Received: June 28, 2021. Revised: August 7, 2021.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Genetic-Based Multi-objective Task Scheduling Algorithm in Cloud Computing Environment

Farouk A. Emara[1]* **Ahmed. A. A. Gad-Elrab[1,2]** **Ahmed Sobhi[1]** **K. R. Raslan[1]**

[1] *Faculty of Science, Al-Azhar University - Cairo, Egypt*
[2] *Faculty of Computing and Information Technology, King Abdul-Aziz University, Jeddah, Saudi Arabia*
Corresponding author's Email: aly_emara86@azhar.edu.eg

Abstract: Recently**,** resource management is the major issues in cloud computing (CC) environment because of dynamic heterogeneity of cloud computing environment. The task scheduling and virtual machines (VMs) allocation play a vital role in resources management of CC. Most of existing works for these issues aim to achieve single objective as maximizing resource utilization, load balancing, or power management. Currently, the big challenge in CC is building task scheduling and virtual machines (VMs) allocation algorithms that consider all these objectives in the same time. This problem is called task scheduling with VMs allocation multi-objectives problem which is NP completeness problem. In this paper, a new task scheduling algorithm is proposed for achieving efficient resource management based on these objectives. This proposed algorithm uses a modified genetic algorithm (GA) to find the optimal solution for choosing the most appropriate VMs for executing received tasks and their appropriate servers that will deploy these VMs. This proposed algorithm uses a matrix structure for representing the chromosome of GAS which combines the ids of tasks, VMs, and servers. Simulation results show that the proposed algorithm achieves better performance than ETVMC, TSACS, and ACO algorithms in terms of makespan, scheduling length, throughput, resource utilization, energy consumption, and imbalance degree.

Keywords: Cloud computing, Task scheduling, VMs placement, Server consolidation, Makespan, Energy consumption.

1. Introduction

Cloud computing system is depended on virtualization technology that is virtualization technology enables the resources of a single physical cloud resources to be divided several isolated execution environments running on virtual machines (VMs). The VMs are instances of the physical resources it is created and managed by a software layer which is called a hypervisor or a Virtual Machine Monitor (VMM). The VM run a task of user and when the task is completed it will shut down or allocated to another task [1, 2]. Using virtualization technology in cloud computing increases the throughput as well as scalability of the system [3].

Uninterrupted service is one of advantages of using cloud computing [4] that is needed to manage physical and virtual resources. Task scheduling and VMs placement are the two of the most important elements of resource management in cloud computing system. Task scheduling done based on objectives such as reduce time execution, reduce cost, load balancing on VMs, Quality of Service (QoS) or another objective. Also, VMs placement done based objective such as for maximizing resource utilization, load balancing, reduce wastage resources, or power management. Sometimes VMs placement done to meet the requirements of user which called initial VMs placement. Initial VMs placement is launching a VM on the nearest host. In this case may be there are VMs cannot deploy on hosts because required resources of VMs cannot provided by hosts. Another problem can happen when VMs assigned on host and the required resources of VMs less than the available resources in hosts (wastage resources this case called server sprawl).

However, because of dynamic heterogeneity of cloud computing environment, virtual machines are need to VMs replacement to improve the service availability which will be attractive for many users. VMs replacement Means movement the one or more VMs from physical host to another physical host this called VMs migration. VMs migration can happen when a physical server fails for any reason, VMs are deployed to other physical servers until the failure is corrected. Also, VM migration can done for maximizing resource utilization, load balancing, and power management [5-8].

This paper proposes a new algorithm for solving the scheduling problem in cloud computing. The proposed algorithm uses a matrix structure for representing the chromosome of GAS which combines the ids of tasks, VMs, and servers to find the optimal solution for choosing the most appropriate VMs for executing received tasks and their appropriate servers that will deploy these VMs. This prevent the occurrence of server sprawl phenomenon and reduce the needing for VMs migration process.

In this paper, the main contributions are:

(1) Solving multi-objective task scheduling problem in cloud computing.
(2) Using a modified genetic algorithm to find the optimal solution for choosing the most appropriate VMs for executing received tasks and their appropriate hosts that will deploy these VMs.
(3) Optimizing resource management for different criteria, including performance, power and cost. G-MOTSA dynamically allocate the CPU, memory and I/O resources to virtual machines according to requirements of cloud and the objective of user.
(4) Preventing the occurrence of server sprawl phenomenon and reduce the needing for VMs migration process.

The rest of this paper is organized as follows: the related work will be introduced in Section 2. Section 3. describes task scheduling and virtual machines (VMs) allocation problem. Section 4. problem formulation. Section 5. explains the proposed algorithm. Section 6. introduces the simulation and evaluation results of the proposed algorithm and Section 7. concludes the paper.

2. Related work

In recent years, a lot of algorithms have been proposed to schedule tasks on VMs [9-19]. These algorithms differ from each other in their methods and their objectives which are divided into three categories which are task scheduling, VMs placement, and complete mapping.

2.1 Task scheduling algorithms

Nasr et al. [9] interduce approach, called Traveling Salesman Approach for task Scheduling (TSACS). The main objective of this approach is minimizing makespan. This approach converts task scheduling problem into Traveling Salesman Problem (TSP) and then apply the nearest neighbour algorithm to solve the problem. This approach consists of three phases, first phase creates a set of clusters equal to number of available VMs by grouping the tasks into several clusters. Second phase calculate the cluster execution time to create Cluster Scheduling (CS) matrix which like the TSP matrix. Third phase apply the nearest neighbour algorithm to solve the problem. The disadvantage of this approach is that it focused only one criteria (makespan) at a time. Alworafi et al. [10] introduced Hybrid-SJF-LJF (HSLJF) algorithm which combines LJF and SJF algorithms. HSLJF sorts the tasks in ascending order, then selects one task based on SJF and another task based on LSF. Finally, the selected task is submitted to select VM that has minimum completion time. The disadvantage of this approach is that it focused only one criterion (minimum completion time) at a time. Fang et al. [11] proposed a scheduling task approach based on load balancing in cloud computing where the VM is described according to the needed resources for executing a task. Next it sorts the hosts in ascending order based on their processing power. Then VM selects a host that can provide the required resources and the load is lightest. Finally, if a task has been completed, the VM will be destroyed. Disadvantage of this work is creating VM for each task which is over head time. Also, if the resources that are needed to deploy VM in host is not available, then the VM waits for the second scheduling, which is not achieve the objective of a user. Sindhu and Mukherjee [12] introduced two scheduling methods to schedule tasks, Shortest Cloudlet Fastest Processing Element (SCFP) and Longest Cloudlet Fastest Processing Element (LCFP). In LCFP, the task that has large number of instructions is mapped to VM that has high computation power for minimizing the makespan. While in SCFP, the task that has short number of instructions is mapped to VM that has high commutating power for reducing FlowTime (completion time summation of a set of tasks). LCFP can minimize makespan but some tasks will be assigned to VM that does not have the minimum execution time for them. While SCFP can minimize FlowTime but maximizes makespan and decreases resource utilization. Also, LCFP may face starvation problem. Rekha and Dakshayini [13] proposed approach, called Efficient Task Allocation Genetic Algorithm (ETA-GA). this approach used genetic algorithm to reduce the task finishing time. Panwar et al. [14] proposed combination of TOPSIS algorithm and PSO algorithm called TOPSIS–PSO algorithm. The proposed algorithm performed in two stages. First stage is applied to obtain the relative closeness of tasks based on selected criteria (execution time, transmission time and cost). In second stage the particle swarm optimization (PSO) begins with computing relative closeness of the given three criteria for all tasks in all VMs. A weighted sum of execution time, transmission time and cost are used as an objective function by TOPSIS to solve the problem of multi objective task scheduling in cloud environment.

2.2 VMs placement algorithms

Sadiq et al. [16] interduce two approach using the cuckoo search optimization (CSO) algorithm to solve the VM placement problem. One for reducing number of active physical resources and another for reduce wastage resources as well as minimize the power consumption. In the second approach, Fuzzy logic is used to combine reducing of the power consumption and resource wastage into a single objective and then applied CSO. The disadvantage of this approach is that load balancing objective is not considered. Tawfeek et al. [17] applied Ant Colony Optimization (ACO) algorithm to solve the VM placement problem. The main objective of this work is reduced wastage resources of memory and CPU. The disadvantage of this work is that load balancing and minimizing power consumption objectives are not considered. Although there are more VMs placement algorithms like [18] but they are developed to achieve a single objective and not to achieve a multi objective.

2.3 Complete mapping algorithms

Complete mapping algorithms combine task scheduling and VMs placement issues in their solutions. Basu et al. [19] developed genetic algorithm with local search (GALS) for task scheduling and VM Placement. The main objective of GA-LS is minimizing the energy consumption and memory usage as well as performed load balancing on physical resource. The main idea in this algorithm is used the local search to select the two best chromosomes for every crossover operation and VM Migration done in mutation process to handles load balancing on physical resource. The disadvantage of this algorithm the time of execution, compilation time, makespan or response time not considered. Also, the chromosome represent distributions tasks on VMs are deployed on one physical resource, that is mean the resources request by all VMs needed to be available on this physical resource. Mishra et al. [6] introduced a complete mapping for task scheduling and VMs placement. The main objective of this work is reduced power consumption, makespan, and task rejection rate. This work does that sort tasks based on deadline time then classify them into four types based on their resource requirement. Also, there are four types of VM. All the VM types are in ascending of their resource capacity. Task assigned to fit VM of the same type. If the required resource of a task is not available in same type of VM, then search in the forward type for suitable VM. After that searches the host where the VM can deployed. The disadvantage of this work sorting and classifying tasks and VMs increase complexity. And load balancing objective is not considered. Also, if task assigned to VM and cannot deployed VM on host at this time, in this case the objective cannot performed.

Most of the previous work provided solutions for either scheduling tasks or scheduling VMs separately. Scheduling tasks algorithms are focused on the performance efficiency in execution, but once of important challenge faced task is that need to initial VMs placement to obtain the objective of these algorithms. In placement of initial VMs, VMs replacement (VMs migration) is needed, if the occurrence of server sprawl phenomenon or load imbalance can happen. In case of VMs placement done to perform maximizing resource utilization, load balancing, and power management, there are VMs not deployed on hosts because the requested resources by them are not available on hosts at this time. That is means objective of task scheduling algorithms may be not achieved. Scheduling VMs, in order to optimize resource management for different criteria, including performance, power and cost. They dynamically allocate the CPU, memory and I/O resources to virtual machines according to requirements of cloud, but they ignore the objective of user.

3. Multi-objective task scheduling problem

Currently, the big problem in CC is building task scheduling and virtual machines (VMs) allocation algorithms that consider all these objectives in the same time. This problem is called Multi-Objective task scheduling problem (MOTSP) which is NP completeness problem. In this section, the description of MOTSP will be introduced, then this problem is formulated.

3.1 Problem description

In cloud computing there are two main stages to perform resource management. The first stage is task scheduling which is determining the efficient VM for executing a task. Task scheduling done based on many objectives such as reducing time execution, maximizing resource utilization, load balancing, power management, or another objective. The second stage is VMs allocation (also called VMs placement) which is assigning VMs on a physical resource (i.e., host). VMs placement can be categorized into two types: 1) Initial VMs placement which is launching a VM on the nearest host to meet the requirements of a user task and 2) Objective-based VMs placement which is launching a VM on the most appropriate host that meets a certain objective such as maximizing resource utilization, load balancing, or power management.

One of the problems in VMs placement is that there are VMs cannot be deployed on the hosts because their required resources cannot be provided by these hosts (VMs rejection). Another problem that can be happened when VMs are assigned to hosts is that the available resources in these hosts are less than the required resources of VMs (wastage resources which is called server sprawl).

To solve these problem, VMs migration is needed. VMs migration is happened when a physical host fails in deploying its assigned VMs, for any reason. As a result, VMs are deployed to some other physical hosts until the failure is corrected. Also, VMs migration is done for maximizing resource utilization, load balancing, and power management. VMs migration is a complex problem which needs to predict or determine the source and destination hosts and the number of VMs that will be replaced. Also, whenever a VM is migrated, its CPU state, storage, main memory, and network connections are taking care of VMs continuous migration that causes additional overhead [3, 5-8, 18, 20].

4. Problem formulation

In cloud computing system there is a set of tasks, T= {T_1, T_2, ..., T_n} where *n* is the number of tasks. And there is a set of physical hosts PR= {R_1, R_2, ..., R_l} in distributed data centres where *l* is the number of hosts. For each physical resource, R_k there is a set of virtual machines, VMs= {VM_1, VM_2, ..., VM_m} will be deployed on R_k, *m* is the number of VMs. The task T_i in T will be assigned to VM_j which is the efficient VM for executing task, and the VM_j will be deployed on physical resource, R_k.

Many parameters need to be considered to achieve the objectives of a user or the objectives of a cloud vendor such as completion time, response time, makespan, resource utilization, wastage memory, wastage CPU, throughput, scheduling length (FlowTime), and power consumption. Assume that the I_i is denoted as the total number of instructions of task T_i, P_j is denoted as the total processing power of VM_j, and EX_{ij} is denoted as the execution time of task T_i on VM_j , which is the expected interval time elapsed to execute T_i on VM_j, then the execution time EX_{ij} is calculated as follows.

$$EX_{ij} = \frac{I_i}{P_j} \quad (1)$$

Where $1 \le i \le n$, $1 \le j \le m$, and assume that S_{ij} is denoted as the started execution time of T_i on VM_j and the completion time of task T_i which is the expected time for task finished execution on VM_j represented by CT_{ij}. If the tasks are independent and nonprimitive, then the completion time of a task CT_{ij} is calculated as follows.

$$CT_{ij} = S_{ij} + EX_{ij} \quad (2)$$

Assume that makespan is denoted as *MK*, which the completion time of last task for all user is and is calculated as follows.

$$MK = max\left(CT_{ij} \right) \quad (3)$$

And assume that scheduling length is denoted as *SL*, which is the total sum of completion time of all tasks (also called FlowTime) and is calculated as follows [12].

$$SL = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_{ij} \times CT_{ij} \quad (4)$$
$$, \quad \alpha_{ij} = \begin{cases} 1 & T_i \ assigned \ to \ VM_j \\ 0 & otherwise \end{cases}$$

Also, assume that the throughput is denoted as *TH*, which is the maximum rate of executing the tasks in unit time (i.e. is the total number of tasks, whose execution has been finished successfully per unit time) and is calculated as follows [10].

$$TH = \frac{n}{MK} \quad (5)$$

where *n* is number of tasks. Assume that the system performance is denoted by *SP* and is calculated as follows

$$SP = w_{mk} \times \frac{1}{MK} + w_{sl} \times \frac{1}{SL} + w_{th} \times TH \qquad (6)$$

where w_{mk}, w_{sl}, and w_{th} are the weights of makespan, scheduling length, and throughput respectively. These weights represent the importance of these terms in the system performance such that the sum of these weights must equal 1 as follows.

$$w_{mk} + w_{sl} + w_{th} = 1 \qquad (7)$$

Assume that the available processing power by physical resource R_k is denoted by β_k and the wastage processing power is represented by ρ_k and is calculated as follows.

$$\rho_k = \beta_k - \sum_{j=1}^{m} p_j \times \sigma_{jk}$$
$$, \sigma_{jk} = \begin{cases} 1 & VM_j \ deployed \ on \ to \ R_k \\ 0 & otherwise \end{cases} \qquad (8)$$

where $1 \leq k \leq l$ and assume that the available memory by physical resource R_k is denoted by μ_k, the request memory for deploying VM_j represented by M_j, and the wastage memory is represented by γ_k and is calculated as follows.

$$\gamma_k = \mu_k - \sum_{j=1}^{m} M_j \times \sigma_{jk},$$
$$, \sigma_{jk} = \begin{cases} 1 & VM_j \ deployed \ on \ to \ R_k \\ 0 & otherwise \end{cases} \qquad (9)$$

Assume that the total wastage resource is denoted by *W* and is calculated as follows.

$$W = w_p \times \sum_{k=1}^{l} \rho_k + w_m \times \sum_{k=1}^{l} \gamma_k \qquad (10)$$

Where w_p and w_m are the weights of processing power and memory respectively. These weights represent the importance of these terms in the system utilization and wastage resource such that the sum of these weights must equal 1 as follows.

$$w_p + w_m = 1 \qquad (11)$$

Assume that the estimated energy consumption by system in unite time *ut* is denoted by $\varepsilon_{k,ut}$, which is the amount of energy consumed by physical resource R_k to deploy all the accepted virtual machines in unite time and is calculated as follows.

$$\varepsilon_{k,ut} = \epsilon \times UP_k \qquad (12)$$

where ϵ is power consumption of utilization by physical resource in a time unit and UP_k is the utilization CPU of physical resource R_k and is calculated as follows.

$$UP_K = \frac{\sum_{j=1}^{m} p_j \times \sigma_{jk}}{\beta_k} \qquad (13)$$

The CPU performance that can significantly reduce energy consumption. The total estimated energy consumption by system is denoted by *EG* and is calculated as follows.

$$EG = \frac{\sum_{k=1}^{l} [\varepsilon_{k,ut} \times \sum_{j=1}^{m} \sum_{i=1}^{n} EX_{ij} \times \propto_{ij} \times \sigma_{jk}]}{ut} \qquad (14)$$
$$, \quad \alpha_{ij} = \begin{cases} 1 & T_i \ assigned \ to \ VM_j \\ 0 & otherwise \end{cases}$$
,and
$$\sigma_{jk} = \begin{cases} 1 & VM_j \ deployed \ on \ to \ R_k \\ 0 & otherwise \end{cases}$$

Server consolidation is done by reduction of the number of physical hosts. To increase energy efficiency and maximize resource utilization, packing the Active physical hosts with VMs as tightly as possible (multiple VMs are packed on fewest possible physical hosts and rest of the hosts are turned down to sleep mode (low power state)). Also, server consolidation technique is used for load balancing by offloading over utilized hosts (movement the VMs from physical hosts that is over load in which the required resources of VMs more than the available resources in host to physical host that is under load in which the required resources of VMs less than the available resources in host). Server Consolidation technique prevent server sprawl phenomenon happens. Inefficient resource allocation makes data centres underutilized (there are multiple underutilized physical hosts consume high resource and power consumption not justified according the workload being executed which called server sprawl phenomenon) [8, 16, 20].

The main objective of this paper is minimizing energy consumption by system (*EG* in Eq. (14)), wastage resource (*W* in Eq. (10)), and numbering of active physical resource and maximizing system performance (*SP* in Eq. (6)) and can express as

$$Maxmize\left(w_{sp} \times SP + w_e \times \frac{1}{EG} + w_r \times \frac{1}{W} + w_n \frac{1}{N}\right) \qquad (15)$$

Such that

$$\sum_{j=1}^{m} \propto_{ij} = 1 \qquad \forall\ i \in \{1,2,\dots,n\} \qquad (16)$$

$$\sum_{k=1}^{l} \sigma_{jk} = 1 \qquad \forall\ j \in \{1,2,\dots,m\} \qquad (17)$$

Where w_{sp}, w_e, w_r, and w_n are the weights of system performance, total energy consumption by system, wastage resource and numbering of active physical resource respectively. These weights represent the importance of these terms in the objective of scheduling in cloud computing such that the sum of these weights must equal 1 as follows.

$$w_{sp} + w_e + w_r + w_n = 1 \qquad (18)$$

And N is the number of active physical resource. The first condition in Eq. (16) meaning that the task will be execute on only one VM. The second condition in Eq. (17) meaning that the VM will be deploy on only one physical resource.

5. The proposed task scheduling and VM allocation algorithm

Efficient resource management in cloud computing environment for multiple tasks which are submitted by a user is one of the most challenging problems. The optimal assigning of tasks to virtual machines (VMs) and VMs to physical machines problem are necessary for advancing energy consumption and resource utilization. The main processes in resource management are determining the efficient VM for executing task and are determining the efficient host for binding VM. To make the best resource management in cloud environment, it is necessary to consider the required resources for tasks and the available resources of cloud hosts to prevent server sprawl happens and VMs rejection happens. In addition, the resource scheduling must satisfy the objectives of users and cloud vendors and improve the overall performance of the cloud computing environment.

To solve MOTSP in CC, a new task scheduling algorithm is proposed called Genetic-Based Multi-Objective Task Scheduling Algorithm, G-MOTSA is proposed. The goals of G-MOTSA are achieving multi-objective issues for resource management and satisfying trade-off conflicting requirements between users and cloud providers without performing any VMs migration. This section introduces the basic idea of G-MOTSA, then the proposed G-MOTSA in details.

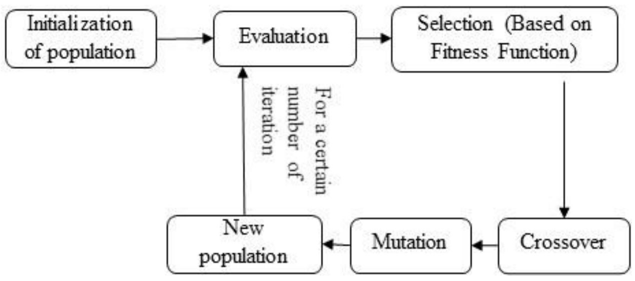

Initialization of population
Evaluation
Selection (Based on Fitness Function)
For a certain number of iteration
Crossover
Mutation
New population

Figure. 1 Flowchart of genetic algorithm

5.1 Basic idea

The basic idea of G-MOTSA is based on the data centre broker receives information about the tasks to be performed in terms required resources, the available servers and their characteristics (available number of core, speed of core , available memory and network bandwidth), as well as information about the VMs in terms of its characteristics (the required resources to deploy VM). After that, G-MOTSA uses a modified genetic algorithm (GA) to choose the appropriate VMs to implement those tasks and the appropriate servers to deploy the VMs, so that these chooses give the best performance of the system with the fewest number of servers, the least consuming energy, and the least wasting resources.

The multi-objective of G-MOTSA considers system performance, number of active hosts, energy consumption, and load balancing on host and resource utilization. G-MOTSA can minimize energy consumption, this is because that if there is no VM assigned to a host, place it to the sleep mode, where the servers switch off unneeded subsystems and put the RAM into a minimum power state. For example, IBM is developing a mode named deep sleep for its new Power processors that will allow them to consume almost no power when they are idle [21].

5.2 The proposed algorithm

For resource management, G-MOTSA uses genetic algorithm (GA) for distributing tasks on VMs and distributing VMs on available hosts of data centres by using three tuples: (1) a tuple for representing the IDs of tasks, (2) a tuple for representing the IDs of VMs, and (3) a tuple for representing the IDs of hosts. Therefore, G-MOTSA proposed a matrix structure for representing the

TId	0	1	2	3	4	5	6	7	8	9	10	11	12
VMId	1	2	4	6	1	5	6	3	5	4	6	6	6
HID	1	1	2	3	1	2	3	1	2	2	3	3	3

Figure. 2 Chromosome representation

chromosome of GA which combines the three different tuples. By using this matrix structure, the flow chart of GA processes is shown in Fig. 1.

In the next subsections, these processes will be described in detail.

5.2.1. Initialization process

Genetic algorithm begins with the initialization of population. The initial population is the set of all individuals that are used in the GA to represent the possible solution to the scheduling problem. Every individual is represented as a chromosome which is one such solution. Every chromosome consists of a set of genes. Gene is a one element position of a chromosome. The value of gene takes for a particular chromosome called Allele. The proposed solution represents the chromosome by *3 x n* matrix, where *n* is the number of tasks. And the first row represents id of a task, the second row represents id of a VM, and the third row represents id of a host as shown in Fig. 2.

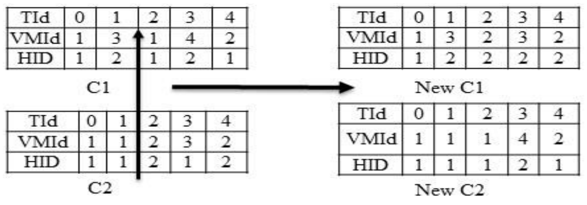
As shown in Fig. 2 tasks have ids 0, 4 will be execute on VM has id 1, while task has id 1 will be execute on VM has id 2. VMs have ids 1,2 will be deploy on host has id 1, so the VM has id 1 must be appropriate to implement the tasks that have ids 0,4 in terms of the required resources and give the highest system. Also, the choosing the least number of servers must be appropriate to deploy the VMs in terms of the required resources and give less wastage resource and less total energy consumption.

5.2.2. Evaluation process

Each chromosome is evaluated by a fitness function (objective). Fitness function measures the fitness value of chromosome. The fitness value determines the performance of an individual in the population. The Fitness function of the proposed solution is to minimize energy consumption by system, wastage resource, and numbering of active physical resource and maximize system performance Eq. (15).

5.2.3. Selection process

This process selects individuals from the population for mating. There are various selection mechanisms to select the best chromosomes such as selection based on rank, roulette wheel, tournament selection, and Boltzmann strategy. G-MOTSA used tournament selection mechanism.



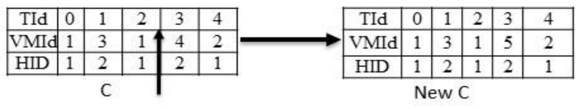
TId	0	1	2	3	4
VMId	1	3	1	4	2
HID	1	2	1	2	1

TId	0	1	2	3	4
VMId	1	1	2	3	2
HID	1	1	2	1	2

TId	0	1	2	3	4
VMId	1	3	2	3	2
HID	1	2	2	2	2

TId	0	1	2	3	4
VMId	1	1	1	4	2
HID	1	1	1	2	1

Figure. 3 Single point crossover operator



TId	0	1	2	3	4
VMId	1	3	1	4	2
HID	1	2	1	2	1

TId	0	1	2	3	4
VMId	1	3	1	5	2
HID	1	2	1	2	1

Figure. 4 Mutation operator

5.2.4. Crossover process

Crossover operation can be applied by selecting two individuals and using one of the two kinds of thecrossover operators that are single point crossover and order crossover operators. G-MOTSA used single point crossover. Crossover operation applied on the second and the third rows of the matrix (id of VM and id of host). Crossover operation generates new individuals see Fig. 3.

5.2.5. Mutation process

After crossover, mutation process takes place to prevent the population of individuals from changing into the same as one other. It occurs during evolution according to mutation probability. Mutation operation changes one or more gene values in the individual from its initial state. Mutation operation applied on the second and the third rows of the matrix (id of VM and id of host). This can produce the entirely new ids of VM and host. In Fig. 4 there are new ids of VM (VM has id 5 is new id) and host (host has id 3 is new id). With these new VM id and new host id, the genetic algorithm may be able to produce a better solution than was previously.

6. Simulation results and analysis

In this section, the simulation results will be presented to show the scheduling performance using G-MOTSA compared with performance ETVMC [6], TSACS [9], and ACO [17] algorithms.

Simulations are performed for two tests, in first test, the number of tasks is fixed and the number of VMs varies from 10 to 50. And in the second test, with the number of VMs is fixed and the number of tasks varies from 500 to 1000. Also, in two tests, the

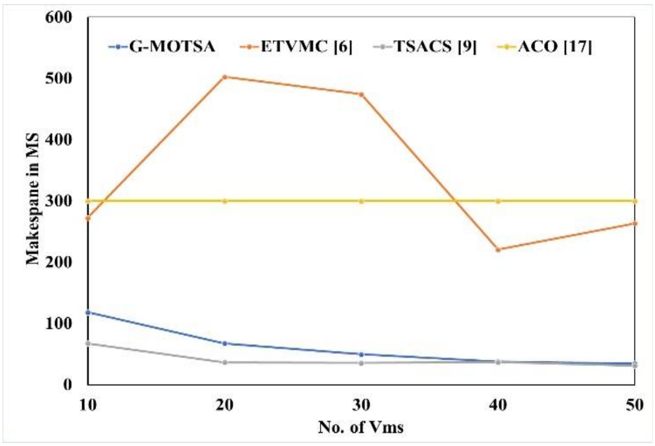


Figure. 5 Makespan vs. No. of VMs

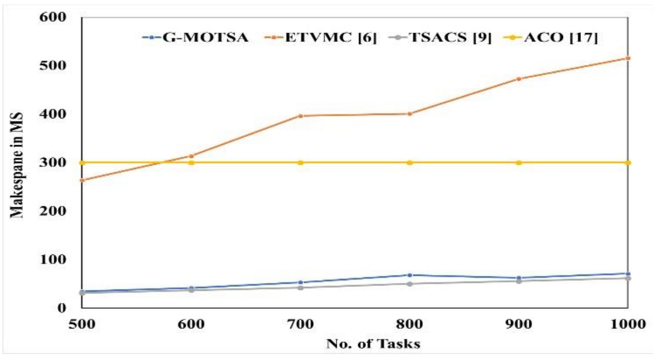


Figure. 6 Makespan vs. No. of tasks

number of hosts is fixed. In each test, the resource requirement, the length of the tasks, and the resources for the VMs are generated randomly. All experiments were done using CloudSim [22] which is a simulator for cloud computing environments supporting multiple VMs within a data centre node.

6.1 Makespan

Minimizing the time consumed for finishing execution time of all tasks (makespan) to satisfy the objectives of user and cloud vendors.

Figs. 5 and 6 show the makespan of G-MOTSA, ETVMC [6], TSACS [9], and ACO [17] algorithms for different values of VMs, when the number of tasks was fixed and for different values of tasks and the number of VMs was fixed, respectively.

As shown in Figs. 5 and 6, makespan for the G-MOTSA and TSACS [9] algorithm is much lower than ETVMC [6] and ACO [17]. Because ACO [17] solves only part of the problem (VMs allocation with the least wastage resources) and ETVMC [6] there are several tasks of a certain classification that have no parallel from the VMS, and therefore they are assigned to the VMs that achieves less execution time.

6.2 Scheduling length

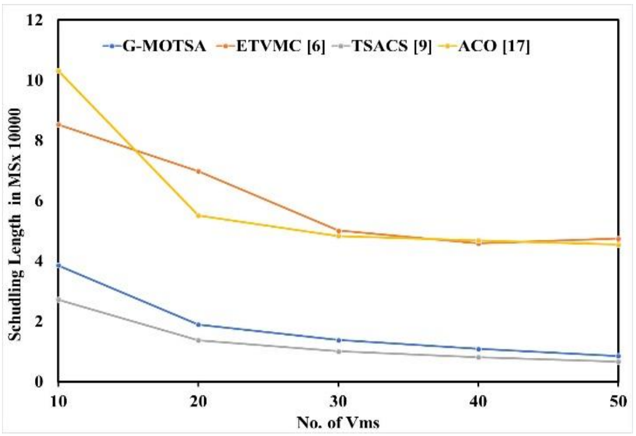


Figure. 7 Scheduling length vs. No. of VMs

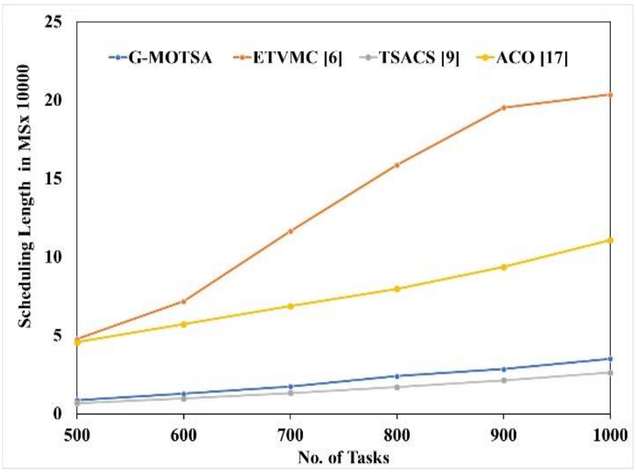


Figure. 8 Scheduling length vs. No. of Tasks

Figs. 7 and 8 show the scheduling length of G-MOTSA, ETVMC [6], TSACS [9], and ACO [17] algorithms for different values of VMs, when the number of tasks was fixed and for different values of tasks and the number of VMs is fixed, respectively.
As shown in Fig. 7 scheduling length for the proposed G-MOTSA, ETVMC [6], TSACS [9], and ACO [17] decreases as the number of VMs increases, this is because, existing of more VMs, will give opportunity to finish tasks earlier.

As shown in Fig. 8, scheduling length for the proposed G-MOTSA, ETVMC [6], TSACS [9], and ACO [17] increases as the number of tasks increases, this is because, adding more tasks with different execution times needs more executing times which increases the total scheduling length. From Figs. 7 and 8, the scheduling length of the G-MOTSA and TSACS [9] algorithm is much lower than ETVMC [6] and ACO [17].

6.3 Throughput

Figs. 9 and 10 show the throughput of G-MOTSA, ETVMC [6], TSACS [9], and ACO [17] algorithms for different values of VMs, when the

Received: June 28, 2021. Revised: August 7, 2021.

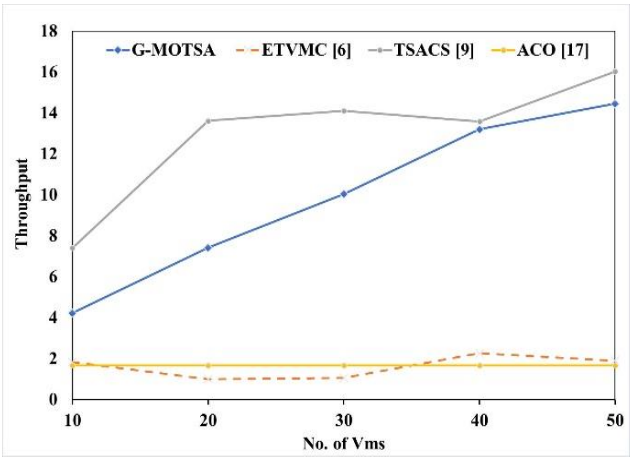


Figure .9 throughput vs. No. of VMs

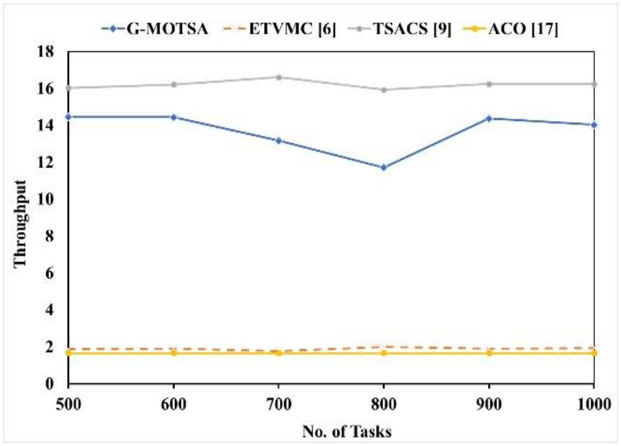


Figure. 10 Throughput vs. No. of tasks

number of tasks was fixed and for different values of tasks and the number of VMs is fixed, respectively.

As shown in Fig. 9, throughput for the proposed G-MOTSA, ETVMC [6], TSACS [9], and ACO [17] increases as the number of VMs increases, this is because, existing of more VMs, will give opportunity to maximize the throughput of the system.

As shown in Fig. 10, throughput for the proposedG-MOTSA and TSACS [9] is less changeable when the number of tasks increases, this is because G-MOTSA can adapt the number of assigned tasks to VMs. While throughput for ETVMC [6] is less affected by the increasing number of VMs, this is because ETVMC [6] cannot adapt the number of required VMs for executing the tasks. From Fig. 9 and 10, the throughput of G-MOTSA and TSACS was much higher than ETVMC [6] and ACO [17].

6.4 Resource utilization

Resource utilization (RU) is one of the objectives of cloud providers. The scheduling technique should improve the system performance and takes resource utilization into consideration. Resource utilization is

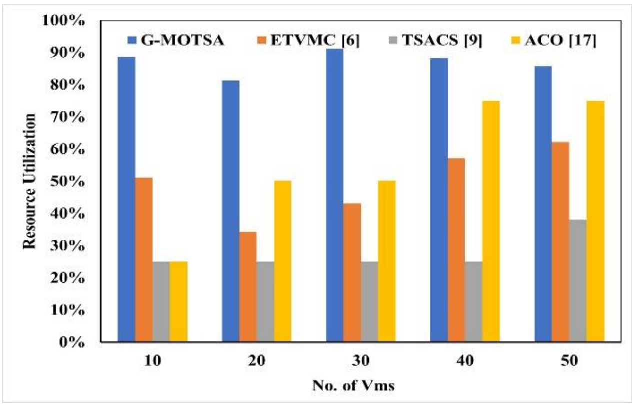


Figure 11. Resource utilization vs. No. of VMs

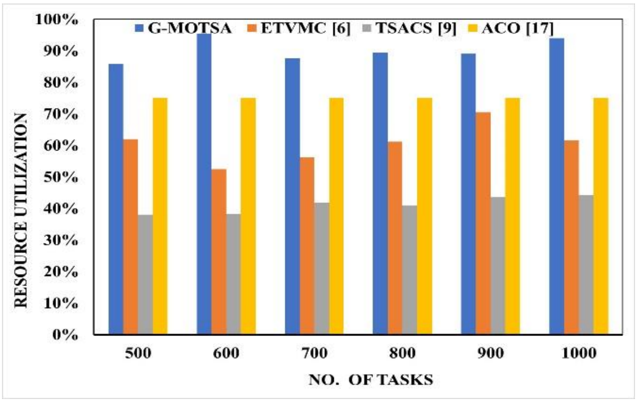


Figure 12. Resource utilization vs. No. of tasks

calculated as follows [10].

$$RU = \frac{\sum_{k=1}^{l} MK_k}{l \times \left(Max(MK_k)\right)} \tag{19}$$

where $1 \leq k \leq l$ and l is the number of hosts and MK_k is the makespan of all tasks that executed on host k. Figs. 11 and 12 show the resource utilization of G-MOTSA, ETVMC [6], TSACS [9], and ACO [17] algorithms for different values of VMs, when the number of tasks was fixed and for different values of tasks and the number of VMs is fixed, respectively.

As shown in Figs. 11 and 12, resource utilization for the proposed G-MOTSA, ETVMC [6], TSACS [9], and ACO [17] algorithms are changed as the number of VMs and tasks increases. Also, resource utilization of G-MOTSA was higher than other algorithms and G-MOTSA can achieve resource utilization up to 90% in two cases.

6.5 Estimated consumed energy

Figs. 13 and 14 show the estimated consumed energy of G-MOTSA, ETVMC [6], TSACS [9], and ACO [17] algorithms for different values of VMs, when the number of tasks was fixed and for different values of tasks and the number of VMs is fixed,

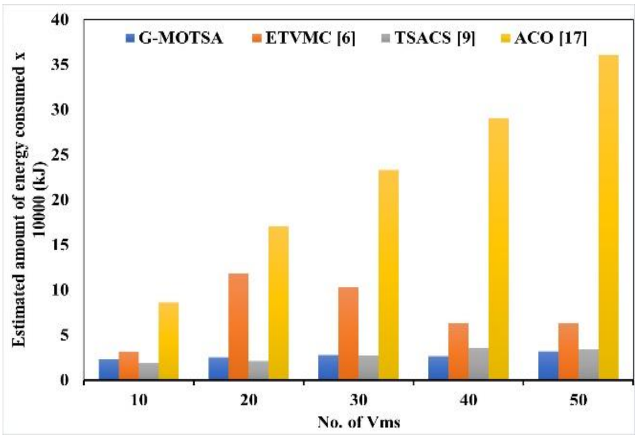


Figure. 13 Estimated consumed energy vs. No. of VMs

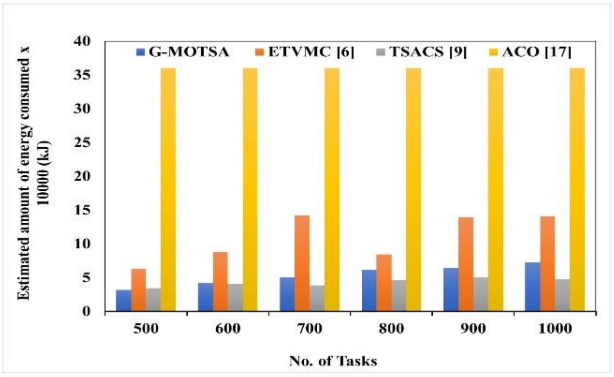


Figure. 14 Estimated consumed energy vs. No. of Tasks

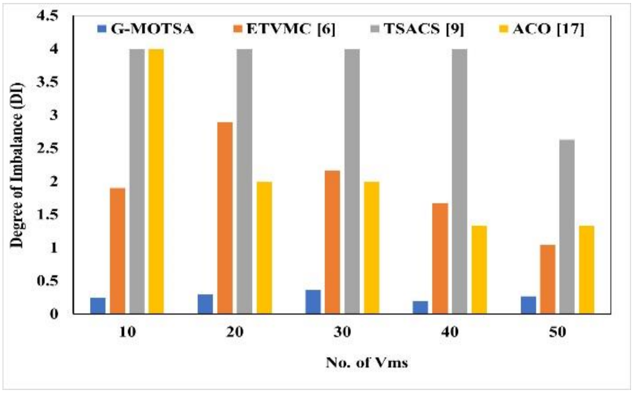


Figure. 15 Degree of imbalance vs. No. of VMs

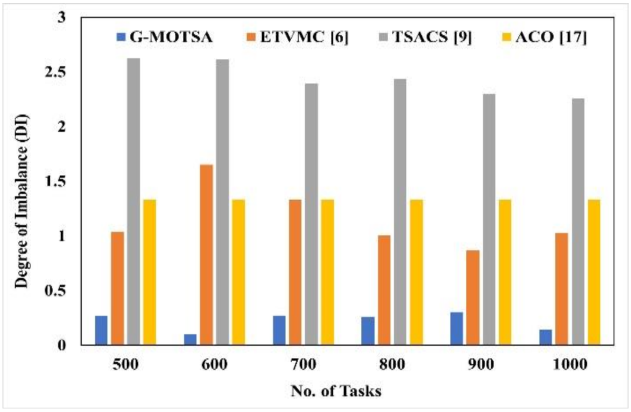


Figure. 16 Degree of imbalance vs. No. of Tasks

respectively. As shown in Figs. 13 and 14, the estimated consumed energy for the proposed G-MOTSA, ETVMC, TSACS [9], and ACO [17] algorithms are changed as the number of VMs and tasks increases. Also, estimated consumed energy of G-MOTSA and TSACS [9] was much lower than ETVMC [6] and ACO [17].

6.6 Degree of imbalance

The degree of imbalance (DI) measures the imbalance among hosts. DI is calculated as follows.

$$DI = \frac{T_{max} - T_{min}}{T_{avg}} \tag{20}$$

Where, T_{max}, T_{min} and T_{avg} are the maximum, minimum and average total execution time (in seconds) of all hosts, respectively.

Figs. 15 and 16 show the degree of imbalance obtained by applying the G-MOTSA, ETVMC [6], TSACS [9], and ACO [17] algorithms for different values of VMs, when the number of tasks was fixed and for different values of tasks and the number of VMs is fixed, respectively. As shown in Figs. 15 and 16, degree of imbalance for the G-MOTSA and other algorithms is changed as the number of VMs and tasks increases. Also, degree of imbalance of G-MOTSA was much lower than other algorithms. This means that the G-MOTSA can achieve good system load balance.

Based on these results, the results obtained by applying the proposed G-MOTSA are the best in terms resource utilization, energy consumption, and imbalance degree among all other algorithms. This prevent the occurrence of server sprawl phenomenon and reduce the needing for VMs migration process. While in terms of makespan, Scheduling length, and throughput, TSACS [9] and G-MOTSA algorithms achieve the best results with small difference between two algorithms, where this difference decreases with the increasing in the number of VMs or tasks. In addition, TSACS [9] algorithm solves only part of the problem (scheduling tasks with the least completion time). Also, TSACS [9] algorithm select All VMs that is need the resource request by VMs should be available by hosts, in most time cannot achieve. Finally, the complexity degree the TSACS [9] algorithm is greater than the others.

7. Conclusion and future work

In this paper, a new task scheduling algorithm is proposed called G-MOTSA for solving multi-objective task scheduling problem in cloud

Received: June 28, 2021. Revised: August 7, 2021.

computing. G-MOTSA uses a modified genetic algorithm to find the optimal solution for choosing the most appropriate VMs for executing received tasks and their appropriate hosts that will deploy these VMs. G-MOTSA can optimize resource management for different criteria, including performance, power and cost. G-MOTSA dynamically allocate the CPU, memory and I/O resources to virtual machines according to requirements of cloud and the objective of user, while most of existing algorithms do not take the objective of user into account. G-MOTSA can prevent the occurrence of server sprawl phenomenon and reduce the needing for VMs migration process. The simulation results shown that G-MOTSA can achieve better performance than some of existing methods in terms of makespan, scheduling length, throughput, resource utilization, and energy consumption. In the future work, the task types (dependent) and priorities, in the optimization model will be considered. In addition, the proposed model will be implemented in a real cloud environment.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

Conceptualization, methodology, software, validation, validation, formal analysis, investigation, resources, data curation, and writing—original draft preparation, Ahmed. A. A. Gad-Elrab and Farouk A. Emara; writing—review and editing, Ahmed. A. A. Gad-Elrab; supervision, Ahmed. A. A. Gad-Elrab, K. R. Raslan, and Ahmed Sobhi.

Acknowledgments

This research was supported by Faculty of Science, Al-Azhar University, Cairo, Egypt. In addition, it was partially supported by King Abdul-Aziz University, Jeddah, Saudi Arabia. I thank both for providing us the ability to conduct and finish this research.

Reference

[1] B. Furht and A. Escalante, *Handbook of cloud computing*, Vol. 3, Springer, 2010.

[2] D. Sullivan, "The definitive guide to cloud computing", *Real Time Nexus*, No. 1, pp. 4–11, 2010.

[3] Z. Usmani and S. Singh, "A survey of virtual machine placement techniques in a cloud data center", In: *Proc. of 1st International Conf. on Information Security and Privacy*, pp. 491-498, 2016.

[4] Y. Jadeja and K. Modi, "Cloud computing - concepts, architecture, and challenges", In: *Proc. of International Conf. on Computing, Electronics and Electrical Technologies (ICCEET)*, pp. 877–880, 2012.

[5] A. Choudhary, S. Rana, and K. Matahai, "A critical analysis of energy efficient virtual machine placement techniques and its optimization in a cloud computing environment", In: *Proc. of 1st International Conf. on Information Security and Privacy*, pp. 132-138, 2016.

[6] S. Mishra, D. Puthal, B. Sahoo, P. Jayaraman, S. Jun, A. Zomaya, and R. Ranjan, "Energy-efficient VM-placement in cloud data center", *Sustainable Computing: Informatics and Systems*, Vol. 20, pp. 48 – 55, 2018.

[7] A. Bhandari and K. Kaur, "An Enhanced Post-migration Algorithm for Dynamic Load Balancing in Cloud Computing Environment", In: *Proc. of International Conf. on Ethical Hacking*, Singapore, pp. 59-73, 2019.

[8] M. Shirvani, A. 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.

[9] A. Nasr, N. E. Bahnasawy, G. Attiya, and A. E. Sayed, "Using the tsp solution strategy for cloudlet scheduling in cloud computing", *Journal of Network and Systems Management*, Vol. 27, No. 2, pp. 366–387, 2019.

[10] M. Alworafi, A. Dhari, S. E. Booz, A. Nasr, A. Arpitha, and S. Mallappa, "An enhanced task scheduling in cloud computing based on hybrid approach", In: *Proc. of International Conf. on Data Analytics and Learning*, pp. 11–25, 2019.

[11] Y. Fang, F. Wang, and J. Ge, "A task scheduling algorithm based on load balancing in cloud computing", In: *Proc. of International Conf. on Web Information Systems and Mining*, Berlin, Heidelberg, pp. 271–277, 2010.

[12] S. Sindhu and S. Mukherjee, "Efficient task scheduling algorithms for cloud computing environment", In: *Proc. of International Conf. on High Performance Architecture and Grid Computing*, Berlin, Heidelberg, pp. 79-83, 2011.

[13] P. Rekha and M. Dakshayini, "Efficient task allocation approach using genetic algorithm for cloud environment", *Cluster Computing*, Vol. 22, No. 4, pp. 1241–1251, 2019.

[14] N. Panwar, S. Negi, M. Rauthan, and K. Vaisla, "Topsis–pso inspired non-preemptive tasks scheduling algorithm in cloud environment", *Cluster Computing*, Vol. 22, No. 4, pp. 1379–1396, 2019.

[15] K. Naik, G. Gandhi, and S. Patil, "Multiobjective virtual machine selection for task scheduling in cloud computing", In: *Proc. of International Conf. on Computational Intelligence: Theories, Applications and Future Directions*, Singapore, pp. 319-331, 2019.

[16] S. Sait, A. Bala, and A. E. Maleh, "Cuckoo search based resource optimization of datacenters", *Applied Intelligence*, Vol. 44, No. 3, pp. 489–506, 2016.

[17] M. Tawfeek, A. E. Sisi, A. Keshk, and F. Torkey, "Virtual machine placement based on ant colony optimization for minimizing resource wastage", In: *Proc. of International Conf. on Advanced Machine Learning Technologies and Applications,* Cham, pp. 153-164, 2014.

[18] D. Patel, R. Gupta, and R. Pateriya, "Energy-Aware Prediction-Based Load Balancing Approach with VM Migration for the Cloud Environment", In: *Proc. of International Conf. on Data, Engineering and applications*, Singapore, pp. 59-74, 2019.

[19] S. Basu, G. Kannayaram, S. Ramasubbareddy, and C. Venkatasubbaiah. "Improved genetic algorithm for monitoring of virtual machines in cloud environment", In: *Proc. of International Conf. on Intelligent Computing and Applications*, Singapore, pp. 319-326, 2019.

[20] N. Vahed, M. G. Arani, and A. Souri, "Multiobjective virtual machine placement mechanisms using nature inspired metaheuristic algorithms in cloud environments: A comprehensive review", *International Journal of Communication Systems (IJCS),* Vol. 32, No. 14, 2019.

[21] D. Meisner, B. Gold, and T. Wenisch, "The powernap server architecture", *ACM Transactions on Computer Systems,* Vol. 29, No. 1, 2011.

[22] R. Calheiros, R. Ranjan, A. Beloglazov, C. D. Rose, and R. Buyya "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", *Software: Practice and Experience*, Vol. 41, No. 1, pp. 23–50, 2011.