

Figure 6. Comparative analysis by varying the number of iterations with incoming task=300 (a) Delay, (b) Cost, (c) Load
4.5.4. Analysis With Incoming Task=400

The analysis of methods in terms of delay, cost, and load by varying the iterations with incoming task=400 is deliberated in figure 7. The analysis of methods in terms of delay parameter is illustrated in figure 7a). When the iteration=90, the delay computed by existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration are 0.234, 0.231, 0.177, 0.162, 0.152 and 0.146. The analysis of methods in terms of cost parameter is illustrated in figure 7b). For 90th iteration, the cost computed by existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration are 0.103, 0.095, 0.067, 0.064, 0.061 and 0.059. The analysis of methods in terms of load parameter is illustrated in figure 7c). For 70th iteration, the load computed by existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration are 0.331, 0.302, 0.282, 0.282, 0.281 and 0.280, respectively.

4.6 Comparative Analysis by Varying the Task Size

Figure 8 depicts the comparative analysis of the proposed CS-based VM migration with the existing techniques like HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, and RU-VMM in terms of the delay, cost, and load parameters by varying the incoming task size with the fixed iteration as 80. Figure 8.a) illustrates the comparative analysis in terms of the parameter delay. For task size 100, the delay evaluated by like HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration is 0.2530, 0.2404, 0.2090, 0.1983, 0.1840, and 0.1748 respectively. Similarly for task size 400, the delay evaluated by like HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration is 0.2450, 0.2320, 0.1780, 0.1635, 0.1523, and 0.1490 respectively. Figure 8.b) illustrates the comparative analysis in terms of the parameter cost.

Figure 7. Comparative analysis by varying the number of iterations with incoming task=400(a) Delay, (b) Cost, (c) Load
For task size 100, the delay evaluated by like HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration is 0.0944, 0.0784, 0.0704, 0.0697, 0.0690, and 0.0689 respectively. Similarly for task size 400, the cost evaluated by like HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration is 0.1098, 0.1016, 0.0743, 0.0722, 0.0702, and 0.0602 respectively. Figure 8.c) illustrates the comparative analysis in terms of the parameter load. For task size 100, the load evaluated by like HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration is 0.2932, 0.2913, 0.2686, 0.2633, 0.2589, and 0.2543 respectively. Similarly for task size 400, the load evaluated by like HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration is 0.2888, 0.2822, 0.2553, 0.2462, 0.2431, and 0.2393 respectively. From the above analysis it is clear that the proposed CS-based VM migration method outperformed other existing techniques for all the performance metric evaluation. Moreover, while increasing the task size the delay associated with the proposed CS-based VM migration is reduced. Similarly for the other metrics like cost and load, the performance enhanced while increasing the task size.

4.7 Comparative Discussion
Table 3 depicts the comparative discussion of previous HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, RU-VMM, and proposed CS-based VM migration in terms of delay, cost, and load parameters with incoming tasks 100, 200, 300, and 400 with different iterations. The minimum performance measured by proposed CS-based VM migration in terms of delay parameter is 0.146, whereas the delay values of existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, and RU-VMM are 0.234, 0.231, 0.177, 0.162, and 0.152, respectively. The minimal cost achieved by the proposed CS-
based VM migration with value of 0.052, whereas the existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, and RU-VMM acquired the cost of 0.093, 0.077, 0.070, 0.066, and 0.060, respectively. The minimal load value computed by proposed CS-based VM migration is 0.183, whereas the existing HGAPSO, IGGA, heuristic algorithm, PBPSO-DCO, and RU-VMM methods acquired the load of 0.269, 0.252, 0.235, 0.211, and 0.201, respectively.

4.8. Managerial Applications

The proposed CS-based VM migration in cloud computing is widely used in several managerial related applications, which are detailed below:

**Resource Management:** By using efficient algorithm for the VM migration with minimal cost, delay and load leads to efficient utilization of resources like energy.

**Traffic Management:** In cloud computing services, the network congestion occurs which leads to bottle neck of the network. The optimal placing of the VM reduces the network congestion by choosing the optimal path.

**Business Management:** The medium and small sized organizations for the efficient storage information and helps in decision making process.

5. CONCLUSION

This research paper presents the VM migration strategy using the novel optimization algorithm, termed CS-based VM migration in cloud computing model. The developed CS is designed by computing CSO in SSA. The factors, such as delay, cost, bandwidth, and load are considered for the VM placement. The fitness function is determined by considering the parameters, like migration factor, bandwidth, and load is carried out with the VM having best fitness function. Accordingly, the searching criteria are computed to identify the best service based on constraints. The VM migration model employs the developed CS optimization approach to assign VM. The developed CS optimization is the nature-inspired algorithm that depends on both the properties of CSO and SSA. The simulation of developed
VM migration strategy with CS optimization algorithm utilized the cloud model with 10 PM as well as 50 VM. The performance of the CS is measured in terms of delay, cost, and load with incoming tasks 100, 200, 300, and 400. The developed CS-based VM migration method achieves the minimal delay of 0.146, minimal cost of 0.052, and the minimal load of 0.183 that indicates the superiority of proposed method. In the future, new algorithms will be developed to provide better VM migration.

FUNDING INFORMATION

The publisher has waived the Open Access Publication fee for this article.
REFERENCES

Bahrami, M., Haddad, O. B., & Chu, X. (2018). Cat swarm optimization (CSO) algorithm. In Advanced Optimization by Nature-Inspired Algorithms (pp. 9–18). Springer.

Beloglazov, A., & Buyya, R. (2010). Energy efficient resource management in virtualized cloud data centers. Proceedings of the 10th IEEE/ACM international conference on cluster, cloud and grid computing, 826-831.

Bobroff, N., Kochut, A., & Beatty, K. (2007). Dynamic placement of virtual machines for managing sla violations. IFIP/IEEE International Symposium on Integrated Network Management, 119-128.

Chen, L., & Shen, H. (2014). Consolidating complementary VMs with spatial/temporal-awareness in cloud datacenters. IEEE INFOCOM 2014-IEEE Conference on Computer Communications, 1033-1041.

Dubey, K., Nasr, A.A., Sharma, S.C., El-Bahnasawy, N., Attiya, G., & El-Sayed, A. (2020). Efficient VM Placement Policy for Data Centre in Cloud Environment. Soft Computing: Theories and Applications. Advances in Intelligent Systems and Computing, 1053.

Han, Z., Tan, H., Wang, R., Chen, G., Li, Y., & Lau, F. C. M. (2019). Energy-Efficient Dynamic Virtual Machine Management in Data Centers. IEEE/ACM Transactions on Networking, 27(1), 344–360.

He, T. Z., Toosi, A. N., & Buyya, R. (2019). Performance evaluation of live virtual machine migration in SDN-enabled cloud data centers. Journal of Parallel and Distributed Computing, 131, 55–68.

Jain, M., Singh, V., & Rani, A. (2019). A novel nature-inspired algorithm for optimization: Squirrel search algorithm. Swarm and Evolutionary Computation, 44, 148–175.

Kansal, N. J., & Chana, I. (2016). Energy-aware Virtual Machine Migration for Cloud Computing - A Firefly Optimization Approach. Journal of Grid Computing, 14, 327–345.

Karthikeyan, K., Sunder, R., Shankar, K., Lakshmanaprabu, S. K., Vijayakumar, V., Elhoseny, M., & Manogaran, G. (2018). Energy consumption analysis of Virtual Machine migration in cloud using hybrid swarm optimization (ABC–BA). The Journal of Supercomputing, 1–17.

Li, H., Zhu, G., Cui, C., Tang, H., Dou, Y., & He, C. (2016). Energy-efficient migration and consolidation algorithm of virtual machines in data centers for cloud computing. Computing, 98, 303–317.

Mapetu, J. P. B., Kong, L., & Chen, Z. (2020). A dynamic VM consolidation approach based on load balancing using Pearson correlation in cloud computing. The Journal of Supercomputing, 77(12).

Medina, V., & García, J. M. (2014). A survey of migration mechanisms of virtual machines. ACM Computing Surveys, 46(3), 1–33.

Meshram, R.A., Hole, K.R., & Deshmukh, P.P., & Karwa, R.A. (2016). Review: Web Semantics in Cloud Computing. International Journal of Scientific and Engineering Research, 7.

Narantuya, J., Zhang, H., & Lim, H. (2018). Service-Aware Cloud-to-Cloud Migration of Multiple Virtual Machines. IEEE Access: Practical Innovations, Open Solutions, 6, 76663–76672.

Osanaiye, O., Chen, S., Yan, Z., Lu, R., Choo, K. K. R., & Dlodlo, M. (2017). From cloud to fog computing: A review and a conceptual live VM migration framework. IEEE Access: Practical Innovations, Open Solutions, 5, 8284–8300.

Pande, S.K., Panda, S.K., Das, S., Sahoo, K.S., Luhach, A.K., Jhanji, N.Z., Alroobaee, R., & Sivanesan, S. (2021). A Resource Management Algorithm for Virtual Machine Migration in Vehicular Cloud Computing. Computers, Materials & Continua, 67(2).

Patel, Y.S., Page, A., Nagdev, M., Choupey, A., Misra, R., & Das, S. K. (2019). On demand clock synchronization for live VM migration in distributed cloud data centers. Journal of Parallel and Distributed Computing.

Paulraj, G. J. L., Francis, S. A. J., Peter, J. D., & Jebadurai, I. J. (2018). Resource-Aware Virtual Machine Migration in IoT Cloud. Future Generation Computer Systems, 85, 173–183.

Prakash, R. G., Shankar, R., & Duraisamy, S. (2021). Resource utilization prediction with multipath traffic routing for congestion-aware VM migration in cloud computing. Indian Journal of Science and Technology, 14(7).
Rodrigues, T. G., Suto, K., Nishiyama, H., & Kato, N. (2017). Hybrid Method for Minimizing Service Delay in Edge Cloud Computing through VM Migration and Transmission Power Control. *IEEE Transactions on Computers*, 66.

Ruan, X., Chen, H., Tian, Y., & Yin, S. (2019). Virtual machine allocation and migration based on performance-to-power ratio in energy-efficient clouds. *Future Generation Computer Systems*, 100, 380–394.

Satpathy, A., Addya, S. K., Turuk, A. K., Majhi, B., & Sahoo, G. (2018). Crow search based virtual machine placement strategy in cloud data centers with live migration. *Computers & Electrical Engineering*, 69, 334–350.

Sharma, N. K., & Reddy, G. R. M. (2016). Multi-objective energy efficient virtual machines allocation at the cloud data center. *IEEE Transactions on Services Computing*, 12(1), 158–171.

Singh, G., & Gupta, P. (2016). A review on migration techniques and challenges in live virtual machine migration. *Proceedings of 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*.

Sivagami, V. M., & Easwarakumar, K. S. (2019). An Improved Dynamic Fault Tolerant Management Algorithm during VM migration in Cloud Data Center. *Future Generation Computer Systems*, 98, 35–43.

Soltanshahi, M., Asemi, R., & Shaﬁei, N. (2019). Energy-aware virtual machines allocation by krill herd algorithm in cloud data centers. *Heliyon*, 5(7), 02066.

Wang, X., Chen, X., Yuen, C., Wu, W., Zhang, M., & Zhan, C. (2017). Delay-cost tradeoff for virtual machine migration in cloud data centers. *Journal of Network and Computer Applications*, 78, 62–72.

Wu, Q., Ishikawa, F., Zhu, Q., & Xia, Y. (2016). Energy and migration cost-aware dynamic virtual machine consolidation in heterogeneous cloud datacenters. *IEEE Transactions on Services Computing*.

Zakarya, M. (2018). An extended energy-aware cost recovery approach for virtual machine migration. *IEEE Systems Journal*, 13(2), 1466–1477.

Zhang, F., Liu, G., Fu, X., & Yahyapour, R. (2018). A Survey on Virtual Machine Migration: Challenges, Techniques and Open Issues. *IEEE Communications Surveys and Tutorials*, 20(2).

Zhang, F., Liu, G., Zhao, B., Kasprzak, P., Fu, X., & Yahyapour, R. (2019). CBase: Fast Virtual Machine storage data migration with a new data center structure. *Journal of Parallel and Distributed Computing*, 124, 14–26.

Zhang, X., Wu, T., Chen, M., Wei, T., Zhou, J., Hu, S., & Buyya, R. (2019). Energy-aware virtual machine allocation for cloud with resource reservation. *Journal of Systems and Software*, 147, 147–161.