A Resource Aware VM Placement Strategy in Cloud Data Centers Based on Crow Search Algorithm

Anurag Satpathy[1], Sourav Kanti Addya[2], Ashok Kumar Turuk[3], Banshidhar Majhi[4] and Gadadhar Sahoo[5]
Department of Computer Science and Engineering,
[1,5]Birla Institute of Technology, Mesra, India.
[2-4]National Institute of Technology, Rourkela, India.
Email: { [1]anurag.satpathy , [2]kanti.sourav, [3]akturuk, [4]bmnitr}@gmail.com, [5]gsahoo@bitmesra.ac.in.

Abstract—Virtual machine (VM) placement in cloud data centers is a challenging task. With the increasing popularity of cloud computing across the globe, a large number of VMs are to be consolidated on a minimum number of data centers (DCs) to optimize the energy consumption and data center utilization. In this paper, we propose a resource aware approach based on a metaheuristic crow search algorithm (CSA) to consolidate a large number of VMs on minimal DCs to meet the Service level agreement (SLA) and desired quality of service (QoS) with maximum data center utilization. We propose two independent techniques, (i) greedy crow search (GCS), (ii) travelling salesman problem based hybrid crow search (TSPCS), to meet the desired objectives. A comparative study has been made from the obtained results. To evaluate the performance of proposed methods we compare them with the classical First Fit (FF) approach and the proposed methods significantly outperform the classical method.

Index Terms—Virtual Machine, Data Center, Cloud Computing, Crow Search Algorithm, Travelling Salesman Problem.

I. INTRODUCTION

Cloud computing is defined as a shared pool of virtualized resources that can be used on an on-demand and pay as you go basis. The virtualized pool of resources consists of virtual computing power (CPUs), virtual storage, virtual networks etc. These resources are scaled up or down according to the user requirements and can be accessed by a public infrastructure like Internet. This pay-as-you use computing model is the prime reason for cloud computing's growing popularity. Several MNCs like Google, Microsoft, Amazon, and IBM have built cloud platforms for enterprises and users to access cloud services [1], [2], [3].

Cloud computing uses virtualization at the servers and virtualization implies running more than one operating system instances on a single physical machine sharing the same hardware resource. These operating system instances are called as VMs. Resources at the data centers are provided to the users in the form of VMs. By using virtualization, we intend to reduce the number of servers, thereby, reducing the amount of power consumption, resource wastage, which sometimes is very high due to inefficient use of the servers [4]. Energy consumption is one of the major concerns in data center management. Reduction in energy consumption not only helps in reducing the power expenses but also helps in reducing the carbon dioxide (CO_2) and other greenhouse gas emissions. Information and communication technology Industry produces 2% of the total CO_2 emissions that is equivalent to aviation industry worldwide [5]. Moreover, data centers worldwide consume 271.8 billion KWh electricity [6]. Hence the issues of power consumption needs to be addressed.

The placement of VMs to DCs is called as the virtual machine placement problem or VMP problem. The objective of the VMP problem is to find an optimal placement that doesn't violate the SLA from the cloud user's perspective and minimizes the number of data centers used by maximizing the per-data center utilization from a CSPs perspective. This optimal placement reduces the number of data centers used by consolidating the VMs on less number of servers [7], it also reduces many operational parameters such as power, space and hardware resources.

In this paper, we propose an optimal placement algorithm which takes its motivation from the intelligent behavior of the crows [8]. The objective is to find an optimal placement of the VMs along the DCs which ensures a balanced reduction in total energy consumed and maximizes resource utilization at each DC. The under-utilized data centers can be turned off to reduce power consumption. This VMP problem is similar to an optimization problem. Intelligent as well as greedy behavior of crows have been employed to meet the desired objectives. Crows hide their extra food in certain positions of the environment and retrieve it as and when the need arises. They also follow each other to find better food sources which itself is a challenging task. If a crow finds out that another crow is following it, the latter tries to fool the initial crow by going to another position in the environment. From optimization point of view, the elements performing the search represents the crows and the search space represents the environment. Each position of the crow is a feasible solution, the quality of the food source is the objective function and the best food source of the environment is the global solution. Based on these behavior of the crows we try to find out the optimal solution to the problem in hand. *To best of our knowledge no literature exists to address VMP problem using Crow Search algorithm (CSA).*

The remainder of the paper is organized as follows, section II outlines related work. Section III describes the distribution model along with the problem definition and its constraints. Section IV outlines the results of the simulation. Finally section V concludes the paper.

978-1-5090-4559-4/17/$31.00©2017IEEE

TABLE I
NOTATIONS USED

NOTATION	DESCRIPTION
DC	Data center.
VM	Virtual machine.
M	Total number of data centers.
N	Total number of virtual machines.
K	Total number of users.
P	Total number of homogenous physical machines in a DC.
VMREQ	Represents the VM request set.
USER	Represents the user set.
USERREQ	This set contains all the VM requests of the users.
VMRESOURCE	This set contains the resource requests of all the VMs.
DCSET	This is the set of all the Data Centers.
memory	Keeps track of placement of VMs along the DCs.
cap_pm_d (j)	Capacity of the physical machine j along the d^{th} dimension.
cap_dc_d (i)	Capacity of the Data Center i along the d^{th} dimension.
cap_vmreq_d	Capacity of the VM request set along the d^{th} dimension.
$util\_dc_d$ (j)	Utilization of the Data Center j along the d^{th} dimension.

II. RELATED WORK

The problem of virtual machine placement has been studied and analyzed by a number of researchers. In [9], a greedy approach has been presented. Various greedy heuristics like Best-fit, First-Fit, Next-Fit, Worst-Fit, and Random-Fit have been applied and Best-Fit has been found to be most optimal. Maximization of the profits as well as the minimization of power consumption and SLA violations of hosted applications is studied in [4]. In [10], a genetic algorithm and fuzzy logic based technique has been used to simultaneously minimize total resource wastage, power consumption and thermal dissipation costs for VM placement. In [11], an optimal virtual machine placement (OVMP) algorithm to minimize the total cost due to buying reservation and on-demand plans of resource provisioning has been proposed.

Ant colony optimization (ACO) has been applied to solve many real world problems like travelling salesman problem (TSP) in [12]. In [13], ACO has been used to find an optimal solution to VM placement that simultaneously minimizes the resource wastage and power consumption. A hybrid queuing model for VM placement is proposed for data centers to improve the total placement time and earn more profit [14]. In [15], an approach to handle requests along different dimensions that are stochastic and time varying in nature is studied. In [16], a modified best fit decreasing (MBFD) algorithm has been proposed, which aims to reduce the number of active servers to obtain a stable host for every VM, such that the number of unnecessary migrations and the total power consumption can be reduced. In [17], a balanced approach has been proposed, to maintain the tightness of packing, as well as, the stability of the VMs by minimizing the number of physical machines and unnecessary migrations. A biogeography based optimization (BBO) technique has been proposed to optimize the VM placement that simultaneously reduces the resource wastage and the power consumption [18]. The overview of all the existing virtual machine placement strategies have been studied in [19].

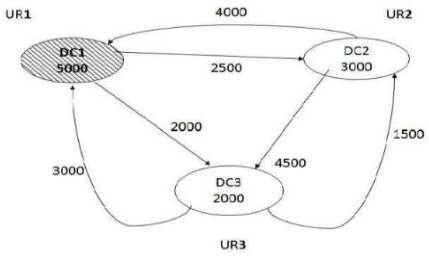

Fig. 1. **Initial Configuration GCS**

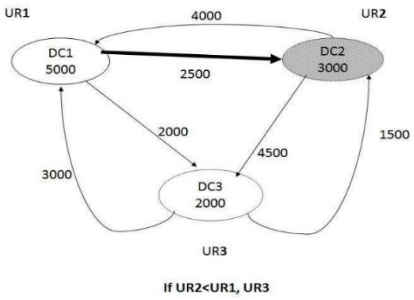

Fig. 2. **Final Configuration GCS**

III. PROPOSED MODEL

The proposed model is motivated by CSA. The crows have shown many evidences of their intelligence by remembering faces, warning each other when an unfriendly one approaches through sophisticated communication. They are also good in retaining their memory of food hiding places. They observe other birds' behavior and grab the chance of stealing their food, as and when they have the chance to.

If a crow has committed theft, then it uses his experience of thievery to protect himself from being a victim of any such theft in future. We have exploited, the intelligent behavior of the crows to suggest for two schemes of placement of VMs in DCs.

The *VMREQ* set {V M_1, V M_2, V M_3, ..., V M_N } consists of 'N' number of VMs, with each VM requesting resources along three dimensions. The VM request set is modelled in the form of crow matrix as shown:

$$crows = \begin{bmatrix} c_1^1 & c_2^1 & c_3^1 \\ c_1^2 & c_2^2 & c_3^2 \\ \vdots & \vdots & \vdots \\ c_1^N & c_2^N & c_3^N \end{bmatrix}$$

Where, an entry c^k_d in the matrix represents the peak request of V M_k along the d^{th} dimension.

There are 'K' users generating such VM requests represented by a *USER* set {U_1, U_2, U_3... U_K }. A user can request more than one VM according to his need. A *USERREQ* set is defined to keep track of requests generated by each user in the form of U_iV M_k, where the i^{th} user has made a request for the k^{th} VM. Similarly, a *VMRESOURCE* set is also introduced, where detailed requests are kept in the form of V M_a R_b for every VM, where R_b is the resource request of the VM_a. A *DCSET* comprising M homogeneous Des {DC_1, DC_2... DC_M} is taken. Each DC has 'P' homogeneous physical machines. As the crows need to store their extra food in optimal hiding places, similarly an optimal placement of VMs to DCs is proposed by maximizing per data center utilization and minimizing the number of DCs. As crows memorize their hiding places a *memory* matrix is used keep track of placements of VMs to DCs. The memory matrix is represented in the form of:

$$memory = \begin{bmatrix} m_{1,1} & m_{1,2} \cdots & m_{1,M} \\ m_{2,1} & m_{2,2} \cdots & m_{2,M} \\ \vdots & \vdots & \vdots \\ m_{N,1} & m_{N,2} \cdots & m_{N,M} \end{bmatrix}$$

Where, an entry $m_{i,j}$ = 1 if V M_i is placed in DC_j, otherwise the entry is a 0.

A global resource awareness matrix is defined, where available resource at any instant is stored. It tells us about the unused resources at any instant along all dimensions for every DC. This matrix is similar to crow matrix but rows here represents the DCs and columns represent the dimensions. An entry in the Resource Awareness matrix $RA_{i,j}$ represents the unused resource of DC_i along the j^{th} dimension. The DCs are interconnected with each other and the connection is represented in the form of a weighted directed graph given by:

$$adj = \begin{bmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,M} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,M} \\ \vdots & \vdots & & \vdots \\ d_{M,1} & d_{M,2} & \cdots & d_{M,M} \end{bmatrix}$$

Where an edge $d_{i,j}$ represents the distance between the two DCs i and j in KMs. A connectivity matrix is taken which represents the maximum range of DCs in Kms. A connectivity of 7000 Kms for a DC implies that in case of a migration the migrating VM will have to be placed in a DC within the 7000 Km range from the migrating DC.

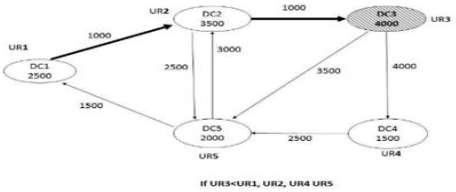

Fig. 3. Initial Configuration TSP

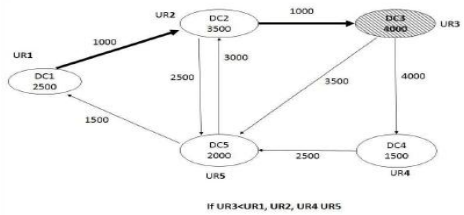

Fig. 4. Final Configuration TSP

A. Constraints

There are certain constraints that the VM placement across DCs must follow.

i. *Assignment constraint*: This constraint makes sure that the VMs are placed in such DCs where the request is met along all the three dimension.

ii. *Capacity constraint*: This constraint makes sure that the total resource requirement of the VMREQ set should be less than or equal to the total available resources taken all the DCs together along all the dimensions so that the VMs can be placed effectively.

iii. *Placement constraint:* This constraint makes sure to distribute a VM to only one data center from the *DCSET* provided, the resource requirements are met along all the dimensions.

B. Problem statement

The VMP problem finds out the best possible placement of the VMs on different data centers.

Suppose cap $\_pm_d(j)$ denotes the capacity of physical machine j along the d^{th} dimension. Then the capacity of the data centers along the d^{th} dimension can be calculated as follows:

$$cap\_dc_d(j) = \sum_{i=1}^{P} cap\_pm_d(i) \qquad (1)$$

The three dimensions under consideration are CPU, memory and disk size.

There are 'M' homogeneous data centers present in the *DCSET* and the capacity of *DCSET* along the d^{th} dimension is given by the following Equation:

$$cap\_dcset_d = \sum_{j=1}^{M} cap\_dc_d(j) \qquad (2)$$

Similar calculations are made for the capacity of the VM requests along a particular dimension as shown:

$$cap\_vmreq_d = \sum_{j=1}^{N} cap\_vm_d(j) \qquad (3)$$

The utilization can also be defined along a particular dimension for a DC as the ratio of all the VMs placed along that particular DC to the total capacity of the DC along that particular dimension.

$$\text{util\_dc}_d(j) = \frac{\sum_{i=1}^{n} cap\_vm_d\ i * memory_{i,j}}{cap\_dc_d\ j} \quad (4)$$

Mathematically, the constraints are represented as:

$$\sum_{i=1}^{n} cap\_vm\ i * memory_{i,j} <= cap\_dc_d(j)\ \forall j \in 1,2,\ldots m\ ,\ \forall d \quad (5)$$

$$\sum_{i=1}^{N} \text{cap\_vm}_d(i) <= \text{cap\_dcset}_d;\ \forall d \quad (6)$$

$$\sum_{j=1}^{M} \text{memory}_{i,j} = 1;\ \forall i \in \{1,2...N\} \quad (7)$$

$$\forall\ \text{memory}_{i,j} \in \{0,1\};\ \forall i \in \{1,2..N\};\ \forall j \in \{1,2,...M\} \quad (8)$$

Equation (5) satisfies the assignment constraint, Equation (6) represents the capacity constraint, Equation (7) and Equation (8) refer to the placement constraints respectively.

C. Greedy Crow Search Algorithm (GCS)

In GCS, the data centers are organized in the form of a complete directed graph as shown in Figure 1. The nodes of the graph represent the DCs and the edges show the connection among the DCs. The weights associated with the edges represents the distance between the two connecting DCs (KMs). Initially, the VMs are placed in the DCs randomly. Then, for every VM, a specific DC is found where, if placed, utilization is highest among all the DCs considering all dimensions.

Migration of a VM is done only if the specific DC found will have higher utilization than the DC where it is placed initially, provided it is reachable from the initial DC. Reachability of the DC, is defined by the maximum distance that a VM placed in it can move and is mentioned within the respective nodes of the graph, for example in Figure 1, 5000 mentioned inside the first DC represents the reachability of DC_1. The reachability is represented by the connectivity matrix. If the above two conditions fail, there is no movement. In Figure 2, it is seen that a VM in DC_I migrates to DC_2.

This movement occurs, because the utilization is maximum at DC_2 or in other words it has minimum unutilized resources (U R2 < U RI, U R3) and is reachable from DC_I. This approach is called a greedy crow search because we greedily place the VMs at those DCs where the utilization is at a cumulative maximum.

D. Hybrid Crow Search Algorithm (TSPCS)

A combined approach of TSP and CSA is used to solve the VM placement problem. The initial placement follows the same random allocation. Then, for every VM, TSP is implemented from the DC where it is placed initially (By Random Allocation). Continuing to visit DCs (nodes of the graph) till either the reachability (connectivity) of the DC of initial placement is exhausted or all the nodes are visited, whichever happens earlier. The selection of DCs in every movement of the TSP is made greedily.

Algorithm 1: Greedy Crow Search Algorithm

```
Input: (resourceawareness, crows, adj, connectivity, memory)
Randomly allocate DCs to N VMs based on the constraints mentioned in
  Equation 5,6,7,8.
Initialize the memory matrix as per the placement. Memory_{i,j} ← 1, if VM_i
  is placed at DC_j, 0 otherwise
Calculate the resource utilization of each DC based on Equation 4.
for i = 1 to N do
    Find a DC 'k' such that memory_{i,k} == 1, calculate the unutilized
      Resource along the DC 'k' in UR_k
    for dim = 1 to d do
        UR_k = UR_k + resourcesawareness_{k, dim};
    for j = 1 to M − {k} do
        if adj_{k,j} <= connectivity_k then
            Check for all the dimension d
            if crows_{i,d} <= resourcesawareness_{j,d} then
                for dim = 1 to d do
                    UR_j = UR_j + (resourceawareness_{j,dim} −
                      crows_{i,dim});
    Find a DC_t such that UR_t is minimum where t ={1,2...M}
    if t! = k then
        Release all the resources
        memory_{i,k} = 0;
        for dim = 1 to d do
            resourceawareness_{k,dim} = resourceawareness_{k,dim} + crows_{i,dim};
        Allocate Resources to the selected DC 't'
        memory_{i,t} = 1;
        for dim = 1 to d do
            resorceawareness_{t,dim} = resourceawareness_{t,dim} − crows_{i,dim};
```

Algorithm 2: Hybrid Crow Search Algorithm

```
Input: (resourceawareness, crows, adj, connectivity, memory)
Randomly allocate DCs to N VMs based on the constraints mentioned in
  Equation 5,6,7,8.
Initialize the memory matrix as per the placement. Memory_{i,j} ← 1, if VM_i
  is placed at DC_j, 0 otherwise
Calculate the resource utilization of each DC based on Equation 4

for i = 1 to N do
    for j = 1 to M do
        visited_j = false;
    Find a DC 'k' such that memory_{i,k} == 1
    Find the unutilized resource of DC 'k' in UR_k along all the dimension
    for dim = 1 to d do
        UR_k = UR_k + resourcesawareness_{k,dim};
    reachablity = connectivity_k;
    visited_k = true;
    for j = 1 to M do
        if reachabilty > 0 && visited_j == False then
            Find a DC_a such that Adj_{k,a} is minimum and Adj_{k,a} != 0
            visited_a = true;
            rechabilty = rechability − Adj_{k,a};
            if placement is possible then
                calculate UR_a
            set k=a;
    Place VM_i in DC_h such that UR_h is minimum.
```

TABLE II
UTILIZATION TABLE (%)

Number of VMs	DC	First Fit			Random Allocation			GCS			TSPCS		
		Dim1	Dim2	Dim3	Dim1	Dim2	Dim3	Dim1	Dim2	Dim3	Dim1	Dim2	Dim3
50	1	35.0	43.0	38.0	8.0	11.0	9.0	28.0	38.0	30.0	21.0	27.0	22.0
	2	0.0	0.0	0.0	7.0	7.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0
	3	0.0	0.0	0.0	6.0	9.0	6.0	0.0	0.0	0.0	0.0	0.0	0.0
	4	0.0	0.0	0.0	7.0	6.0	7.0	7.0	7.0	8.0	14.0	16.0	16.0
	5	0.0	0.0	0.0	7.0	9.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0
100	1	75.0	83.0	71.0	13.0	13.0	11.0	0.0	0.0	0.0	0.0	0.0	0.0
	2	0.0	0.0	0.0	19.0	24.0	19.0	75.0	83.0	71.0	15.0	17.0	16.0
	3	0.0	0.0	0.0	14.0	15.0	15.0	0.0	0.0	0.0	0.0	0.0	0.0
	4	0.0	0.0	0.0	18.0	18.0	18.0	0.0	0.0	0.0	60.0	66.0	55.0
	5	0.0	0.0	0.0	10.0	12.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0
150	1	100.0	100.0	96.0	18.0	22.0	23.0	0.0	0.0	0.0	0.0	0.0	0.0
	2	19.0	25.0	17.0	26.0	24.0	22.0	0.0	0.0	0.0	60.0	62.0	63.0
	3	0.0	0.0	0.0	30.0	30.0	23.0	97.0	100.0	94.0	37.0	38.0	31.0
	4	0.0	0.0	0.0	22.0	25.0	19.0	22.0	25.0	19.0	22.0	25.0	19.0
	5	0.0	0.0	0.0	22.0	23.0	25.0	0.0	0.0	0.0	0.0	0.0	0.0
200	1	100.0	100.0	93.0	36.0	38.0	41.0	99.0	100.0	95.0	2.0	2.0	2.0
	2	64.0	69.0	68.0	33.0	32.0	28.0	0.0	0.0	0.0	0.0	0.0	0.0
	3	0.0	0.0	0.0	30.0	37.0	34.0	0.0	0.0	0.0	63.0	67.0	65.0
	4	0.0	0.0	0.0	40.0	38.0	33.0	65.0	69.0	66.0	99.0	100.0	93.0
	5	0.0	0.0	0.0	24.0	24.0	25.0	0.0	0.0	0.0	0.0	0.0	0.0
300	1	100.0	95.0	100.0	49.0	48.0	52.0	100.0	92.0	100.0	16.0	14.0	16.0
	2	100.0	85.0	100.0	47.0	37.0	45.0	0.0	0.0	0.0	25.0	20.0	23.0
	3	36.0	37.0	40.0	41.0	40.0	41.0	0.0	0.0	0.0	0.0	0.0	0.0
	4	0.0	0.0	0.0	52.0	49.0	53.0	100.0	95.0	100.0	99.0	96.0	100.0
	5	0.0	0.0	0.0	47.0	44.0	49.0	36.0	36.0	40.0	95.0	88.0	100.0
400	1	100.0	100.0	99.0	61.0	54.0	62.0	0.0	0.0	0.0	35.0	25.0	32.0
	2	82.0	94.0	100.0	61.0	55.0	64.0	100.0	94.0	99.0	0.0	0.0	0.0
	3	100.0	100.0	100.0	61.0	73.0	72.0	84.0	100.0	99.0	94.0	94.0	100.0
	4	28.0	23.0	28.0	68.0	76.0	66.0	100.0	100.0	100.0	91.0	100.0	95.0
	5	0.0	0.0	0.0	58.0	60.0	66.0	25.0	24.0	28.0	89.0	93.0	100.0
500	1	100.0	93.0	100.0	74.0	85.0	79.0	71.0	77.0	72.0	0.0	0.0	0.0
	2	100.0	93.0	93.0	85.0	71.0	81.0	100.0	91.0	97.0	100.0	87.0	99.0
	3	92.0	98.0	100.0	78.0	70.0	81.0	95.0	87.0	100.0	100.0	99.0	100.0
	4	98.0	96.0	98.0	88.0	87.0	84.0	100.0	100.0	98.0	100.0	99.0	98.0
	5	0.0	0.0	0.0	64.0	65.0	66.0	23.0	25.0	25.0	90.0	94.0	95.0

Here, we are greedy about the distance as the DC placed at a minimum distance adjacent to the current DC is selected as the next probable DC for placement. By visiting a DC which is at a lesser distance, we make sure that we traverse as many DCs as possible. By traversing many DCs we improve our chances of getting a better placement. Finally, the VM is moved to the DC having maximum utilization, from among the visited DCs during the traversal. Otherwise, there is no movement. The initial configuration for the TSPCS in depicted in Figure 3, similar to GCS, the only difference is all the DCs are not connected to each other by a direct edge. A VM placed at DC_1 migrates to DC_3 as shown in Figure 4. The path traversed is represented by darkened arrows.

IV. SIMULATION RESULTS AND DISCUSSION

The proposed models are implemented using CSA and obtained results are compared with the existing classical First Fit strategy. An in-house simulation is done using JAVA on a desktop system with Intel (R) Core (TM) i3-380M processor with 2.53 GHz and 2 GB memory. In the simulation, a configuration of 5 homogeneous DCs are taken where each DC has 100 homogeneous physical machines. The PMs have resources along three dimensions namely, CPU, memory and disk size. Every PM is assumed to have a maximum of 90% utilization capacity along all three dimensions. The VM requests are taken in order of 50 to 500 with an interval of 50 for the first four observations and 100 for the next three observations. The VM requests, weighted adjacency matrix and connectivity (reachability) are generated randomly.

The Table II represents the resource utilization of the 5 DCs involved in the simulation. In the First-Fit allocation, DCs

encountered first (DCs having lower index) are filled up first. For the same random allocation GCS and TSPCS are performed and the utilization of DCs for these two approaches are shown in the third and fourth column respectively.

For the GCS the DCs that have higher utilization that is more VMs are placed in it, have higher chances of VMs migrating to them provided they are reachable from the other DCs. Otherwise that DC is selected for migration which has the next highest utilization. By making such movements we try to fill up those DCs that have higher utilization but still have unused resources.

TSPCS is similar to GCS but the difference lies in the connection among the DCs, a complete directed graph is used in GCS which implies that every DC has a direct connection to every other DC, which is not practically feasible. Taking the same directed graph into consideration, we randomly delete edges from it without compromising with the connectivity of the graph (i.e. there is a path from every node to every other node) which represents a more practical scenario of connection among the DCs.

Results show that the TSPCS is faster than both the classical First Fit approach and the GCS. Not only does TSPCS take lesser time to execute than the other two algorithms as shown in Table III but also a comparative study in Table IV shows TSPCS takes less number of migrations than GCS.

TABLE III
TIME CONSUMED BY ARIOUS APPROACHES

No of VMs	Time Consumed (in milliseconds)		
	First Fit	GCS	TSPCS
50	7	13	5
100	10	14	7
150	10	17	9
200	11	20	7
300	14	24	10
400	14	28	13
500	34	31	19

TABLE IV
NU M B E R OF MI G R AT I O N S

No of VMs	Total number of migrations	
	GCS	TSPCS
50	39	29
100	48	48
150	106	82
200	131	115
300	185	159
400	164	39
500	107	69

V. CONCLUSION

In this paper, we addressed the issue of VM placement. With increasing use of cloud, more number of VMs are to be consolidated on less number of DCs through effective utilization of DCs. We proposed two different algorithms which use the clever behavior of the crows to solve the placement problem. The parameters to be optimized have been carefully studied and it has been found that TSPCS technique performs the best. VM placement is associated with migration and migration is a costly process and involves many more parameters including distance. These issues will be addressed in the extended versions of this paper.

REFERENCES

[1] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, Aug 2009, pp. 44–51.

[2] W. Voorsluys, J. Broberg, and R. Buyya, "Introduction to cloud computing," *Cloud computing: Principles and paradigms*, pp. 1–44, 2011.

[3] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of- the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010. [Online].

[4] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, Dec 2010, pp. 179–188.

[5] A. Khosravi, S. K. Garg, and R. Buyya, *Energy and Carbon-Efficient Placement of Virtual Machines in Distributed Cloud Data Centers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 317–328.

[6] K. Bilal, S. U. Khan, and A. Y. Zomaya, "Green data center networks: Challenges and opportunities," in *Frontiers of Information Technology (FIT), 2013 11th International Conference on*, Dec 2013, pp. 229–234.

[7] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *IEEE Transactions on Services Computing*, vol. 3, no. 4, pp. 266–278, Oct 2010.

[8] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm," *Computers and Structures*, vol. 169, pp. 1 – 12, 2016.

[9] A. K. Paul, S. K. Addya, B. Sahoo, and A. K. Turuk, "Application of greedy algorithms to virtual machine distribution across data centers," in *2014 Annual IEEE India Conference (INDICON)*, Dec 2014, pp. 1–6.

[10] H. N. Van, F. D. Tran, and J. M. Menaud, "Performance and power management for cloud infrastructures," in *2010 IEEE 3rd International Conference on Cloud Computing*, July 2010, pp. 329–336.

[11] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, Dec 2009, pp. 103–110.

[12] C. Blum, "Ant colony optimization: Introduction and recent trends," *Physics of Life Reviews*, vol. 2, no. 4, pp. 353 – 373, 2005.

[13] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230 – 1242, 2013.

[14] S. K. Addya, A. K. Turuk, B. Sahoo, and M. Sarkar, "A hybrid queuing model for virtual machine placement in cloud data center," in 2015 IEEE International Conference on Advanced Networks and Telecommunication Systems (ANTS), Dec 2015, pp. 1-3.

[15] H. Jin, D. Pan, J. Xu, and N. Pissinou, "Efficient vm placement with multiple deterministic and stochastic resources in data centers," in Global Communications Conference (GLOBECOM), 2012 IEEE, Dec 2012, pp. 2505-2510.

[16] A. Kaur and M. Kalra, "Energy optimized vm placement in cloud environment," in 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence), Jan 2016, pp. 141–145.

[17] M. Mishra and U. Bellur, "Whither tightness of packing? The case for stable vm placement," IEEE Transactions on Cloud Computing, vol. PP, no. 99, pp. 1–1, 2015.

[18] Q. Zheng, R. Li, X. Li, N. Shah, J. Zhang, F. Tian, K.-M. Chao, and J. Li, "Virtual machine consolidated placement based on multi-objective biogeography-based optimization," Future Generation Computer Systems, vol. 54, pp. 95 – 122, 2016. [Online]. Available:http://www.sciencedirect.com/science/article/pii/S0167739X15000564

[19] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," Journal of Network and Computer Applications, vol. 66, pp. 106 – 127, 2016.