Received: December 17, 2018

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Meta-heuristic Techniques for Placement of Virtual Machines in Distributed Cloud Centers

Kumaraswamy Siddagangaiah[1]* **Mydhili Krishnan Nair[2]**

[1]*Department of Computer Science and Engineering,*
Global Academy of Technology, Bengaluru 560098, India
[2]*Department of Information Science and Engineering,*
Ramaiah Institute of Technology, Bengaluru 560054, India
* Corresponding author's Email: skswamy99@gmail.com

Abstract: The Virtual Machine Placement techniques provide an attractive opportunity to achieve the goal of energy conservation in Cloud Centers. The Virtual Machines can be broadly classified into data intensive and CPU intensive based on their workload characteristics. Placing or migrating large number of data intensive Virtual Machines can lead to degradation of task execution efficiency in Cloud Centers. Hence, the Virtual Machine Placement techniques need to consider the Virtual Machine workload characteristics to prevent associated negative consequences. Even though workload characteristics aware Virtual Machine migration technique has been presented in the literature for single-location Cloud Centers, it does not cater to Distributed Cloud Centers, where the cloud resources are distributed in different geographical locations, and in such setting, new performance issues arise which have to be effectively addressed. In this work, the NP-Complete Bin Packing Problem framework is extended to model the Workload Characteristic aware Virtual Machine Placement problem in Distributed Cloud Centers. The proposed solution is based on two meta-heuristic techniques having polynomial complexity: Particle Swarm Optimization and Ant Colony Optimization technique. The proposed techniques are compared against the contemporary technique in simulated environment. In the simulated study of the proposed techniques, both these techniques discover approximate solutions to the Bin Packing Problem which are extremely close to the optimal solution, and thereby directly contribute in efficient load distribution in Distributed Cloud Centers.

Keywords: Cloud computing, Virtual machine placement, Distributed cloud data centers, Bin packing, Meta-heuristic techniques.

1. Introduction

1.1 Overview on virtual machine placement

Cloud Center (CC) refers to commercial computational service center, which provides computational services to clients on an ad-hoc basis over the Internet. Cloud Computing limits the effort and cost required to maintain and procure computational resources and software systems for the organizations. In-fact, the organizations can access the Cloud services when and where required. Currently, many organizations are moving their operations to the CCs. This trend has resulted in the enormous expansion of CCs.

The CCs consists of numerous Physical Machines (PM), and each PM is associated with number of Virtual Machines (VMs), which abstracts the PM [1]. Most of the services are provided to the clients through the VMs. In certain scenarios, some of the PMs might have very few active VMs, and a recent study [2] has revealed that, even a single active VM can result in 50% power consumption in the corresponding PM. If such VMs are migrated to other PMs which have multiple active VMs, then, the PMs from which migration has been performed can be shutdown, which results in reduction in total power consumption in the CCs. The problem of

migrating the required VMs to other PMs is known as -- Virtual Machine Placement (VMP) or Virtual Machine Migration -- problem.

Each VM is associated with an image, which stores all the data and code corresponding to the VM. This image is stored in the PM on which the VM executes. Since, distributed architecture is prevalent in most of the cloud centers, the image can be accessed from other PMs.

The existing perception about VMs is that, all of them are similar in their workload characteristics. Recently, it has been shown that [2], there are two broad classifications of VMs based on their workload characteristics: data intensive and CPU intensive VMs. The former involves large proportion of data intensive tasks, and the latter involves large proportion of CPU intensive tasks.

If a data intensive VM is migrated without copying its corresponding image to a new PM, performance degradation of around 40% WRT execution efficiency of tasks inside these VMs is observed [2]. However, the equivalent impact is not observed regarding CPU intensive VMs. It can be inferred that, migrating large number of data intensive VMs away from their images is not recommended [2].

If a PM has large number of data intensive VMs, then, competition for accessing the disk resources can increase significantly, which in-turn can degrade the task execution efficiency performance. Hence, the number of data intensive VMs running in a PM has to be limited [2].

The VM migration technique presented in [2] was based on VM workload characteristics, and considered non-Distributed CCs -- here:(1) workload characteristics of VMs were considered for classifying them as data intensive or CPU intensive VMs; (2) VM migration was performed by considering these classified VMs. The proposed VM migration technique in [2], achieved noticeable task execution efficiency improvement when compared to the VM migration technique which did not consider VM workload characteristics.

1.2 Motivation

Currently, computational load in the CCs is increasing rapidly due to the popularity of Cloud Computing. The CCs are not just confined to a single geographical location [1], and such CCs are denoted as Distributed CCs (DCCs). Obviously, it can be hypothesized that, by employing workload characteristics based VM migration technique which was presented in [2], effective VM migration in even DCC environments can be achieved. However, the associated unpredictability in accurately predicting the future computation load in different locations of the DCC is usually high [1]. Hence, performing large number of VM migration to different geographical locations may lead to frequent migration scenario [1], which can seriously degrade the efficiency of DCC. Ideally, it is recommended that, each PM should host limited number of VMs migrated from different geographical locations [1]. Thus, in DCC environments, classifying VMs as data intensive and CPU intensive will not suffice, and further classification of these VMs based on their locations is also essential. However, the presented VM migration technique in [2], the issue of further classifying the VMs based on their locations for DCC environments, is not addressed, and which will be addressed in this work by suitably extending the VM migration technique presented in [2], and such endeavor has not been carried out previously in the literature [1].

1.3 Contributions

In this presented work, the following contributions are made:

1. The VMP problem for DCC environment which is based on VM Workload characteristics, is modeled through 3–slot Bin Packing Problem (BPP). The NP completeness justification of 3–slot BPP is presented. Two meta-heuristic solutions based on Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) techniques respectively are presented.
2. The proposed solution techniques are evaluated through MATLAB simulation, and compared against the contemporary technique [1]. Both these solution techniques exhibit their effectiveness by providing solutions that are closer to their optimal counterparts, and with appreciable execution efficiency.

1.4 Paper organization

The paper is organized as follows: Section 2 describes the related work. Section 3 presents the proposed VMP techniques. Section 4 presents the empirical results, and finally, the work is concluded in Section 5.

2. Related work

The VM migration techniques presented in the literature can be broadly classified into: Live and Non-Live techniques; wherein, the former indicates

that, the VM migration does not affect normal services of the Cloud, and in the latter, such constraint is not enforced.

The Non-Live migration techniques have become unpopular in the literature due to the issue of halting of services during VM migration, because such issue cannot meet the required Service Level Agreement (SLA) of many clients who would not prefer halting of such services. Some of the popular Non-Live migration techniques and their characteristics have been outlined in [1].

The Live VM migration techniques are the most preferred techniques in the literature [1], which are further classified into: Memory migration techniques, Storage migration techniques and Network connection migration techniques.

The memory migration techniques aim at continuing the current processing of those VMs which have to be migrated, and to achieve this goal, the memory contents of these VMs are also transferred; along with the memory content, CPU current state is also migrated. Many variations of memory migration techniques have been presented in the literature recently: hybrid memory copy-oriented technique [3]; efficiency-oriented technique [4]; multiple application-oriented technique [5]; workload prediction-oriented technique [6]; NFV featured network-oriented technique [7]; Traffic sensitive technique [8]; Agile live migration technique [9]; Hash-Table oriented technique [10]; VNF live migration technique [11]. However, these techniques do not specifically address the issue of VM migration in DCCs by considering VM workload characteristics.

In certain cloud environments, a scenario might occur; wherein, the data storage of the source server from which a certain VM has to be migrated might not be accessible to the target server to which the VM is migrated. In such scenarios, Virtual Disks from the source server are migrated to the target server, and the VM migration techniques which address such scenarios are indicated as Storage Migration techniques. Storage Migration techniques are usually required in WAN environments. Many varied contributions have been recently made in the literature WRT Storage Migration techniques: time-constraint oriented technique [12];caching based technique [13]; geographically shifting CC based technique [14]; machine learning based technique [15]; three-layer image-based technique [16]. However, again, all these contributions do not specifically address the issue of VM migration in DCC environment by considering VM workload characteristics.

The VMs which get migrated to another server or Physical Machine (PM) has to become accessible to its users. In some PMs, due to system restrictions, the migrated VMs might not be available to the clients at the required flexibility, and new constraints might be imposed. Hence, the Network Connection migration techniques aim to achieve VM migration by providing the VM services to the user at the required flexibility, and this goal is achieved through varied tools and techniques which have been recently presented in the literature: topology adaptive technique [17]; SDN-based technique [18]; optical circuit switching oriented technique [19]; seamless migration-oriented technique [20]. Since, in this paper, the problem of effective VM migration in DCC environments is considered without specifically addressing the Network Connection migration technique, hence, these Network connection migration techniques are further not considered in this paper.

Recently, VM migration in DCC environment was presented in [21]. It was highlighted in [21] that, carbon footprint is a major concern in DCC environment, which also results in higher operation costs for CCs -- hence, a VM migration technique which specifically targets reduction in carbon footprint was presented, and the VM migration technique considered the CC locations which are powered by renewable energy sources as one of the important parameters in migration of VMs. However, in [21], the issue of task execution efficiency after VM migration in DCC environment has been completely ignored, without addressing this issue, the reduction in carbon footprint may not correlate to improvement inresource utilization and task execution efficiency.

In [22] authors developed system to minimize power consumption and resource wastage in virtual machine placement using ant colony optimization with hypercube framework. In [23] generalized online optimization methodology was used to place as many virtual machines as possible in both non-distributed and distributed cloud data centers to increase the cloud service provider's revenue.

From the above outlined Related Work, it is clear that, the problem of designing workload specific VMP technique for DCCs, has not been effectively addressed in the literature, and this problem will be effectively addressed in the subsequent sections.

3. Meta-heuristic techniques for VMP

3.1 Problem framework

The 3–slot BPP has n items, which are exclusively divided into three classes indicated by I_1, I_2 and I_3. Here, $|I_1| = n_1$, $|I_2| = n_2$, $|I_3| = n_3$, and $n_1+n_2+n_3 = n$. The term $i_{kj}(1 \leq k \leq 3)(1 \leq j \leq n_k)$ indicates the j^{th} item of the class I_k. Each i_{kj} is associated with the corresponding item weight indicated by $0 < w_{kj} < 1$. Each bin has a maximum item weight capacity of 1. The weight capacity of each bin is divided into three slots. The first slot has a maximum weight capacity indicated by $C_1(0 < C_1 < 1)$, and only the items ϵ I_1 and I_3 can be stored. The second slot has a maximum weight capacity indicated by $C_2(0 < C_2 < 1)$, and only the items ϵ I_2 and I_3 can be stored. The third slot has a maximum weight capacity indicated by $C_3(0 < C_3 < 1)$, and only the items I_3 can be stored. Here, $C_1+C_2+C_3 = 1$. The goal of BPP is to store the n items in the fewest bins possible.

The VMP problem in DCC is modeled by using the 3–slot BPP. Here, n indicates the number of VMs that are subjected to placement in the DCC. Each bin corresponds to a PM. The first slot of each bin can store either different location data intensive VMs (I_1) or same/differentlocation CPU intensive VMs (I_3). The second slot of each bin can store either same location data intensive VMs (I_2) or same/different location CPU intensive VMs. The third slot can only store same/different location CPU intensive VMs. The weight of each item i_{kj} is calculated as represented in Eq. (1). Here, $\alpha_k (0 < \alpha_k < 1)$ indicates the tuning parameter for k^{th} slot, which ensures that, $0 < w_{kj} < C_k$, and $size(i_{kj})$ indicates the size of i_{kj}.

$$w_{kj} = \alpha_k size(i_{kj}) \tag{1}$$

A solution to the 3–slot BPP which satisfies all the specified constraints is denoted as *feasible solution*. The task is to find the most optimal feasible solution, which corresponds to placement of VMs in the fewest possible PMs.

3.2 Problem complexity

The complexity of the proposed 3–slot BPP in a simple instance, wherein, $|I_1| = 0$ and $|I_2| = 0$ is presented in Theorem 1. Since, this problem is NP-complete for this simple instance, if, $|I_1| \neq 0$or $|I_2| \neq 0$, then, the complexity of 3–slot BPP further increases. Hence, approximate solution techniques, which have polynomial complexity need to be designed. In this work, PCO and ACO based approximate solution techniques are presented.

Theorem 1. *If, $|I_1| = 0$ and $|I_2| = 0$, then, the 3–slot BPP is NP-complete.*

Proof. Consider a simple instance of 3 slot BPP, wherein, $|I_1| = 0$ and $|I_2| = 0$. Clearly, 3–slot BPP transforms into BPP, which is NP-complete, and which makes this instance of 3–slot BPP also NP-complete.

3.3 ACO based solution technique

The ACO technique is based on the principle of *Biologically Inspired Computing* [24]. This meta-heuristic technique provides an approximate solution to the optimization problem. The main design of ACO technique is inspired from the path discovery process utilized by ants to discover shortest path between food source and their nest. Multiple ants are involved in the approximate solution discovery, and each ant builds its own approximate solution, which is based on the information collected by other ants. Initially, each ant utilizes an empty bin and set of all n items. The items are incrementally added to the corresponding class slots of the empty bin ensuring that, the specified constraints of 3–slot BPP are not violated. If the new items, which are yet to be placed, cannot fit into this existing bin, then, new bin is created. Each ant chooses the next item randomly, to build the incremental approximate solution. This item selection probability is represented in Eq. (2). Here, $p_k(b, s, j)$ indicates the probability that, the k^{th} ant selects the j^{th} item for bin b w.r.t. partial solution s, β is a suitable tuning parameter, $J_k(s, b)$ indicates the qualified set of items, which can be included in the current/newly created bin, $\eta(j)$ indicates the weight of j^{th} item, $\tau_b(j)$ indicates the *pheromone value* of j^{th} item in bin b, which signifies the merit of selecting j^{th} item for bin b, and it is calculated as represented in Eq. (3).

$$p_k(b,s,j) = \begin{cases} \frac{[\tau_b(j)][\eta(j)]^\beta}{\sum_{g \epsilon J_k(s,b)}[\tau_b(g)][\eta(g)]^\beta}, & if\ j \in J_k(s,b) \\ 0, & otherwise \end{cases} \tag{2}$$

$$\tau_b(j) = \begin{cases} \frac{\sum_{i \epsilon b}\tau(i,j)}{|b|}, & \text{if } b \neq \phi \\ 1, & otherwise \end{cases} \tag{3}$$

Consider Eq. (3), and the two items–item j and item i, which belong to the same class. Here, $\tau(i,j)$ indicates the merit of adding j^{th} item into bin b when i^{th} item is already in bin b, and this metric is calculated as represented in Eq. (4). Here, s^{best} indicates the best partial solution found until now, ρ

is a tunable parameter, *m* indicates the number of times the pair of items in which the weight of first item = $\eta(i)$ and the weight of second item = $\eta(j)$ is added to s^{best}, and $f()$ indicates the fitness function represented in Eq.(5). Here, *N* indicates the number of used bins, *c* is a tunable weight parameter and F_i indicates the total left over capacity of the i^{th} bin by considering all the three slots.

$$\tau(i,j) = \rho\tau(i,j) + mf(s^{best}) \quad (4)$$

$$f(s) = \frac{\sum_{i=1}^{N} F_i^c}{N} \quad (5)$$

Algorithm 1 outlines the ACO based solution technique. The *r* ants indicated by (a_1, a_2, ….a_r) are initialized w.r.t. their corresponding empty bin and set of *n* items through the function initialize(a_1, a_2, …. a_r,*n*). The i^{th} ant ($1 \leq i \leq r$) builds its corresponding partial solution *s* by considering the remaining unplaced items. The placement of items is performed for the newly created bin indicated by *b*, and it is achieved through the function *item_placement*($J_i(s,b)$,*flag*),which attempts to place the unplaced item set indicated by $J_i(s,b)$ based on the ACO item placement principle outlined above, here, each item ϵ $J_i(s,b)$ is considered to be placed in its corresponding class slot in the bin *b*, and if, the placement of any item $\epsilon J_i(s,b)$ in *b* is not possible, then, *flag* is set as 1, which initiates a new bin creation through the function *create_new_bin*(a_i). This item placement continues until all the items are placed.

Algorithm 1ACO Based Solution Technique

initialize(a_1, a_2, ……. a_r, n)
for *i* = 1 to *r* **do**
  while $J_i(s,b) \neq \phi$ **do**
    flag = 0
    *item*_placement($J_i(s,b)$, *flag*)
    if *flag* == 1**then**
      create_new_bin(a_i)
    end if
  end while
end for

Theorem 2 indicates that, if the number of ants used in Algorithm 1, is sufficiently large, then, with high chances, the approximate solution obtained for the 3-slot BPP will be closer to optimal solution.

Theorem 2. *For Algorithm 1, if the value of r is sufficiently large, then, the chances of obtaining the approximate solution to 3-slot BPP closer to optimal solution is high.*

Proof. Let, S_o and S_a indicate the optimal solution and approximate solution obtained through Algorithm 1 respectively, for 3-slot BPP. Assume a scenario that, $S_a - S_o$ is large. The ACO technique effectiveness is significantly influenced through *r*, because a greater number of ants lead to effective calculation of τ_b , which in-turn leads to better approximate solution production [15]. Hence, the chances of the assumed scenario occurring when *r* is sufficiently large is minimal, which immediately proves the theorem.

Theorem 3 establishes that fact that, Algorithm 1 executes in polynomial time complexity. Let, $ant\_search\_time(a_i)$ indicate the execution time for a_i . If, each a_i is assigned a separate computational node for executing a_i, then, it is clear that, the complexity of Algorithm 1, will be $\approx$ $ant\_search\_time(a_i)$. Suppose, all a_i execute in a single node, then, it leads to sequential execution of Algorithm 1, which leads to the Algorithm 1 complexity of $\approx r \times ant\_search\_time(a_i)$.

Theorem 3. *Algorithm 1 executes in polynomial time complexity*.

Proof. In Algorithm 1, each a_i picks the item randomly, and places it in the required bin based on the probability score denoted by p_k. Form this item placement procedure, it is clear that, this procedure executes in *O(n)* complexity. Hence, the proof is completed.

3.4 PSO based solution technique

PSO technique [24] is another meta-heuristic technique which provides an approximate solution to the optimization problems, and it is inspired by the social behavior of birds. The search for optimal solution is carried out by group of particles, where in, each particle has an exclusive zone in the feasible solution space, and union of all particle zones is equal to the feasible solution space. Each point in the feasible solution space represents a feasible solution vector. The particles are continuously moving in their corresponding feasible solution space to identify the optimal solution, and are involved in continuous communication for exchanging their locally discovered best solution, which in-turn decides the corresponding velocity of the particle for navigation. The particles continue their search until acceptable solution is obtained.

The PSO based solution technique for 3–slot BPP utilizes r particles. Here, the current position of the i^{th} particle at iteration t is indicated by $\vec{X}_i(t)$, and the position for the next iteration is indicated by $\vec{X}_i(t+1)$, which is calculated as represented in Equation 6. Here, $\vec{V}_i(t)$ indicates the velocity of i^{th} particle for t+1 iteration, and it is calculated as represented in Equation 7. Here, D_1 and D_2 indicate the degree of particle attraction towards individual and group success respectively, $\vec{x}_{gbest}$ and $\vec{x}_{pbesti}$ indicate the global best solution obtained by all the particles until the current iteration and the local best solution obtained by the i^{th} particle until the current iteration respectively, W indicates a control variable, and $r_1, r_2 \in [0,1]$ indicate the random factors.

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \vec{V}_i(t+1) \quad (6)$$

$$\vec{V}_i(t+1) = W\vec{V}_i(t) + D_1 r_1(\vec{x}_{pbest_i} - \vec{X}_i(t)) + D_2 r_2(\vec{x}_{gbest} - \vec{X}_i(t)) \quad (7)$$

The PSO based solution technique for 3–slot BPP is outlined in Algorithm 2. Here, *initialize_PSO*(P) divides the feasible solution space among the r search particles indicated by $P = p_1, p_2 \ldots\ldots p_r$, and assigns each particle to some arbitrary positions in their corresponding feasible solution space. Each particle calculates its feasible solution for the corresponding current position through *compute_score*($\vec{X}_i$ (t)), which utilizes Equations 6 and 7. The values for $\vec{x}_{pbest_i}$ and $\vec{x}_{gbest}$ are calculated through *local_best*($score_i$) and *global_best*(P, $\vec{x}_{pbest_i}$) respectively. The particles continue to search until the acceptable solution is found, and which is calculated through acceptable ($\vec{x}_{gbest}$).

Algorithm 2 PSO Based Solution Technique

$P = p_1, p_2, \ldots\ldots p_r$
initialize_PSO(P)
flag = 0
t = 0
while *flag* == 0 **do**
  $t = t + 1$
  for i = 1 to r **do**
    $score_i$ = *compute_score* ($\vec{X}_i$(t))
    $\vec{x}_{pbest_i}$ = *local_best*($score_i$)
    $\vec{x}_{gbest}$ = *global_best*(P, $\vec{x}_{pbest_i}$)
    if *acceptable*($\vec{x}_{gbest}$) **then**
      flag = 1
    end if
  end for
  $t = t + 1$
end while

The effectiveness of Algorithm 2 in producing approximate solutions WRT optimal solution for 3-BPP is established through Theorem 4. The polynomial complexity property of Algorithm 2 is established through Theorem 5. It is clear from the proof of Theorem 5 that, by exploiting parallelism, substantial improvement in complexity can be obtained for Algorithm 2.

Theorem 4. *For Algorithm 2, If the number of particles (r) is sufficiently large, then, the chances of obtaining the approximate solution to 3-slot BPP closer to optimal solution is high.*

Proof. Let, N_{so} indicate the number of feasible solutions. Assume a scenario where $2r = N_{so}$. Also, each particle will have at least 2 iterations. In this scenario, it is clear that, the optimal solution will be found, because some particle will reach it. However, this search will not have polynomial complexity -- if the number of computing nodes are very limited. The inference from this scenario is that, with large number of particles, the search effectiveness WRT quality of approximate solution improves. By using this inference, it can be deduced that, having sufficiently large value for r such that, the search complexity does not become non-polynomial, the chances of obtaining the approximate solution to 3-slot BPP closer to optimal solution is high.

Theorem 5. *Algorithm 2 executes in polynomial time complexity.*

Proof. As already outlined in Proof of Theorem 4, extremely large number of search particles can lead to non-polynomial complexity for Algorithm 4. However, this non-polynomial complexity can be avoided through the scheme where each particle is executed in a separate computing node. However, this scheme can become impractical due to infeasible resource requirement. It must be noted that, the design of PSO search mechanism ensures that, even with a single search particle, the search complexity always remains polynomial [18]. Let's indicate this complexity as C. Hence, by using sufficiently large number of particles, such that, each particle is executed in separate computing node, the complexity of Algorithm 2 can become $\approx \frac{C}{r}$. Hence, the proof immediately follows.

4. Results and discussions

4.1 Simulation setup

The proposed solution techniques for 3–slot BPP are implemented in MATLAB, and for the ease of reference, ACO based solution technique and PSO based solution technique are referred as *ACOST* and *PSOST* respectively. The 3-slot BPP has not been previously presented in the literature. Hence, *PSOST* and *ACOST* are the first techniques to provide solution to the 3-slot BPP, and due to which there are no contemporary techniques presented in the literature to perform relative simulation study along with *PSOST* and *ACOST*. However, in [1], the technique of discovering the optimal solution for classical BPP was outlined, which basically performs brute force search of all the possible solutions in-order to obtain the optimal solution. By extending this optimal solution search technique to 3-slot BPP, the optimal solution for the 3-slot BPP can be discovered, and this technique is denoted as *Optimal [1]*. It must be noted that, since, 3-slot BPP is NP-Complete, Optimal would incur Non-Polynomial complexity in its execution, and hence not suitable to be used in real-world scenarios due to extremely large execution complexity. However, Optimal [1] will be subjected to relative comparison with *PSOST* and *ACOST* in the simulation study WRT solution quality, and not WRT execution efficiency; so that, the effectiveness of *PSOST* and *ACOST* WRT solution quality can be analyzed.

Table 1.Simulation parameter settings

Simulation Parameter	Set Value
Number of bin items (n)	Varied between 10^3 and10^4
Number of search particles (*PSOST*)	10
Number of search ants (*ACOST*)	10
C_1	Varied between 0.2–0.3
C_2	Varied between 0.2–0.4
C_3	Varied between 0.3–0.6
W	0.9
r_1	0.7
r_2	0.8
D_1	0.95
D_2	0.95
c	4
Item distribution	$n_1 = n_2 = n_3$
β	0.85
ρ	0.85

The simulation parameter settings are outlined in Table 1. Search functionality of each particle for *PSOST* and each ant for *ACOST* are executed on an exclusive computing node to exploit parallelism.

4.2 Simulation results and discussions

The first experiment analyzes the performance of *PSOST* and *ACOST* when the number of bin items are varied. The utilized slot capacity setting is (C_1=0.25, C_2=0.25, C_3=0.5). The result of this experimental analysis WRT number of utilized bins is illustrated in Fig. 1, and tabulated in Table 2. Both *PSOST* and *ACOST* provide approximate solutions which are closer to the *Optimal* mainly due to the scrupulous design proposed for these two solution techniques. The second and third experiments are similar to the first experiment, only there is a change in the slot capacity setting. The result of experimental analysis for second experiment with slot capacity setting (C_1=0.2, C_2=0.4, C_3=0.4) and third experiment with slot capacity setting (C_1=0.2, C_2=0.2, C_3=0.6) are presented in Figs. 2 and 3, and tabulated in Tables 3 and 4 respectively. Again, results that are similar to the first experiment are obtained. The results illustrated in Figs. 1, 2 and 3, clearly demonstrate the superior effectiveness of *PSOST* and *ACOST*, in providing approximate solutions to 3-slot BPP which are closer to the corresponding solutions provided by the Optimal.

Table 2. C_1 = 0.25, C_2 =0.25, C_3=0.5

No of Bin Items	Optimal [1] (No of Bins Used)	PSOST (No of Bins Used)	ACOST (No of Bins Used)
1000	402	453	461
2000	1105	1204	1201
3000	1607	1702	1684
4000	2132	2305	2321
6000	3023	3321	3423
8000	3314	3821	3794
10000	4516	4743	4812

Table 3. C1 = 0.2, C2 =0.4, C3=0.4

No of Bin Items	Optimal [1] (No of Bins Used)	PSOST (No of Bins Used)	ACOST (No of Bins Used)
1000	512	576	584
2000	950	1005	1001
3000	1214	1340	1335
4000	2126	2304	2396
6000	2732	2901	2898
8000	3220	3408	3404
10000	4804	5026	5006

Table 4. C1 = 0.2, C2 =0.2, C3=0.6

No of Bin Items	Optimal [1] (No of Bins Used)	PSOST (No of Bins Used)	ACOST (No of Bins Used)
1000	431	500	505
2000	1004	1242	1228
3000	1503	1700	1645
4000	2002	2338	2290
6000	3004	3223	3400
8000	3514	3724	3820
10000	4115	4640	4700

Table 5. C1 = 0.25, C2 =0.25, C3=0.5

No of Bin Items	PSOST (Exe Time) (s)	ACOST (Exe Time) (s)
1000	73	81
2000	89	88
3000	148	156
4000	194	190
6000	280	292
8000	345	348
10000	610	600

The execution time analysis results for the first, second and third experiments are illustrated in Figures 4, 5and 6, and tabulated in Tables 5, 6 and 7 respectively. Both, *PSOST* and *ACOST* incur similar execution costs, because the number of searching elements for both these techniques is equal, and the navigation complexity of the searching elements is also similar. The results illustrated in Figures 4, 5 and 6, clearly demonstrate the noticeable execution efficiency of PSOST and ACOST, in searching for the approximate solution for the 3-slot BPP.

Table 6. C1 = 0.2, C2 =0.4, C3=0.4

No of Bin Items	PSOST (Exe Time) (s)	ACOST (Exe Time) (s)
1000	58	65
2000	71	74
3000	127	120
4000	158	156
6000	245	248
8000	340	360
10000	620	630

Table 7. C1 = 0.2, C2 =0.2, C3=0.6

No of Bin Items	PSOST (Exe Time) (s)	ACOST (Exe Time) (s)
1000	77	85
2000	88	88
3000	152	146
4000	200	183
6000	270	275
8000	343	338
10000	590	610

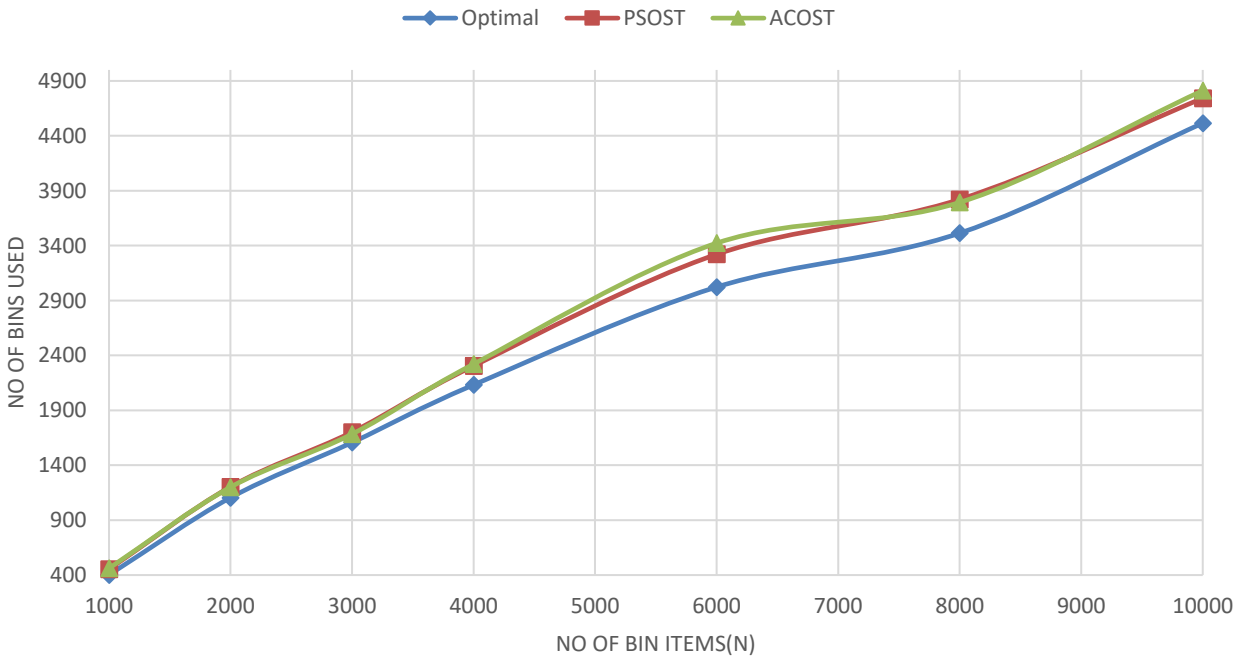

Figure. 1 C_1 = 0.25, C_2 =0.25, C_3=0.5

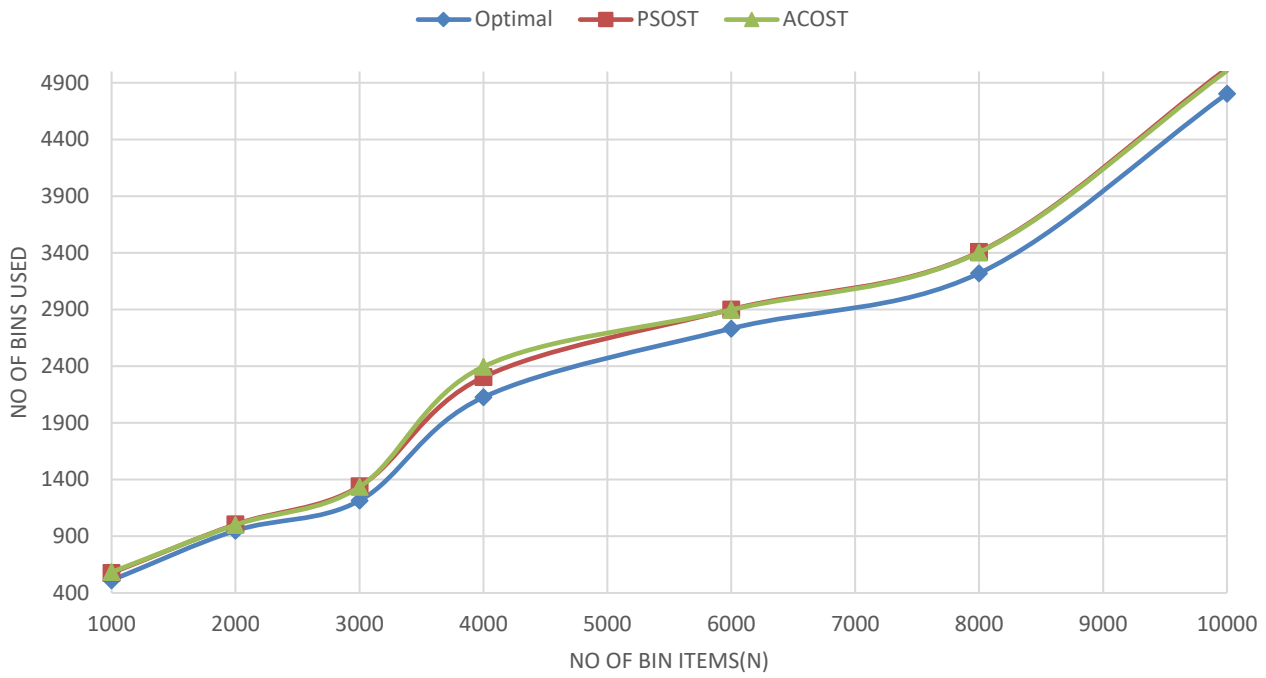

Figure.2 C_1 = 0.2, C_2 =0.4, C_3=0.4

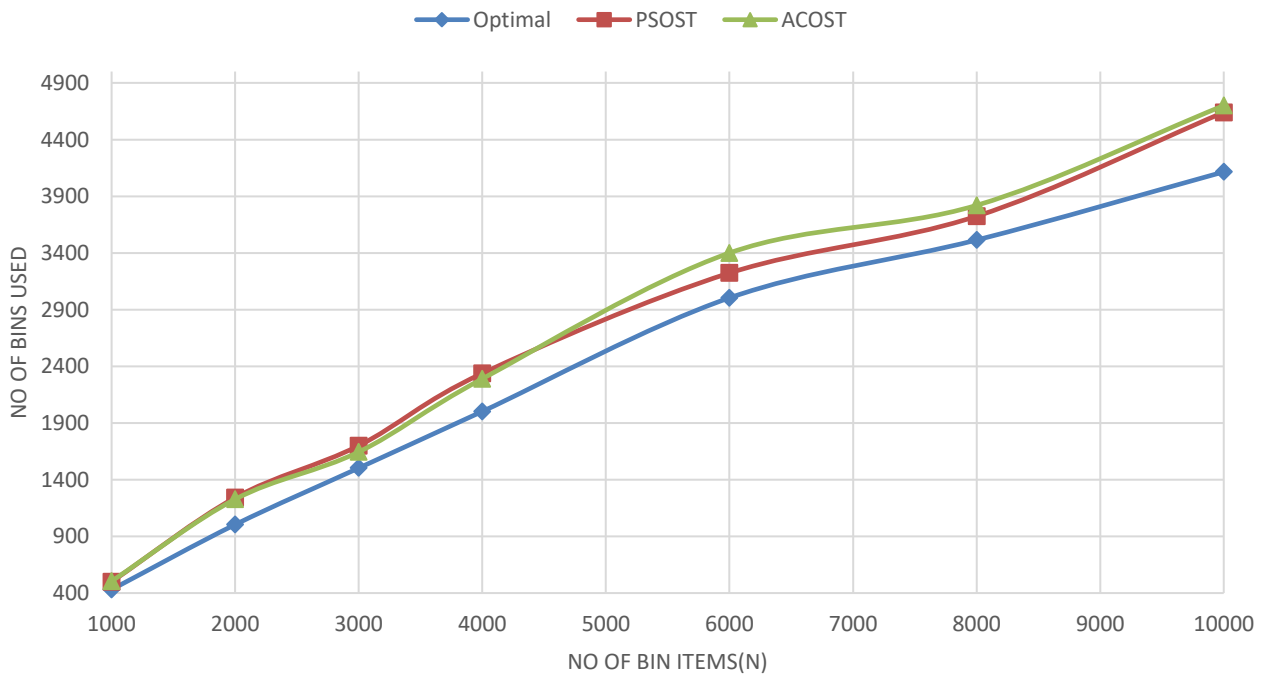

Figure.3 C_1 = 0.2, C_2 =0.2, C_3=0.6

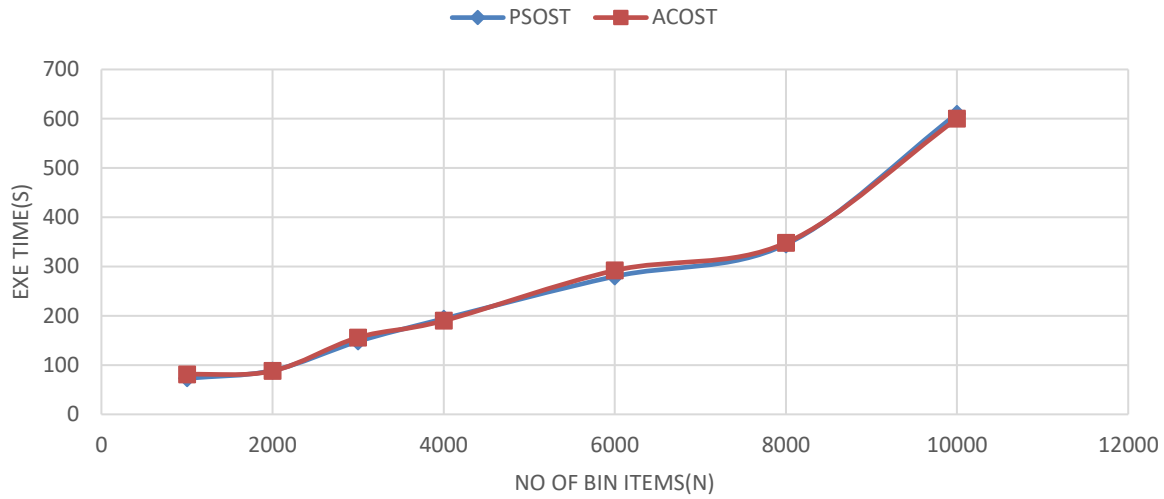

Figure.4 C_1 = 0.25, C_2 =0.25, C_3=0.5

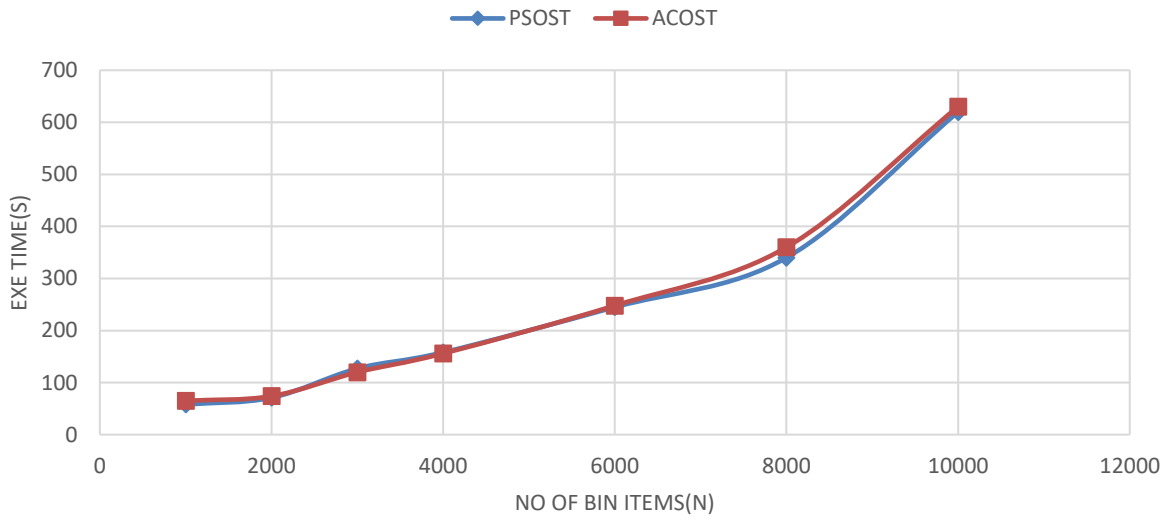

Figure.5 C_1 = 0.2, C_2 =0.4, C_3=0.4

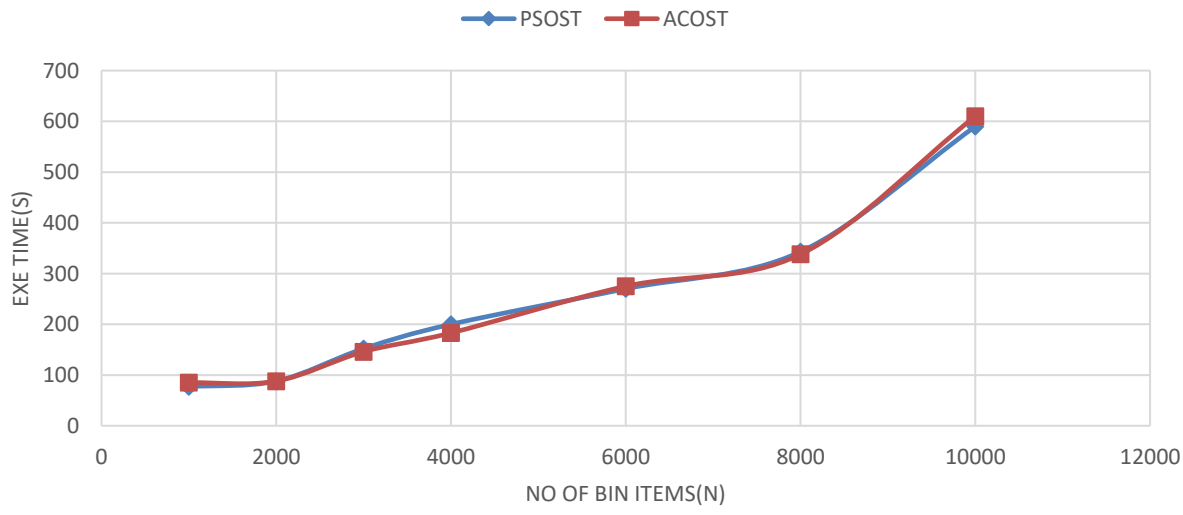

Figure.6 *C_1 = 0.2, C_2 =0.2, C_3=0.6*

5. Conclusion

In this work, VMP techniques which consider workload characteristics in DCC environment was presented. The proposed VMP techniques were modeled through 3-slot BPP framework. The 3-slot BPP was proved as NP-Complete, and hence, two meta-heuristics solutions were presented, which were based on PSO and ACO techniques. Both the proposed solutions were simulated, and their effectiveness were analyzed against the contemporary technique. Both the proposed techniques provided approximate solutions to 3-slot BPP which are closer to the optimal solution, and these demonstrated results will directly aid in improving load distribution in DCCs. Also, both these proposed techniques exhibit appreciable execution latency, and thereby, are well suited to be utilized in real-world scenarios. However, both these proposed techniques are meta-heuristic techniques, and hence, do not provide any performance guarantees. In-future, 3-slot BPP needs to be solved through approximation techniques which can provide appreciable performance guarantees.

References

[1] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A Survey on Virtual Machine Migration: Challenges, Techniques and Open Issues", *IEEE Communications Surveys and Tutorials*, pp. 1206-1239, 2018.

[2] J.S. Yang, P. Liu, and J.J. Wu, "Workload characteristics-aware virtual machine consolidation algorithms", In: *Proc. of IEEE 4th International Conference on Cloud Computing Technology and Science*, pp. 42-49, 2012.

[3] L. Hu, J. Zhao, G. Xu, Y. Ding, and J. Chu, "HMDC: Live Virtual Machine Migration based on Hybrid Memory Copy and Delta Compression", *Appl. Math*, Vol. 7, No. 2L, pp. 639–646, 2013.

[4] J. Kim, D. Chae, J. Kim, and J. Kim, "Guide-Copy: Fast and Silent Migration of Virtual Machine for Datacenters", In: *Proc. of Int. Conf. High Perform Comput. Netw. Stor. Anal.*, 2013.

[5] H. Liu and B. He, "VM buddies: Coordinating Live Migration of Multitier Applications in Cloud Environments", *IEEE Trans. Parallel Distrib. Syst.*, Vol. 26, No. 4, pp. 1192–1205, 2015.

[6] B. R. Raghunath and B. Annappa, "Virtual Machine Migration Triggering using Application Workload Prediction", *Procedia Comput. Sci.*, Vol. 54, pp. 167–176, 2015.

[7] J. Xia, D. Pang, Z. Cai, M. Xu, and G. Hu, "Reasonably Migrating Virtual Machine in NFV-Featured Networks", In: *Proc. of IEEE Int. Conf. Comput. Inf. Technol.*, pp. 361–366, 2016.

[8] U. Deshpande, and K. Keahey, "Traffic-Sensitive Live Migration of Virtual Machines", *Future Gener. Comput. Syst.*, Vol. 72, pp. 118–128, 2016.

[9] U. Deshpande, D. Chan, T.Y.Guh, J. Edouard, K. Gopalan, and N. Bila, "Agile Live Migration of Virtual Machines", In: *Proc. of IEEE Int. Parallel Distrib. Process. Symp.*, pp. 1061–1070, 2016.

[10] M. Zheng and X. Hu, "Template-based Migration between Data Centers using Distributed Hash Tables", In: *Proc. of the 12th Int. Conf. Fuzzy Syst.Knowl. Disc.*, pp. 2443–2447, 2015.

[11] J. Zhang, L. Li, and D. Wang, "Optimizing VNF Live Migration via Para-Virtualization Driver and Quick-Assist Technology", In: *Proc. of IEEE Int. Conf. Commun.*, pp. 1–6, 2017.

[12] K. Tsakalozos, V. Verroios, M. Roussopoulos, and A. Delis, "Live VM Migration under Time-Constraints in Share-Nothing IaaS-Clouds", *IEEE Trans. Parallel Distrib. Syst.*, Vol. 28, No. 8, pp. 2285–2298, 2017.

[13] T. Lu, P. Huang, M. Stuart, Y. Guo, X. He, and M. Zhang, "Successor: Proactive Cache Warm-Up of Destination Hosts in Virtual Machine Migration Contexts", In: *Proc. of the 35th Annu. IEEE Int. Conf. Comput. Commun.*, pp. 1–9, 2016.

[14] Z. Shen, Q. Jia, G.E. Sela, B. Rainero, W. Song, R. Van Renesse, and H. Weatherspoon, "Follow the Sun through the Clouds: Application Migration for Geographically Shifting Workloads", In: *Proc. of the 7th ACM Symp. Cloud Comput.*, pp. 141–154, 2016.

[15] M. Arif, A. K. Kiani, and J. Qadir, "Machine Learning based Optimized Live Virtual Machine Migration over WAN Links", *Telecommun. Syst.*, Vol. 64, No. 2, pp. 245–257, 2017.

[16] F. Zhang, X. Fu, and R. Yahyapour, "Layermover: Storage Migration of Virtual Machine across Data Centers based on Three-Layer Image Structure", In: *Proc. of IEEE 24th Int. Symp. Modeling Anal. Simulat. Comput. Telecommun. Syst.*, pp. 400–405, 2016.

[17] S.Xiao, Y. Cui, X. Wang, Z. Yang, S. Yan, and L. Yang,"Traffic-Aware Virtual Machine

Migration in Topology-Adaptive DCN", In: *Proc. of IEEE 24th Int. Conf. Netw. Protocols*, pp. 1–1, 2016.

[18] J. Liu, Y. Li, and D. Jin, "SDN-Based Live VM Migration across Data-Centers", *ACM SIGCOMM Comput. Commun. Rev.*, Vol. 44, No. 4, pp. 583–584, 2015.

[19] P. Samadi, J. Xu, and K. Bergman, "Virtual Machine Migration over Optical Circuit Switching Network in a Converged Inter-Intra Data Center Architecture", In: *Proc. of Opt. Fiber Commun. Conf. Exhibit.*, pp. 1–3, 2015.

[20] R. Xie, Y. Wen, X. Jia, and H. Xie, "Supporting Seamless Virtual Machine Migration via Named Data Networking in Cloud Data Center", *IEEE Trans. Parallel Distrib. Syst.*, Vol. 26, No. 12, pp. 3485–3497, 2015.

[21] A. Khosravi, L. L. Andrew, and R. Buyya, "Dynamic VM Placement Method for Minimizing Energy and Carbon Cost in Geographically Distributed Cloud Data Centers", *IEEE Transactions on Sustainable Computing*, Vol. 2, pp. 183 – 196, 2017.

[22] B. Perumal and A. Murugaiyan, "Virtual Machine Placement Using Hypercube Ant Colony Optimization Framework", *International Journal of Intelligent Engineering and Systems*, Vol. 10, No. 5, pp. 77-86, 2017.

[23] F. Hao, M. Kodialam, T. V. Lakshman, and S. Mukherjee, "Online Allocation of Virtual Machines in a Distributed Cloud", *IEEE/ACM Transactions on Networking*, pp.238-249, 2017.

[24] K. Kar, "Bio Inspired Computing -- A Review of Algorithms and Scope of Applications", *Expert Systems with Applications*, pp.20-32, 2016.