ISSN 2347 - 3983
Volume 8. No. 6, June 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter92862020.pdf
https://doi.org/10.30534/ijeter/2020/92862020

An Enhanced Cloud Computing Architecture focussed at Optimising VM Migration through an efficient Placement Algorithm

K. Aruna Kumari[1, 2], Dr.JKR Sastry[3], Dr. K Rajasekhara Rao[4]
[1]Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur District, AP, arunasatti@gnail.com
[2]SRKR Engineering College, China amiram, Bhimavaram, WG District, AP, arunakumarisatti@srkrec.ac.in
[3]Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur District, AP, drsastry@kluniversity.in
[4]Usha Rama College of Engineering, Telaprolu, Krishna District, AP, krr_it@yahoo.com

ABSTRACT

Generally, the architecture of a cloud computing system contains several layers, which includes hardware layer, infrastructure layer, Platform layer, and the Services layer. Each layer performs different kinds of operations and provides different types of services.

VM Migration is undertaken by a scheduler installed within computing nodes with the main aim of balancing the Load on Physical servers. The Migration is undertaken based on the internal design of a cloud computing system. The response time agreed with the user and contained within an SLA gets compromised due to the heavy processing required for deciding on the migration and actually effecting the Migration.

Unlike a traditional Virtual Machine (VM), a container is an emerging lightweight virtualization technology that operates at the operating system level to encapsulate a task and its library dependencies for execution.

In this paper, a modified layered architecture is presented which considers the addition of another layer (Optimised Migration Layer) that is responsible for optimising the process of migration focusing on reduction of delay time caused due effecting the migration. The additional layer uses the concept of containerisation for optimising and reduction in delay time caused due to effecting the Migration. An efficient placement algorithm has been implemented in the OML layer. OML includes certain components, such as validation, business, transformation, and the deployment phase. The component that is responsible for effecting the Migration is quarantined by making changes to the configuration files. OML is designed using four phases that include Validation, Business phase, Transformation phase.

The scheduling algorithm calls the software situated in OML layer which checks the validity of the request and the actual migration is effected in the Business phase in which an efficient Migration/Placement algorithm is implemented. Here, the placement algorithm, named Squirrel Whale Optimization Algorithm (S-WOA) is implemented. The migration is done based on five conditions, which include VM to Physical Machine (PM), container to VM under same PM, and the container to VM under different PM, tasks in VM to container, and container tasks to VM. A fitness function is designed and implemented to find, best migration method. The fitness function is designed using several parameters that include bandwidth, resource utilization and load.

In the transformation phase the transformation from VM to container and vice versa is undertaken. The performance of the migration strategy in cloud based on S-WOA is evaluated in terms of number of instantiated VMs, CPU utilization, memory utilization, number of activated PMs, and time. The proposed S-WOA method achieves the maximal CPU utilization of 0.483, maximal memory of 0.523, and minimal time of 4.99sec.

Key words*:* Cloud computing, Migration, Squirrel search algorithm, Optimized Migration Layer, Whale optimization algorithm.

1. INTRODUCTION

Cloud computing has been emerged as the mainstream service-oriented architecture. However, the large-scale application of the cloud computing involves huge number of workloads and increasing number of tasks [1]. Due to the existence of varying loads while some computing nodes are overloaded and some underutilised leading unbalanced Load distribution [2]. Hence, it is very imperative for spreading loads across the computing nodes for taking complete advantage of the cloud computing system with improved user satisfaction [3].

As the new prevalent commercial paradigm, the cloud computing has been paid great attention in both industrial and academic communities. Thanks to advanced improvement of cloud computing, several enterprises and the individuals are permitted for outsourcing significant data to the cloud in spite of maintaining and building local data centres. In addition, the cloud users also enjoy several kinds of computing services provided by the public cloud [4].

Cloud computing is defined by the NIST as the model to enable on-demand network access, which is convenient to the shared pool of the computing resources, like servers, networks, applications, storage, and the services that is rapidly released and provisioned with reduced cloud provider or interaction management effort. Furthermore, the cloud computing is considered as then ovel computing paradigm as it allows use of computing infrastructure at more than one levels of abstraction over another computer network or the Internet. Due to the implications for higher flexibility and the availability at the lower cost, cloud computing is the subject that is receiving the good deal of attention [5].

The services provided through the cloud computing system share the network and the infrastructural facilities like storage, Physical servers etc. The key aspect behind the efficiency and performance is actual placement of service functions, storage allocation, data flow paths, and network routing flows [**6**].

A Cloud Service Distribution Problem (CSDP) [7] is designed with the main aim to identify the placement of virtual functions and deciding on the routing of the network flows, which satisfies resource capacities, meets QoS requirements, and mitigates overall infrastructure cost. The method did not help when cloud computing services are provided as backend services to an IoT based application which considers a separate device layer and many other layers that gets connected to cloud computing layers [**8**] [9].

Over the years, various techniques for improving performance of cloud computing systems have been developed that include priority-enabled work consolidation [10], process synchronization using simulation instances [11] resource sharing between parallel jobs based on gang scheduling approach [12], and the usage of shared event queue between cloud multi-core systems. However, the approaches lead to serious bottlenecks especially the need to handle huge number of threads for simulating a running cloud.

A Recent technique presented in the literature is federation-enabled which is meant for improving performance of large-scale data centre. In this case, Genetic algorithm are used for optimising the placement of VMs. In addition, ordering algorithms are employed to generate VM placement, and prediction techniques are introduced for addressing time varied predicted demands. Moreover, the dynamic VM placement method is introduced to deal with VM placemen. The VM place methods are divided into Quality of Service (QoS), and power-based techniques.

The existing VM placement approaches are separated as dynamic and static. Additionally, the forecasting algorithms are to be designed considering several search spaces for prediction. More amount of execution time is required when Migration decisions are taking using genetic algorithm.

Most of the algorithms have been experimented in simulation mode and the performance of those algorithms in real cloud computing environment is never tested. Integrating new algorithms with the existing cloud computing system has been seen to a big bottleneck.

Many migration strategies exist having the issue of containerisation wide open. Thus there is a requirement of selecting a best migration strategy and then effecting the same based on the placement strategy, keeping in vies of fulfilling SLA condition through choice of a proper fitment function.

2. LITERATURE SURVEY

The eight classical strategies based on migration in cloud computing along with its limitations have been deliberated. Mohamed K. Hussein *et al.*[13] designed placement architecture for container as a service (CaaS) in cloud. This architecture employed the scheduling heuristics, such as Max Fit (MF) and Best Fit (BF). Here, the BF, and MF was computed using the fitness function, which simultaneously computed the remaining resource waste of both VMs and PMs. Moreover, the meta-heuristic placement approach was introduced that employed Ant Colony Optimization based on Best Fit (ACO-BF). The method was Highly effective in terms of VM and PM utilization and in minimizing the number of active PMs and instantiated VMs, but failed to consider additional computing resources like processing cores, memory, storage, and data transfers.

Rong Zhang *et al.*[14] developed Container-VM-PM architecture for the Docker container placement. The Docker container placement issue was focussed under CVP architecture by simultaneously considering three involved entities. In addition, the fitness function was considered for selecting PM and VM. The method was more conductive in using the resource effectively, however container consolidation and load balance under the CVP architecture was not considered for better system performance.

Li Chunlin*et al.*[15] presented dynamic multi-objective optimized replica placement and the migration techniques for the SaaS applications in the edge cloud. Here, the replica placement issue was solved based on fast-non-dominated sorting genetic algorithm. In addition, the replica migration model for the access hotspots was introduced for obtaining pairing migration relationship from target to source node. Thus, the method reduced the migration time and response time and improved the network resource utilization. However, the method did not consider the additional I/O overhead caused by the migration process.

SaadZaheer*et al.*[16] developed an approach for facilitating the implementation and for evaluating the process of placement algorithms within the three-tier cloud data centre. Additionally, classical clustering technique was introduced for testing various process placement strategies. Then, the locality-aware criterion was established to restructure the underlying network automatically for placement process efficiently. The method failed to consider dynamic load

balancing, process consolidation, and adaptive learning techniques to enhance the performance.

Ruiting Zhou *et al.*[17] presented online algorithm for maximizing the aggregate value of all the served clusters. Here, one-shot approach was introduced for determining the placement scheme of given container cluster (CC). In addition, the primal-dual online placement algorithm was established that used one-shot approach as the building block in order to make the decisions upon an arrival of every CC request. The method achieved better computational and economical efficiencies, but still the on-spot decisions was not produced without relying on knowledge of future request arrivals.

Bo Liu *et al.*[18] developed an approach, named multi-objective container scheduling based on multi-objective optimization. This framework considers memory usage of each node, CPU usage of each node, time consumption transmitting images on network, the association among containers and the nodes, the clustering of containers that affect the application performance in the containers. Then, the appropriate node was selected for deploying the containers required to be allocated in scheduling process. Consequently, the metric method was defined for each key factor, and then the scoring function was established for each factor. Finally, combined the output of each factor to the composite function for better system performance. However, the fault tolerance of the containers was not considered for the training process.

Kamran, and Babar Nazir [19] developed an approach for the VM's placement and migration by considering several users quality of the service requirement for decreasing SLA violations and energy consumption because of the underutilization of the data centres. Additionally, the heuristics-enabled energy aware resource allocation approach was introduced for allocating user's tasks in cloudlets into cloud resources, which yields minimal energy. The aspect of fault-tolerant VM migration was not considered to achieve better system performance.

Anurag Satpathy*et al.*[20] presented two-tier virtual machine placement approach in the cloud data centres with the live migration.. Initially, the queueing structure was introduced for managing and scheduling huge set of VMs. After that, the multi-objective VM placement approach, named crow search-enabled VM placement (CSAVMP) was established for reducing power consumption and resources wastage at data centres. The method undergoes data centre management, which is complex, and time-consuming task.

A number of recommendation made related to VM Migrations but most of the recommendations did not consider different strategies of fitness function that is built using different SLA conditions [21][22][23][24][25][26][27] [28][29][30][31][32][33] [34][35[36] [37][38][39]

3. CHALLENGES CONFRONTING THE VM MIGRATION

The challenges confronted by the conventional strategies are deliberated below:

- The Container as a Service (CaaS) method becomes very prominent kind of the cloud service model. However, the placing of container on the VM is a classical scheduling issue. Existing research separately focused on either VM placement on PMs or the container, or only the tasks without containerization, placement on VM. Although, this method leads to over-utilized or underutilized PMs, and over-utilized or underutilized VMs [13].
- In [14], the container placement method is developed for the VM placement. Here, the Docker is the mature containerization scheme utilized for performing the operating system level virtualization. One open problem in cloud is how to select the VM properly for initializing the container that is same to conventional issue of VM placement towards PMs. Though, the major reason for scattered the distribution of containers in data centre that results in worse physical resource utilization.
- The placement of replicas must select the appropriate location for maintaining system load balancing and for improving the performance of the system. Because of the huge number of the heterogeneous nodes in various locations of edge cloud system, the inefficient replica placement algorithm affects system load balancing. Hence, how to place replicas in edge cloud environment is the major issue**[15]**.
- In the traditional cloud services, the major challenges in the container cluster provisioning lies in optimal container placement when considering inter-container traffic in container cluster. The key challenge further escalates, when the container cluster are provisioned in the online manner**[16].**
- VM migration is the process of migrating VMs from one physical server to the other. It provides various benefits to data center in several scenarios, which includes fault tolerance, manageability load balancing, improved performance, and the power management, but the VM's migration leads to service-level agreement (SLA) violations and performance degradation that cannot be ignored, particularly if the critical business goals are to be met.

4. THE GAP

The issue of containerisation has not been much addressed especially considering the issue of VM Migration. The time taken for migrating a VM is based on the size of the VM. The lesser the size of a VM, the faster the Migration. One can have several options to migrate a VM from one Physical

machine to the other. The existing algorithms have considered only one option of moving a VM to a Physical machine and ignored other alternatives that can be considered for effecting VM migration. The choice of a fitness function that includes all performance issues also is never considered

5. PROBLEM DEFINITION

The problem thus is to determine best migration strategy that takes into account a fitness function into which all performance and resource utilisation requirements are included

6.INVESTIGATIONS AND FINDINGS

6.1 The enhanced Cloud Computing Architectural Model

The primary intention of this research is to develop an approach for migration in cloud through addition of a layer that caters for effective VM migration. The scheduling and Migration algorithms are implemented in this layer.

In the cloud computing architecture, there are four layers, such as hardware layer, infrastructure layer, platform layer, and the application layer. In addition, the OML is newly introduced in the cloud architecture, which is meant to perform migration. In the OML, the placement algorithm, considers various placements options and finds the best and then initiates the migration that is chosen to be the best. 5 Different strategic options are considered that include Migrating a VM to a Physical Machine, Converting a VM to a Container which is then migrated to a Physical machine and then the VM is constructed back, Converting a Container to a VM and then Migrating to a different machine, Migrating a Container to another Physical Machine and then converting it back to a Physical Machine.

OML follows four phases that includes validation, business, transformation, and the deployment for effecting VM migration under clouds.

The scheduler of a cloud computing system initiates the process for effecting VM migration by interfacing with Validation program situated in the OML layer. Once the request is found to be genuine, then the actual Migration is undertaken through another program called as business program. The business program is designed to determine best migration option. The conversion of VM to container or container to VM is undertaken by a separate program which is affected based on the strategic decision identified by the business program. The interaction between different programs in OML layer is shown in the Figure 1.

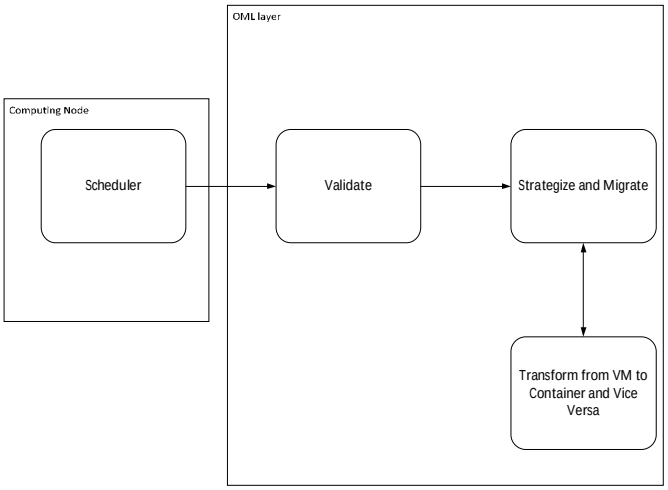

Figure 1: TheOML Layer Architecture

6.2 The Migration Approach

The migration approach based on S-WOA optimization algorithm in cloud computing platform is shown in Figure 2that depicts the system model of migration in cloud. In the last few decades, the cloud computing has been paid great attention in the field of computer science. The cloud computing model provides flexible, and the simplest way to manage and retrieve files and data. However, the cloud model composed of several PMs for tackling requests from the users, and the PM collects the VMs to process the tasks dynamically. The VM available in cloud produced in the dynamic way to reduce the bottleneck and the visualization problem. In addition, the container is the novel lightweight virtualization technology that does not necessitate Virtual Machine Monitor (VMM). In addition, the container offers the isolated virtual environment at operating system level. However, various containers executing on the particular operating system share same operating system kernel, and then the container does not need launching an operating system. However, the migration is carried out on VM to container, or Container to VM or VM cluster to Container cluster.

6.3 The Five-layer architecture

This section presents the proposed S-WOA algorithm using Optimized Migration Layer (OML) architecture in cloud computing for migration. In general, the cloud computing architecture can be divided into several layers, like hardware layer, infrastructure layer, platform layer, and application layer. In addition, the OML is newly devised, and is placed above the application layer using certain components, such as validation phase, business Phase, transformation phase, and the deployment phase. In the validation phase, the request of migration is first checked, and then the status of VM/Container is checked. After that VM/Container metrics is validated for the migration in cloud. In business Phase, the actual information is checked. Here, the Placement Engine algorithm is utilized using newly devised optimization algorithm, namely S-WOA for selecting the suitable server

(Physical Machine) for migration. The migration is carried out on VM to container, or Container to Virtual Machine or Virtual Machine cluster to Container cluster.

The proposed S-WOA is designed by integrating SSA[40] and WOA[41]. In addition, the fitness function is newly devised based on bandwidth, resource utilization and load. In the transformation phase, the transformation of the VM information to the container information or vice versa in Migration Object Notation (MON) is done. Finally, in deployment Phase, the placement algorithm adapted in the business phase is utilized in the destination location for the migration, and from the transformation phase the converted code is available, which is deployed according to the required strategy. Once the deployment is completed the source resource is deleted. Figure 3 illustrates the architecture of cloud computing model.

6.4 Cloud computing architecture using proposed Squirrel-based Whale Optimization algorithm for migration

The flowchart of the developed S-WOA for migration in cloud computing architecture is depicted in figure 4.The OML consists of following components, like validation phase, business phase, transformation phase, and the deployment phase, and is elaborately explained below.

6.5 The Phases in OML Layer

6.5.1 Validation phase

Let us assume the cloud model with different VMs, PMs, and the containers. The cloud model containing J number of PMs, represented as $J = \{J_1, J_2, ..., J_i, ..., J_n\}; 1 \le i \le n$,and several VMs are available under each PM. In addition, the V number of VMs is indicated as, $V = \{V_1, V_2, \ldots, V_j, \ldots V_m\}; 1 \le j \le m$, where, the total number of VMs is denoted as V_m .Additionally, the C number of available container is indicated as, $C = \{C_1, C_2, \ldots, C_k, \ldots C_p\}; 1 \le k \le p$, where, the total amount of container is denoted as C_k . Thus, in the validation phase, whenever a necessity/ request comes for migration it first checks for the following to be true: a) VM/Container is existing, up and running, b) VM/Container metrics are correct. However, each container consists of several parameters, such as CPU utilization, memory, Million Instructions per Second (MIPS), number of processing elements, and the frequency, and the equation is expressed as,

$$C_k = \{D_k, H_k, B_k, A_k, I_k\} \tag{1}$$

where, the term D_k denotes the CPU utilization in k^{th} container, and the term H_k refer to the number of memory used by k^{th} container. The total number of MIPS utilized by k^{th} container is denoted as B_k , the symbol A_k signifies the total processing elements utilized by k^{th} container, and the term I_k indicates the frequency of k^{th} container. Five conditions should be followed for migration, given below.

- From VM to PM
- Container to VM under the same PM
- Container to VM under different PMs.
- Migrate tasks in VM to container
- Migrate container tasks to VM

Task Quality Rate computation

The Task Quality Rate (TQR) is computed by considering the number of tasks with original deadline of tasks and time taken to complete the task. Thus, the TQR is computed using the below equation.

$$TQR = \frac{1}{|t|}\sum_{r=1}^{|t|}\left(\frac{\mathrm{P}_{ori} - \mathrm{P}_{comp}}{G}\right) \tag{2}$$

where, the number of tasks is denoted as $|t|$, the original deadline of tasks is indicated as P_{ori} , and the completed task time is represented as P_{comp} .Two conditions should be followed when the TQR is greater than or lesser than or equal to threshold.

- If $TQR > T$, compute the load in the cloud that followed three conditions.
- If $TQR \le T$, and load of VM is greater than threshold, perform condition-4
- If $TQR \le T$, and the load of VM islesser thanor equal to threshold, perform condition-5.

Computation of load

The load value is calculated based on the employed resources by the container to perform the task achieved from the user. Memory, CPU, number of processing elements, MIPS, and frequency are the parameters considered for load computation in the cloud platform. Therefore, the load of container is computed using the expression given below,

$$L_k = \sum_{q=1}^{r} \frac{\left(G^D + G^H + G^B + G^A + G^I\right) * y}{e * \left(\max\left(G^D\right) + \max\left(G^H\right) + \max\left(G^B\right) + \max\left(G^A\right) + \max\left(G^I\right)\right)} * \frac{1}{G_1} \tag{3}$$

where, the term G^D denotes the total amount of CPU utilized in the k^{th} container, G^H represents the total number of memory utilized in k^{th} container, G^B indicates the number of MIPS in k^{th} container, G^A is the number of

processing elements in k^{th} container, the symbol G^I is the number of frequency in k^{th} container, the term r refer to number of task, and G_1 is the normalizing factor, and the maximum value is denoted as $\max(.)$. If the q^{th} task is run by the k^{th} container, then parameter y is denoted as 1, or else y becomes 0.

$$y = \begin{cases} 1 \ ; If \ q^{th} \ user \ request \ is \ run \ by \ k^{th} \ container \\ 0 \ ; Otherwise \end{cases}$$

(4)

The load of VM is computed by considering the load of container and the normalizing factor. Hence, the load of VM is determined as,

$$L_j = \sum_{k=1}^{p} L_k * \frac{1}{G_2}$$

(5)

where, the term L_k represents the load of k^{th} container, and the normalizing factor is indicated as G_2. Then, the load of PM is calculated by considering the load of VM with the normalizing factor, and the expression for finding the load of PM is given by,

$$L_i = \sum_{j=1}^{m} L_j * \frac{1}{G_3}$$

(6)

where, the term L_j refer to the load of j^{th} VM, and normalizing factor is represented as G_3. Then, the computation of load in the cloud follows three conditions,

- If the load of PM is greater than the threshold, perform condition-1, otherwise no migration.
- If the load of VM is greater than threshold, perform condition-2.
- If the load of PM and that of VM is greater than threshold, perform condition-3.

Here, the mentioned condition-1, 2, and 3 are explained in the business phase as solution encoding. After that, if $TQR \le T$ it satisfies the following conditions.

- If the load of VM is greater than threshold, perform condition-4
- If the load of VM is lesser than threshold, perform condition-5.

6.5.2 Business phase

The actual information is checked in this business phase, and for all conditions, the placement algorithm, namely S-WOA is developed for selecting the suitable server (PM) for migration. The solution encoding, fitness function computation, and the algorithmic procedure of the proposed S-WOA are illustrated in the below section.

6.5.2.1 Solution encoding

Solution encoding is the representation of solution to be determined with the proposed algorithm. Here, for solution encoding five conditions are appropriate for better migration in cloud.

Condition-1

Figure 5 illustrates the solution encoding the developed model for condition-1. Here, the migration is performed from VM to PM. Here, the Proposed S-WOA selects the VM that are suitable for the migration. Let us consider that there are n number of PMs. The first VM operates under first PM, third VM operates under second PM and the fourth VM operates under m^{th} PM. Here, the PM is represented as, $\{1,2,...,n\}$, where n is the number of PMs. The index of VM is represented as, $\{1,3,...,4\}$

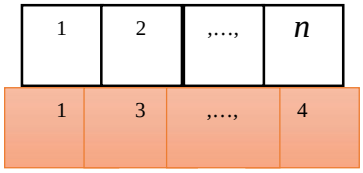

Figure 5: Solution Encoding for condition-1

Condition-2

Figure 6 represents the solution encoding the developed model for condition-2. In this condition, the migration is done from the container to VM under the same PM. Here, the Proposed S-WOA selects the container that are suitable for the migration. Assume the number of VMs in the corresponding PM is denoted as l. The first container operates under first VM in the corresponding PM, third container operates under second VM under the particular PM, and the fourth container operates under y^{th} VM in the corresponding PM. Here, the number of VMs in the corresponding PM is represented as, $\{1,2,...,l\}$, where l is the number of VMs in the corresponding PM. The index of container is represented as, $\{1,3,...,4\}$ under the particular PM.

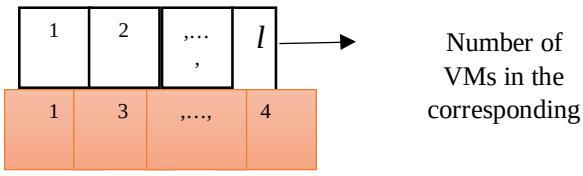

Figure 6: Solution Encoding for condition-2

Condition-3

The solution encoding of the developed model for condition-3 is illustrated as figure 7. In this case, the migration is performed from the container to VM under different PMs. Thus, the developed S-WOA choses the particular container for migration. Let us consider the number of VMs is denoted as m. Here, the first container operates under first VM, third container operates under second VM and the fourth container operates under p^{th} VM under different PMs.

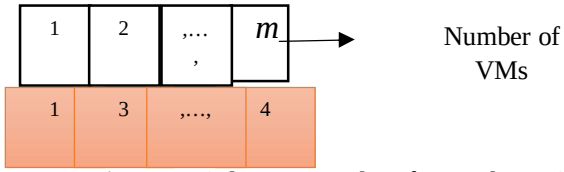

Figure 7: Solution Encoding for condition-3

The above-mentioned three conditions are followed when the TQR is greater than the threshold.

Condition-4

The solution encoding of the developed model for condition-4 is deliberated in Figure 8 illustrates. Here, the migration is carried out from tasks in VM to the container. Here, the Proposed S-WOA selects the tasks in VM, which is appropriate for the migration. Assume $1\times k$, where k denotes the number of containers, where the first tasks in VM operates under first container, second tasks in VM operates under second container, fourth tasks in VM operates under third container, and the fifth tasks in VM operates under $1\times k$ container. Here, the $1\times k$ number of containers is represented as, $\{1,2,3,...,1\times k\}$,. The index of tasks in VM is indicated as, $\{1,2,4,...,5\}$.

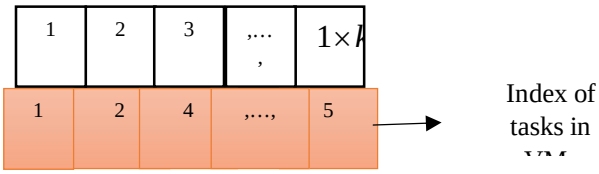

Figure 8 : Solution Encoding for condition-4

Condition-5

The solution encoding for condition-5 using developed model is depicted in **Figure 9**. Here, the migration is performed from container tasks to VM. In addition, the Proposed S-WOA selects the container tasks that are suitable for migration. Assume $1\times m$, where the term m represents the number of VM. Here, the first tasks in container operate under first VM, second tasks in container operate under second VM, sixth tasks in container operate under third VM, and the tenth tasks in container operates under $1\times m$ number of VM. Here, the $1\times m$ number of VMs is represented as, $\{1,2,3,...,1\times m\}$,. The index of tasks in container is indicated as, $\{1,2,6,...,10\}$.

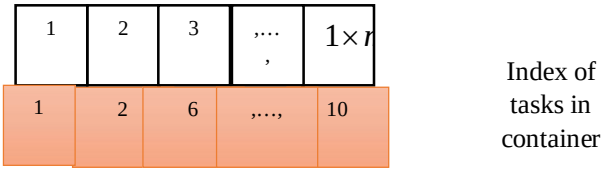

Figure 9: Solution Encoding for condition-5

The condition-4 and 5 should be followed when the TQR is less than or equal to threshold, and the load of VM is greater than, or less than or equal to threshold.

6.5.2.2 Fitness Evaluation

The fitness function is estimated for optimal solution determination from the solution set. The fitness of S-EWO is estimated using bandwidth, resource utilization, and load. The fitness with minimal value is the optimal solution for the migration in cloud. Here, the fitness function is computed for three conditions, and is given below. The fitness function computed for condition one is expressed as,

$$F_1 = \frac{1}{n}\sum_{i=1}^{n} S_i \tag{7}$$

where, the term F_1 is denoted as fitness function for condition-1, and the number of PMs is denoted as n.

$$S_i = \begin{cases} \frac{1}{2}\sum_{\substack{j=1 \\ j\in i}}^{m}\left(L_j + M_j\right) & ;if\ 1 is\ unloaded \\ 1 & ;if overloaded \end{cases} \tag{8}$$

where, the term L_j refer to the load of j^{th} VM, and the migration cost is denoted as M_j. Hence, the migration cost is computed using the expression given below.

$$M_j = \frac{1}{n}\sum_{i=1}^{n}\frac{u}{c*m} \tag{9}$$

where, the number of migrations is denoted as u, and migration constant is indicated as c. Then, for condition-2, and 3, the expression of fitness function is represented as,

$$F_2 = \frac{1}{n}\sum_{i=1}^{n} S_i \tag{10}$$

where, $S_i = \begin{cases} \frac{1}{2}\sum_{\substack{k=1 \\ k\in j}}^{p}\left(L_{ij}+M_{ij}\right) & ;if\ 1 is\ unloaded \\ 1 & ;if overloaded \end{cases}$, and the migration cost is expressed as,

$$M_{ij} = \frac{1}{m}\sum_{j=1}^{m}\frac{u^*}{c*p} \quad (11)$$

where, the term u^* indicates the number of migrations from the container to VM.For condition-4, and 5, the same fitness function equation of condition-1 should be followed.

6.5.2.3 Algorithmic procedure of the proposed Squirrel whale optimization algorithm

In this section, the migration approach is elaborated using the proposed S-WOA in order to select the appropriate server (PM) for migration in cloud computing platform. The S-WOA is developed by combining SSA, with WOA [26]. SSA is the simple and the powerful nature-inspired approach to tackle optimization issues. This approach uses dynamic foraging strategy of the southern flying squirrels and the way of locomotion is termed as gliding. On the other hand, WOA is duly based on hunting mechanism of humpback whales that searches for their prey based on the bubble-net attacking mechanism, which leads to optimal solutions. It is worth notable that the search for the prey is both associated within or outer the search spaces through a series of steps, like encircling, exploitation, and exploration. In WOA, an awareness probability is utilized, which helps to control the diversity of algorithm and is simpler to implement. By integrating the WOA with SSA, the parametric features from both the optimization are inherited, which boost the performance of classification accuracy. The algorithmic procedure for migration in cloud is given as,

Step 1

Initialization: The first step is the initiation of whale's population, which is given as,

$$X = \{X_1, X_2, \cdots, X_u, \cdots X_v\}; 1 \le u \le v \quad (12)$$

where, v denotes total population size and X_u indicate the u^{th} solution

Step 2

Evaluation of fitness function

The fitness for each solution is computed on the basis of fitness function depicted in equation (5) and equation (8) for consition-1, 2, and 3. The fitness function is taken as maximization function, and solution producing the maximum fitness is considered as the best solution.

Step 3

Position update using Squirrel whale optimization algorithm

The WOA algorithm assures better accuracy and the effectiveness for selecting the best sequences. According to the WOA algorithm [26], the update equation is given by,

$$X(h+1) = X^*(h) - RS \quad (13)$$

where, h indicate current iteration, R indicate coefficient vector, X^* denote the position vector of best solution, S indicate the behavior of best search agent.

$$X(h+1) = X^*(h) - R\,|\,K.X^*(h) - X(h)\,| \quad (14)$$

Where, K represent coefficient vector. Assuming $X^*(h)$ is greater,

$$X(h+1) = X^*(h)(1-RK) + RX(h) \quad (15)$$

(16)

In order to obtain global optimal solutions in migration, the SSA [25] is utilized in the algorithm. Thus, the update equation SSA is given as,

$$X(h+1) = X(h) + f_d T_y \times \left(X^*(h) - X(h)\right)Q_1 \quad (17)$$

As FS_{nt} in SSA refer to the position of flying squirrel that reached the hickory nut tree, which is the optimal solution. Hence, it can be considered as the best solution.

$$X(h+1) = X(h) + f_d T_y X^*(h)Q_1 - f_d T_y X(h)Q_1 \quad (18)$$

$$X^*(h) = \frac{X(h+1) - X(h) + f_d T_y X(h)Q_1}{f_d T_y Q_1} \quad (19)$$

Substituting equation (19) in equation (16), the solution becomes,

$$X(h+1) = \frac{X(h+1) - X(h) + f_d T_y X(h)Q_1}{f_d T_y Q_1}(1-RK) + RX(h) \quad (20)$$

$$X(h+1) = \frac{X(h+1)(1-RK)}{f_d T_y Q_1} + \frac{X(h)\left(f_d T_y Q_1 - 1\right)}{f_d T_y Q_1}(1-RK) + RX(h) \quad (21)$$

$$X(h+1) - \frac{X(h+1)}{f_d T_y Q_1}(1-RK) = \frac{X(h)\left(f_d T_y Q_1 - 1\right)}{f_d T_y Q_1}(1-RK) + RX(h) \quad (22)$$

$$X(h+1)\left[1 - \frac{1-RK}{f_d T_y Q_1}\right] = \frac{X(h)\left(f_d T_y Q_1 - 1\right)}{f_d T_y Q_1}(1-RK) + RX(h) \quad (23)$$

$$X(h+1)\left[\frac{f_d T_y Q_1 - 1 + RK}{f_d T_y Q_1}\right] = \frac{X(h)\left(f_d T_y Q_1 - 1\right)}{f_d T_y Q_1}(1-RK) + RX(h) \quad (24)$$

Thus, the final equation of the proposed S-WOA for migration in cloud computing platform is given by,

$$X(h+1)=\frac{f_d T_y Q_1}{f_d T_y Q_1+RK-1}\left[\frac{X(h)(f_d T_y Q_1-1)(1-RK)}{f_d T_y Q_1}+RX(h)\right] \quad (25)$$

where, the random gliding distance is denoted as f_d, the random number is represented as Q_1 ranging between 0 and 1, and the term h denotes the current iteration. The value of T_y is 1.9 that is achieved after the rigorous analysis. The terms R and K represents the coefficient vectors, which are based on the constants s, and t, and is expressed as

$$R = 2t.s - t \quad (26)$$

$$K = 2.s \quad (27)$$

The constant t decreases linearly from 2 to 0 indicating the switching between the exploration and the exploitation phases, and the constant s is a random vector that vary in the range $[0,1]$.

Step 4

***Checking the feasibility of solution*:**

The feasibility of the solution is computed based on the fitness function. If the newly generated solution is best than the previous one, then it is changed by the new solution.

Step 5

Termination:

Repeat the steps for the maximal iterations until the global optimal best solutions are determined. Thus, the optimization algorithm discussed in this section aims at determining the optimal weights for migration in cloud. The pseudo code of the developed S-WOA is depicted in Algorithm 1, which demonstrates step-wise description of the algorithm.

Algorithm 1: Pseudo code of proposed S-WOA algorithm

Input:Whales population $X = \{X_1, X_2, \cdots, X_u, \cdots X_v\}; 1 \le u \le v$
Output: X^* **best solution**
Procedure:
Begin
Initialize the population of whales $X = \{X_1, X_2, \cdots, X_u, \cdots X_v\}; 1 \le u \le v$
Compute the fitness of each solution using equation (5), and (8)
while $(h < h_{\max})$
for each search agent
If $l < 0.5$
Update the location of S-WOA using equation (20)
End if
End for
Ensure if any search agent go beyond search space
Update X^* **if there exist better solution**
$h = h + 1$
End while
Return X^*

6.5.2.3 Transformation phase

Based on the validation phase, the condition is determined, and the placement algorithm using the proposed S-WOA algorithm. In addition, the fitness for the corresponding condition is selected. Thus, the transformation of the VM information to the container information or vice versa in MON (Migration Object Notation) is done.

6.5.2.4 Deployment phase

Through the placement algorithm in business phase, the destination location for the migration and from the transformation phase the converted code is available, which is deployed according to the strategy required. Depending on the condition, the migration is deployed.

7. EXPERIMENTATION AND COMPARATIVE ANALYSIS

7.1 Experimental setup

The execution of the developed method is done in OpenStack in PYTHON using PC with the Windows 10 OS, 2GB RAM, and Intel i3 core processor.

7.2 Evaluation metrics

The performance of proposed S-WOA is employed for analysing the methods using number of instantiated VMs, CPU utilization, memory utilization, number of activated PMs, and time.

7.3 Computations and Data Analysis

The performance of the developed method is analysed by comparing the developed model with existing methods, like ACO-BF [1], Container VM-PM [2], and Multi-objective optimized replica placement method [3].

Analysis with number of tasks= 100

The comparative analysis based on metrics for migration in cloud using 100 tasks is depicted in figure 10.

Figure 10a illustrates the analysis using number of instantiated VM metric with different number of containers. For 50 containers, the number of instantiated VM measured by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 24, 24, 24, and 23. When container=80, the number of instantiated VM computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 24, 24, 24, and 23.

The analysis of methods using CPU utilization is deliberated in **Figure 10b**. For 50 containers, the CPU utilization computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 0.259, 0.273, 0.280, and 0.283. When container=80, the CPU utilization computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 0.407, 0.408, 0.420, and 0.460.

The analysis of methods using memory utilization parameter is deliberated in **Figure 10c**. For 50 containers, the memory utilization computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 0.256, 0.274, 0.300, and 0.305. When container=80, the memory utilization computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 0.309, 0.323, 0.372, and 0.377.

The analysis of methods using number of activated PMs metric is deliberated in **Figure 10d**. For 50 containers, the number of activated PMs computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 3,3,3,and 4. When container=80, the number of activated PMs computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 3, 3, 4, and 4.

The analysis of methods using time metric is deliberated in **Figure 10e)**. For 50 containers, the time computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 19.44sec, 18.99sec, 9.96sec, and 4.99sec. When container=80, the time computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 47.88sec, 35.19sec, 22.73sec, and 12.56sec.

Analysis with number of tasks=200

Figure 11 portrays analysis ofproposed S-EWO with number of tasks=200 based on performance metrics. **Figure 11a** illustrates the analysis using number of instantiated VM metric with different number of containers. For 50 containers, the number of instantiated VM measured by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 25, 24, 24, and 22. When container=80, the number of instantiated VM computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 24, 23, 23, and 23.

The analysis of methods using CPU utilization is deliberated in figure 11b. For 50 containers, the CPU utilization computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 0.298, 0.335, 0.341, and 0.364. When container=80, the CPU utilization computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 0.350, 0.455, 0.456, and 0.483.

The analysis of methods using memory utilization parameter is deliberated in **Figure 11c**. For 50 containers, the memory utilization computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 0.265, 0.285, 0.288, and 0.294. When container=80, the memory utilization computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 0.410, 0.448, 0.505, and 0.523.

The analysis of methods using number of activated PMs metric is deliberated in **Figure 11d**. For 50 containers, the number of activated PMs computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 3, 3, 4, and 4. When container=80, the number of activated PMs computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 3,3, 3, and 4.

The analysis of methods using time metric is deliberated in **Figure 11e**. For 50 containers, the time computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 24.35sec, 13.45sec, 12.76sec, and 6.53sec. When container=80, the time computed by ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method and proposed S-WOA are 63.67sec, 58.87sec, 34.49sec, and 17.63sec.

7.4 Comparative discussion

Table 1 deliberates the analysis of methods in terms of number of instantiated VMs, CPU utilization, memory utilization, number of activated PMs, and time with incoming tasks 100, and 200. The minimal number of

instantiated VMs of 23 obtained by proposed S-WOA, whereas the existing ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method are 25, 24, and 24. The maximal CPU utilization computed by S-WOA is 0.483, whereas the existing ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method are 0.350, 0.455, and 0.458 with incoming task=200, respectively. In addition, the maximal memory utilization value measured by S-WOA is 0.523, whereas the existing ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method are 0.410, 0.448, and 0.505 by considering task=100. The minimal time value computed by S-WOA is 4.99 sec, whereas existing ACO-BF, Container VM-PM, and Multi-objective optimized replica placement method are 19.44 sec, 18.99sec, and 9.96sec.

8. CONCLUSIONS

This paper presents the placement algorithm, namely S-WOA for migration in cloud. Generally, the cloud computing architecture consists of application layer, platform layer, infrastructure layer, and the hardware layer.
The Hardware layer is utilised to handle the physical resources of the cloud, including network devices physical hardware, and the power systems.

The Infrastructure layer is also said to be virtualization layer, which partitions the hardware and provides the pool of disk storage and computing resources.

The next layer is the platform layer, which broadly covers the operating systems and the application frameworks depending on every specific platform. Thus, this layer is utilized for reducing development efforts by providing the development platform to developers as the service without installing any framework or software's on their computers.

Application layer offers cloud applications to the end users as service. The OML is newly introduced and is placed above the application layer using the certain components, like validation phase, business Phase, transformation phase, and deployment phase.

In business Phase, the actual information is checked. Here the Placement algorithm is performed using optimization algorithm, namely S-WOA for selecting the suitable server (Physical Machine) for migration by satisfying five conditions.

The proposed S-WOA is designed by integrating SSA and WOA. Along with the proposed S-WOA method, a fitness function is considered using bandwidth, resource utilization and load.

The proposed S-WOA and the fitness function improved the overall network performance for better migration, and then the transformation and deployment phase are carried out based on condition.

The effectiveness of the proposed S-WOA is computed with respect to other methods and showed effective results with maximal CPU utilization of 0.483, maximal memory of 0.523, and minimal time of 4.99sec, respectively.

The work can be extended by determining the solutions for the limitations encountered in the survey of migration in cloud techniques.

REFERENCES

1. G. Mateusz, G. Alicja and B. Pascal," Cloud brokering: Current practices and upcoming challenges", IEEE Cloud Comput.,Vol: 2,no:2,pp: 40–47,2015. https://doi.org/10.1109/MCC.2015.32
2. A. S. Milani and N. J. Navimipour," Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends", J. Netw. Comput. Appl. Vol:71, no:1, pp: 86–98,2016 https://doi.org/10.1016/j.jnca.2016.06.003
3. Y. C. Jiang, "A survey of task allocation and load balancing in distributed systems", IEEE Trans. Parallel Distrib. Syst., Vol: 27no:2,pp: 585–599,2016
4. Jianting Ning, Zhenfu Cao, Xiaolei Dong, Kaitai Liang, Hui Ma and Lifei Wei," Auditable -Time Outsourced Attribute-Based Encryption for Access Control in Cloud Computing", IEEE Transactions on Information Forensics And Security, 2017.
5. Wayne Jansen and Timothy Grance, "Guidelines on security and privacy in public cloud computing", pp: 800-144, 2011. https://doi.org/10.6028/NIST.SP.800-144
6. Barcelo, M., Correa, A., Llorca, J., Tulino, A.M., Vicario, J.L. and Morell, A., "IoT-cloud service optimization in next generation smart environments," IEEE Journal on Selected Areas in Communications, vol.34, no.12, pp.4077-4090, 2016.
7. M. Barcelo, J. Llorca, A. Tulino, and N. Raman, "The cloud service distribution problem in distributed cloud networks," in IEEE ICC 2015 SAC - Data Storage and Cloud Computing, London, United Kingdom, Jun. 2015.
8. I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security," in proceedings of Federated Conference Computer Science and Information Systems (FedCSIS), on, pp. 1–8, September, 2014. https://doi.org/10.15439/2014F503
9. Jamshidi, P., Ahmad, A. and Pahl, C., "Cloud migration research: a systematic review," IEEE Transactions on Cloud Computing, vol.1, no.2, pp.142-157, 2013.
10. Liu X, Wang C, Zhou BB, Chen J, Yang T, Zomaya AY, "Priority-based consolidation of parallel workloads in the cloud," IEEE Trans Parallel DistribSyst, vol.24, no.9, 2012.
11. Yao F, Yao Y, Chen H, Li T, Lin M, Zhang X, "An intelligent scheduling algorithm for complex manufacturing system simulation with frequent synchronizations in a cloud environment," 2019. https://doi.org/10.1007/s12293-019-00284-3

12. Wiseman Y, Feitelson DG, "Paired gang scheduling," IEEE Trans Parallel DistribSyst, vol.14, no.6, pp.581–592, 2003.
13. Hussein, M.K., Mousa, M.H. and Alqarni, M.A., "A placement architecture for a container as a service (CaaS) in a cloud environment," Journal of Cloud Computing, vol.8, no.1, pp.7, 2019.
14. Zhang, R., Zhong, A.M., Dong, B., Tian, F. and Li, R., "Container-VM-PM architecture: A novel architecture for docker container placement," In proceedings of International Conference on Cloud Computing, Springer, pp. 128-140, 2018. https://doi.org/10.1007/978-3-319-94295-7_9
15. Li, C., Wang, Y., Tang, H. and Luo, Y., "Dynamic multi-objective optimized replica placement and migration strategies for SaaS applications in edge cloud," Future Generation Computer Systems, vol.100, pp.921-937, 2019.
16. Zaheer, S., Malik, A.W., Rahman, A.U. and Khan, S.A., "Locality-aware process placement for parallel and distributed simulation in cloud data centers," The Journal of Supercomputing, pp.1-23, 2019.
17. Zhou, R., Li, Z. and Wu, C., "An Efficient Online Placement Scheme for Cloud Container Clusters," IEEE Journal on Selected Areas in Communications, vol.37, no.5, pp.1046-1058, 2019.
18. Liu, B., Li, P., Lin, W., Shu, N., Li, Y. and Chang, V., "A new container scheduling algorithm based on multi-objective optimization," Soft Computing, vol.22, no.23, pp.7741-7752, 2018.
19. Nazir, B., "QoS-aware VM placement and migration for hybrid cloud infrastructure," The Journal of Supercomputing, vol.74, no.9, pp.4623-4646, 2018. https://doi.org/10.1007/s11227-017-2071-1
21. Satpathy, A., Addya, S.K., Turuk, A.K., Majhi, B. and Sahoo, G., "Crow search based virtual machine placement strategy in cloud data centers with live migration," Computers & Electrical Engineering, vol.69, pp.334-350, 2018.
22. JKRSastry, M TrinathBasu, Securing SAAS service under cloud computing-based multi-tenancy systems, Indonesian Journal of Electrical Engineering and Computer Science, Volume 13, Issue 1, Page 65-71, 2019 https://doi.org/10.11591/ijeecs.v13.i1.pp65-71
23. JKRSastry, M TrinathBasu, Securing Multi-tenancy systems through multi DB instances and multiple databases on different physical servers, International Journal of Electrical and Computer Engineering (IJECE), Volume 9, Issue 2, Pages 1385-1392, 2019
24. M. TrinathBasu, Dr.JKRSastry, A full security included Cloud Computing architecture, International Journal of Engineering & Technology, Volume 7, Issue 2.7, Page 807-812, 2018
25. JKRSastry, M TrinathBasu, Securing Multi-tenancy systems through user spaces defined within the database level, Jour of Adv Research in Dynamical & Control Systems, Volume 10, issue 7, Page 405-412, 2018
26. J. K. R. Sastry, K. Sai Abhigna, R. Samuel and D. B. K. Kamesh, Architectural models for fault tolerance within clouds at the infrastructure level, ARPN Journal of Engineering and Applied Sciences, VOL. 12, NO. 11, 2017, Pages 3463-3469,
27. DBK Kamesh, JKRSastry, Ch. Devi Anusha, P. Padmini, G. Siva Anjaneyulu, Building Fault Tolerance within Clouds at Network Level, International Journal of Electrical and Computer Engineering (IJECE), Vol. 6, No. 4, pp. 1560~1569, 2016
28. S. L. SUSHMITHA, Dr. D. B. K. J.K. R. SASTRY, V. V. N. SRI RAVALI, Y.SAI KRISHNA REDDY, building fault tolerance within clouds for providing uninterrupted software as service, Journal of Theoretical and Applied Information Technology, Vol.88. No.1, Pages 65-76, 2016
29. NVSPavan Kumar, Dr.JKRSastry, Dr. K Raja Sekhara Rao, Mining Distributed Databases for Negative Associations from Regular and Frequent Patterns, International Journal of Advanced Trends, Volume 8, Issue 4, Pages 1440-1463, 2019
30. NVSPavan Kumar, Dr. JKRSastry, Dr. K Raja Sekhara Rao, On Incremental mining Databases for Regular and Frequent Patterns, International Journal of Emerging Trends and engineering research, Volume 7, Issue 9, Pages 291-305, 2019 https://doi.org/10.30534/ijeter/2019/12792019
31. NVSPavan Kumar, Dr. JKRSastry, Dr. K Raja Sekhara Rao, Mining Negative Frequent regular Itemsets from Data Streams, International Journal of Emerging Trends and engineering research, Volume 7, Issue 8, Pages 85-98, 2019 https://doi.org/10.30534/ijeter/2019/02782019
32. M. TrinathBasu, JKRSastry, Improving the Open Stack Authentication system through federation with JASON Tokens, International Journal of Advanced Trends n Computer Science and Engineering, 3596-3614,2019. https://doi.org/10.30534/ijatcse/2019/143862019
33. JKR Sastry, M TrinathBasu, Multi-Factor Authentication through Integration with IMS System, International Journal of Emerging Trends in Engineering Research, Volume 8, Issue 1, 2020, PP. 87-113 https://doi.org/10.30534/ijeter/2020/14812020
34. JKRSastry, M TrinathBasu, Strengthening Authentication within OpenStack Cloud Computing System through Federation with ADDS System, International Journal of Emerging Trends in Engineering Research, Volume 8, Issue 1, 2020, PP. 213-238 https://doi.org/10.30534/ijeter/2020/29812020
35. JKR Sastry, M TrinathBasu, Enhancing Data Security under Multi-Tenancy within Open Stac, International Journal of Advanced Trends in Computer Science and Engineering, Volume 8, Issue 1, 2020, PP. 533-544. https://doi.org/10.30534/ijatcse/2020/73912020
36. Dr.JKRSastry,B. TrinathBasu, Enhancement of Security within OpenStack – Some measures, , International Journal of Emerging Trends in Engineering Research, Volume 8, Issue 3, 2020, PP. 919-938 https://doi.org/10.30534/ijeter/2020/49832020

37. K. Aruna Kumari, J. K. R. Sastry, K. Rajasekhara Rao, Energy Efficient Load Balanced Optimal Resource Allocation Scheme for Cloud Environment, International Journal of Recent Technology and Engineering(IJRTE), Volume-8 Issue-1S3, 2019, pp. 146-153
38. K. Aruna Kumari, K. Raja Sekhara Rao, JKR Sastry, ON SPEEDING UP VIRTUAL MACHINE MIGRATION THROUGH INTEGRATED DATA DE-DUPLICATION METHODS, ARPN Journal of Engineering and Applied Sciences, VOL. 13, NO. 5, 2018, pp. 1957-1964
39. K. Aruna Kumari, K. Raja Sekhara Rao, J.K.RSastry, On Speeding up Virtual Machine Migration Through Integrated Data De-Duplication Methods,Research Journal of Applied Sciences 12 (2): 131-138, 2017
40. Jain, M., Singh, V. and Rani, A., "A novel nature-inspired algorithm for optimization: Squirrel search algorithm," Swarm and evolutionary computation, vol.44, pp.148-175, 2019.
41. Mirjalili, S. and Lewis, A., "The whale optimization algorithm," Advances in engineering software, vol.95, pp.51-67, 2016.

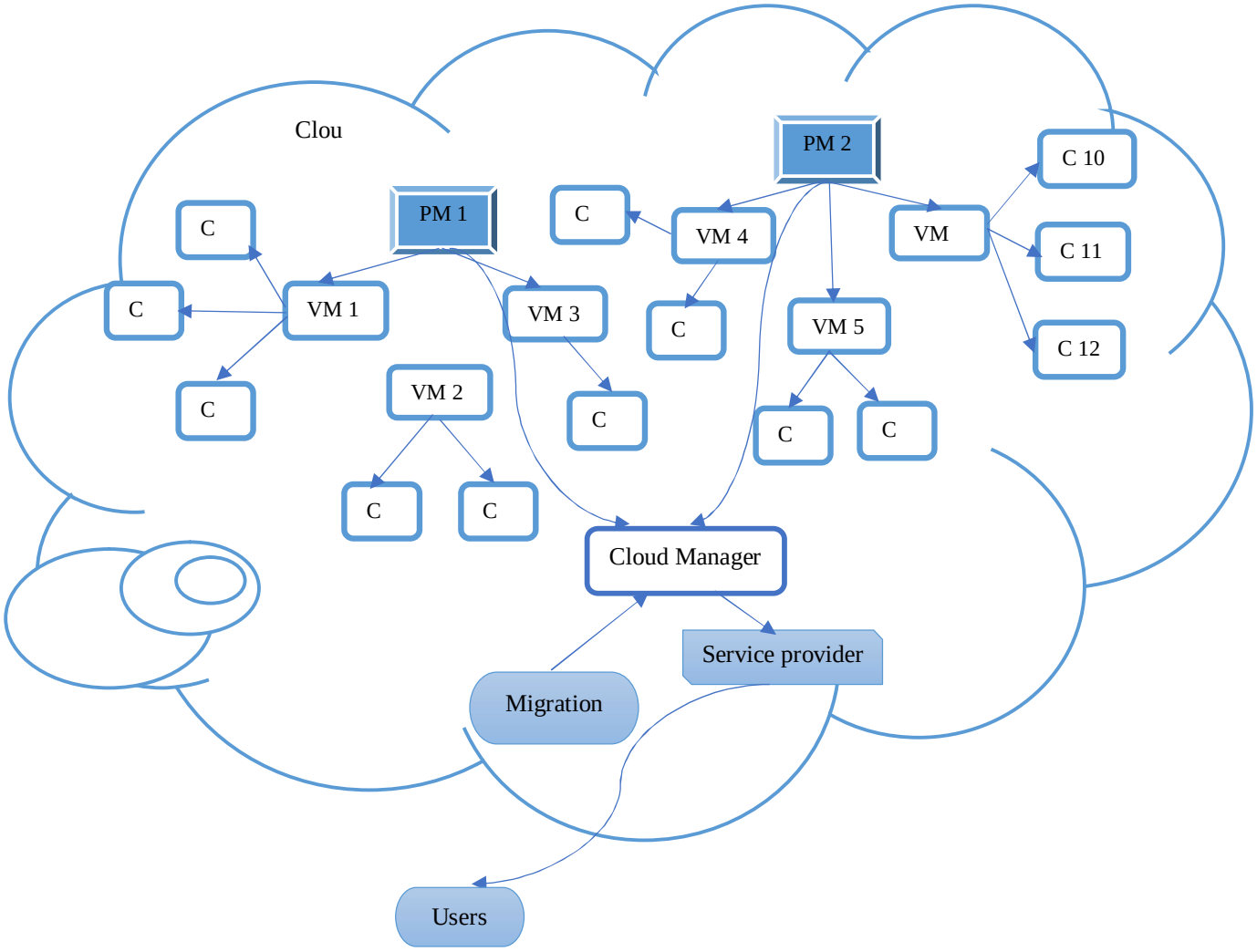

Figure 2: System model of migration in cloud platform

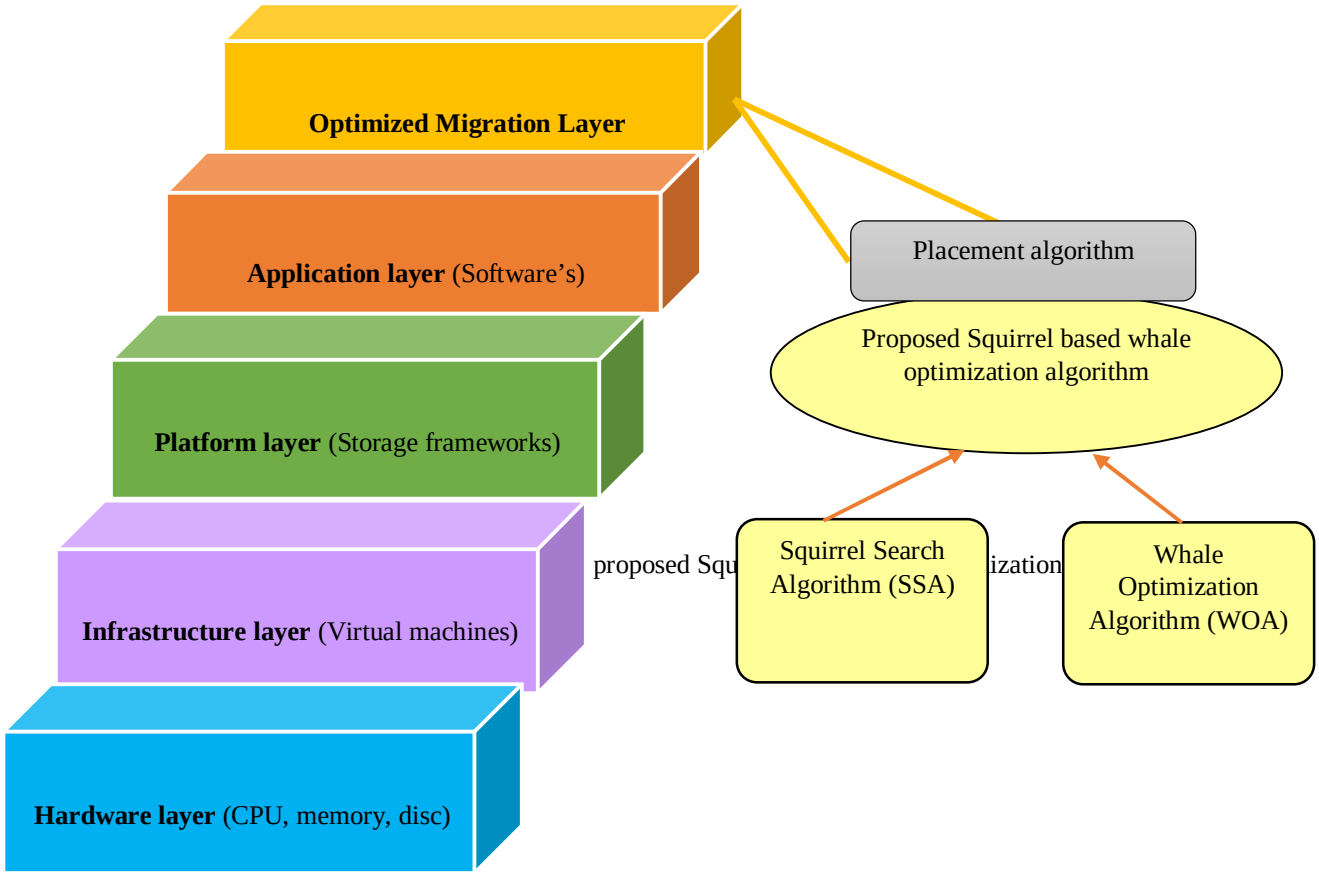

Figure 3: Cloud computing architecture using proposed Squirrel based whale optimization algorithm

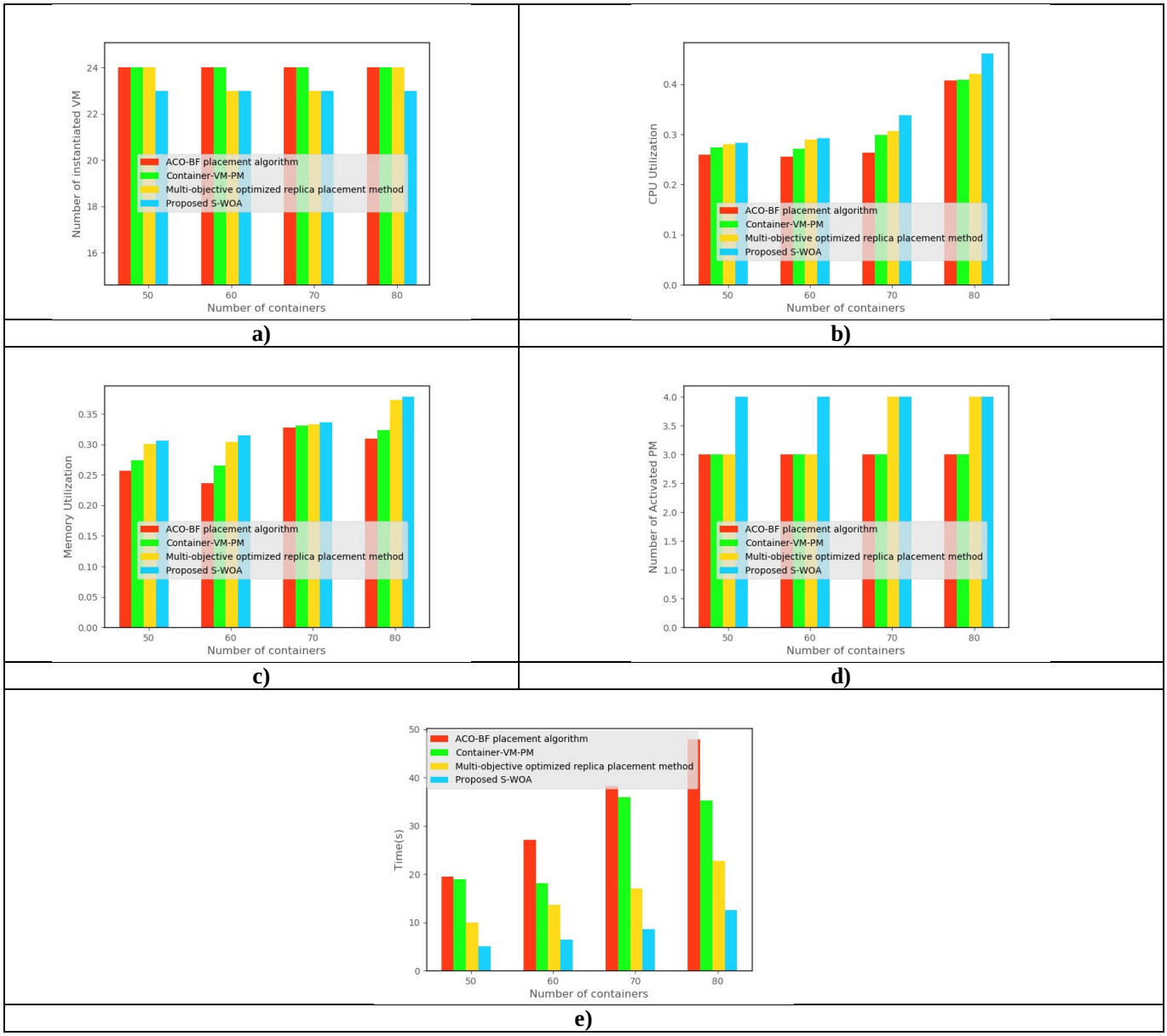

Figure 10.: Analysis with number of tasks=100 a) number of instantiated VM b) CPU utilization c) memory utilization, d) number of activated PMs, and e) time

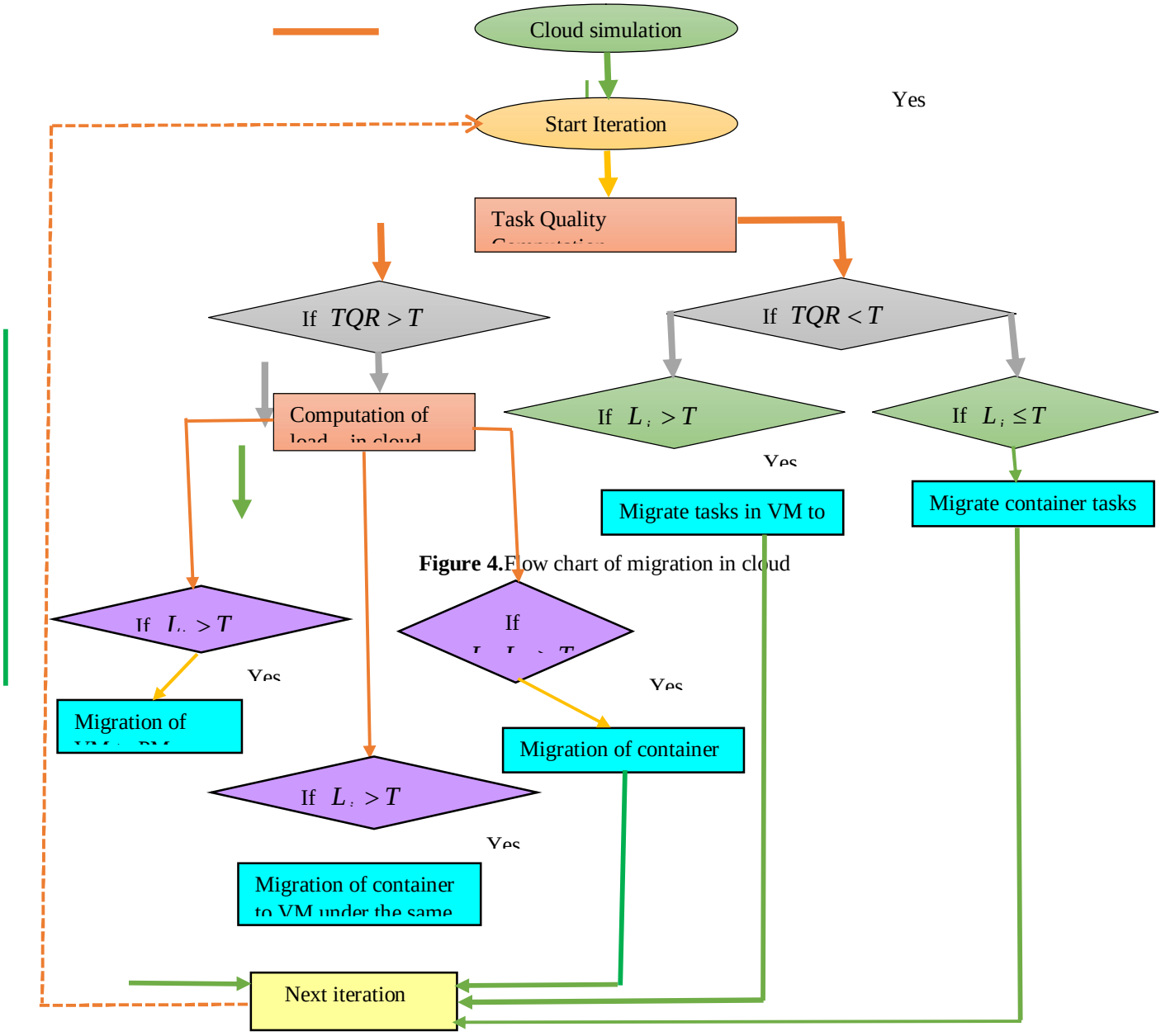

Figure 4. Flow chart of migration in cloud

Figure 4: Flow chart of migration in cloud

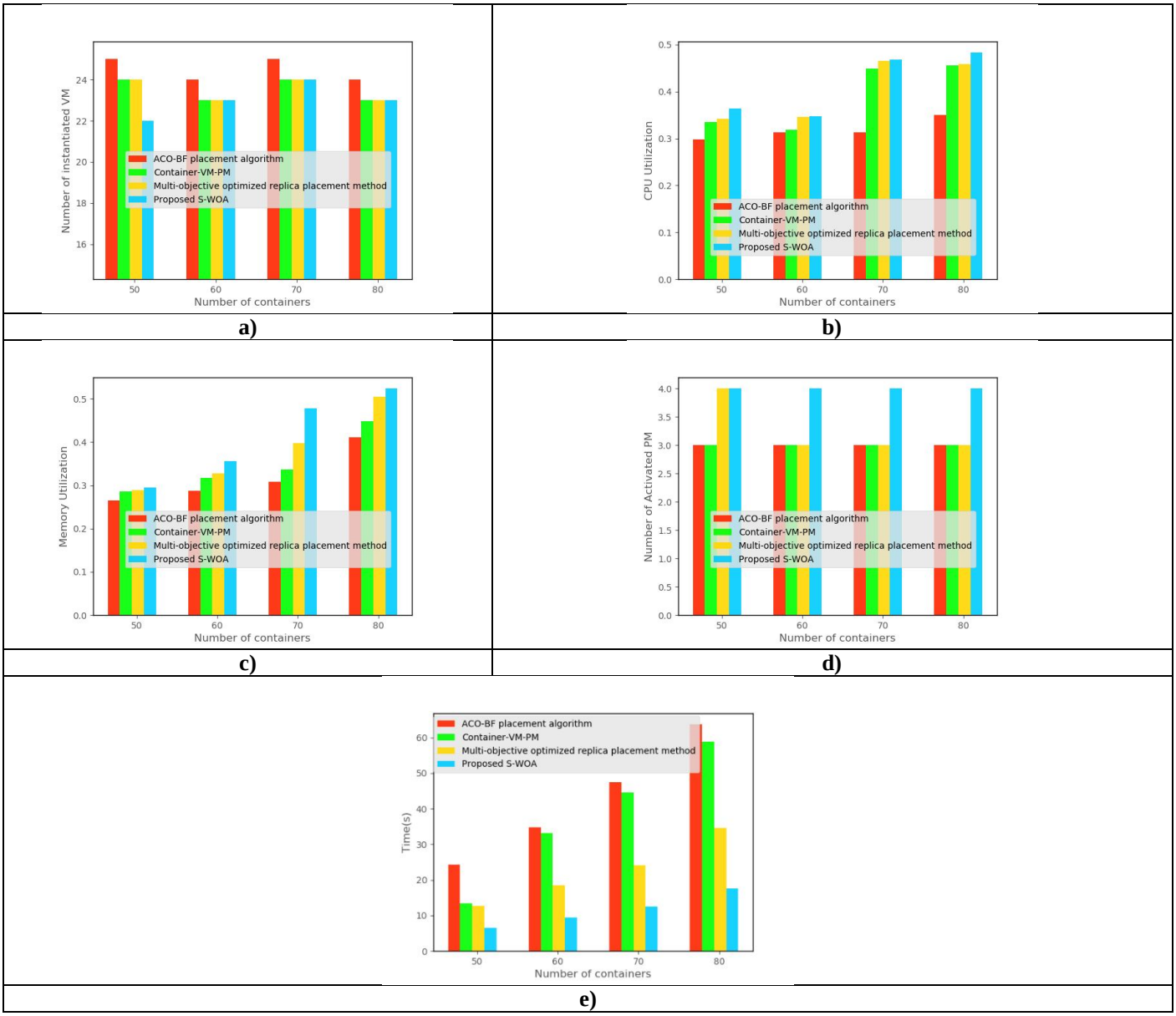

Figure 11: Analysis with number of tasks=200 a) number of instantiated VM b) CPU utilization c) memory utilization, d) number of activated PMs, and e) time

Table 1: .Comparative analysis

Incoming Tasks	Metrics	ACO-BF	Container VM-PM	Multi-objective optimized replica placement method	Proposed S-WOA
100	***Number of instantiated VMs***	24	24	24	**23**
	CPU utilization	0.407	0.408	0.420	**0.460**
	Memory utilization	0.309	0.323	0.372	**0.377**
	Number of activated PMs	2	3	3	**4**
	Time (Sec)	19.44	18.99	9.96	**4.99**
200	***Number of instantiated VMs***	25	24	24	**23**
	CPU utilization	0.350	0.455	0.458	**0.483**
	Memory utilization	0.410	0.448	0.505	**0.523**
	Number of activated PMs	3	3	3	**4**
	Time (Sec)	24.35	13.45	12.76	**6.53**