BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **17**, No 4
Sofia • 2017 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2017-0041

Fuzzy Bio-Inspired Hybrid Techniques for Server Consolidation and Virtual Machine Placement in Cloud Environment

Boominathan Perumal[1], *Aramudhan M.*[2], *Saravanaguru Ra K.*[3]

[1]*School of Computer Science and Engineering, VIT University, Vellore, India*
[2]*Perunthalaivar Kamarajar Institute of Engineering and Technology, Puducherry, India*
[3]*School of Computer Science and Engineering, VIT University, Vellore, India*
E-mails: boomi051281@gmail.com aramudhan1973@yahoo.com saravanank@vit.ac.in

Abstract: *Cloud computing technology has transformed the information and communication technology industry by authorizing on-demand resource delivery to the cloud users. Datacenters are the major resource storage places from where the resources are disseminated to the requesters. When several requests are received by datacenters, the available resources are to be handled in an optimized way; otherwise the datacenters suffer from resource wastage. Virtualization is the technology that helps the cloud providers to handle several requests in an optimized way. In this regard, virtual machine placement, i.e., the process of mapping virtual machines to physical machines is considered to be the major research issue. In this paper, we propose to apply fuzzy hybrid bio-inspired meta-heuristic techniques for solving the virtual machine placement problem. The cuckoo search technique is hybridized with the fuzzy ant colony optimization and fuzzy firefly colony optimization technique. The experimental results obtained show competing performance of the proposed algorithms.*

Keywords: *cloud computing, virtual machine placement, server consolidation, power consumption, resource wastage, cuckoo, ant colony system, firefly colony.*

1. Introduction

Virtualization is the core technology that makes cloud computing possible. Cloud computing mainly focusses on delivering services over the internet in different layers like Infrastructure as a Service (IaaS), Software as a Service (SaaS) and Platform as a Service (PaaS) [1]. For a cloud user, the cloud provider provides the resources based on pay per use concept. Datacenters are the most power consuming storage areas. Even a fraction of power reduction in datacenters benefits the cloud providers to great extent.

Virtualization is the core concept to achieve server consolidation which mainly aims to minimize the number of physical servers used for virtual machine placement. The process of mapping of a set of Virtual Machines to a set of Physical Machines (VM-PM mapping) is called virtual machine placement and the process of finding optimal placement solution is considered here as virtual machine placement problem. In literature, Gupta et al. [2] introduced a two stage heuristic approach to solve server consolidation problem focusing on bin-item incompatibility constraints. Some of the heuristic techniques widely used for consolidation problem are the First Fit Decreasing (FFD) [3], Best Fit [4], Best Fit Decreasing [5] and other heuristics [6, 7]. To apply Heuristic techniques simple sorting mechanisms are given by Maruyama, Chang and Tang [8] which can be used.

Few researchers have taken efforts to convert the multidimensional sizes considered as vectors into sizes of scalar type. Panigrahy et al. [9] proposed a novel method of geometric heuristics and have presented a report on their findings related to different combinations of vectors to generate the scalar (size). Wood et al. [3] proposed a Sandpiper system that automates the task of monitoring and detecting hotspots so as to facilitate the initiation of VM migrations. Mishra and Sahoo [4] introduced a novel vector based approach to overcome the anomalies in the existing VM placement technologies. The other works of VM placement following bin packing heuristic are given by Jung et al. [10], Li et al. [6], etc.

Non-deterministic approaches like genetic algorithm, simulated annealing, ant colony optimization, particle swarm optimization, etc., are also widely used for VM placement. Falkenauer [11] proposed an enhanced approach of genetic algorithm called grouping genetic algorithm to handle the server consolidation problem. As another variant, Brugger et al. [12] proposed an ACO metaheuristic that had better performance than genetic algorithm for large problem instances. Rohlfshagen and Bullinaria [13] proposed another variant of GGA called exon shuffling genetic algorithm. A group genetic algorithm is proposed by Agarwal, Bose and Sundarrajan [14] to solve server consolidation problem as a vector packing problem with conflicts. As a further improvement Wilcox, Mc Nabb and Seppi [15] proposed two different solutions encoding schemes and also its fitness function were designed in such a way that it can solve Multi-Capacity Bin Packing Problems. Feller, Rilling and Morin [16] used another version of ACO to address VM consolidation and have shown better results than FFD. Perumal and Murugaiyan [17] proposed fuzzy firefly algorithm for server consolidation and virtual machine placement problem. Wu, Tang and Fraser [18] applied simulated annealing technique for VM placement.

In addition, Luke [19] designed new mutation and cross over operations for steady state genetic algorithm for VM placement. Gao et al. [20] proposed Pareto-dominance approach to determine best solution in multi objective context and results were compared with Xu and Fortes [21]. Suseela and Jeyakrishnan [22] proposed a multi-objective hybrid ACO-PSO optimization algorithm for minimizing resource wastage, power consumption for load balancing in physical servers. Zhao et al. [23] proposed an improved particle swarm optimization with simulated annealing for energy saving during placement and live migration. Nguyen, Le and Nguyen [24] proposed energy efficient resource allocation strategies for providing virtual services based on heterogeneous shared hosting platforms. Sanyasi and Bhagat [25] emphasized on cloud optimization and security gaps. In this paper we propose hybrid algorithms for solving server consolidation and virtual machine placement problem.

The rest of the paper is organized as follows. In Section 2, we formulate the server consolidation problem. In Section 3, we give the preliminaries of ant colony system, firefly colony system and cuckoo search for server consolidation problem. In Section 4 we present the hybrid approaches and the computational experimental study conducted. In Section 5, we formulate the VM placement problem and propose the multi-objective hybrid techniques. In Section 6 we give the concluding remarks followed by references.

2. Server consolidation problem

The server consolidation problem is addressed in the following subsections.

2.1. Problem formulation

Server consolidation aims to minimize the number of servers required for placing Virtual Machines (VMs). Considering there are n VMs ($i \in I$) to be placed in m Physical Machines (PMs) ($j \in J$), assume no virtual machine requires more capacity than that can be provided by a single server. Let $\left[R_{\text{cpu}}^{i} \quad R_{\text{mem}}^{i}\right]$ be the CPU demand and memory demand vector for each VM. Let also $\left[\text{TC}_{\text{cpu}}^{j} \quad \text{TC}_{\text{mem}}^{j}\right]$ be the processing unit capacity and memory capacity associated with each physical machine. A threshold of 90 % is set for $\left[\text{TC}_{\text{cpu}}^{j} \quad \text{TC}_{\text{mem}}^{j}\right]$ to avoid physical machine resource utilization reaching 100% as it may lead to severe performance degradation when it is fully utilized [20]. We use two decision variables allocation matrix and binary variable defined as follows: allocation matrix $\text{alloc}_{i,j} \in \{0, 1\}$ is set to 1 if vm_i is allocated to the server j, otherwise it is set to zero. A binary variable $y_j \in \{0, 1\}$ indicates whether a server is in use or not.

2.2. Resource wastage modelling

The potential cost of wasted resource (Resource Wastage – RW) is computed by equation [20]

$$\text{RW}_j = \frac{\left|L_j^{\text{cpu}} - L_j^{\text{mem}}\right| + \varepsilon}{\left|U_j^{\text{cpu}} + U_j^{\text{mem}}\right|}, \tag{1}$$

where RW_j represents the resource wastage of the j-th server; U_j^{mem} and U_j^{cpu} denotes the normalized memory and CPU resource usage; L_j^{mem} and L_j^{cpu} denote the normalized remaining memory and CPU usage, respectively. A small positive constant value of 0.0001 is set for ε to avoid resource wastage of a server returning as zero.

2.3. Objective function

The objective is to minimize the number of servers used without any violation of capacity constraints. The minimization function is formulated as follows [27]:

$$\text{Minimize } \sum_{j=1}^{m} y_j, \tag{2}$$

subject to constraints:

$$\sum_{j=1}^{m} \text{alloc}_{ij} = 1 \quad \forall i \in I, \tag{3}$$

$$\sum_{i=1}^{n} R_{\text{cpu}}^{i} \text{alloc}_{ij} \le \text{TC}_{\text{cpu}}^{j} y_j \quad \forall j \in J, \tag{4}$$

$$\sum_{i=1}^{n} R_{\text{mem}}^{i} \text{alloc}_{ij} \le \text{TC}_{\text{mem}}^{j} y_j \quad \forall j \in J, \tag{5}$$

$$y_j, \text{alloc}_{ij} \in \{0,1\} \quad \forall i \in I,\ j \in J. \tag{6}$$

Constraint (3) assigns a VM i to only one of the servers. The capacity constraint of the servers is specified in constraint (4) and (5). The domain of the binary decision variables used are given in (6).

3. Preliminaries of ant colony, firefly and cuckoo for placement problem

3.1. Ant colony system

Ant Colony Optimization (ACO) [28] is inspired by the ants' searching behavior and their inherent ability to find the shortest path between their nest and the food source. Ants choose a particular path to follow by making a probabilistic decision biased by the amount of pheromone deposited. Other important information is heuristic information. It helps in guiding the ants to construct good solution. The heuristic information of assigning VM i to server j is

$$\eta_{ij} = \frac{\left|R_{\text{cpu}}^{i} + R_{\text{mem}}^{i}\right|}{\left|L_j^{\text{cpu}} - L_j^{\text{mem}}\right| + \varepsilon}. \tag{7}$$

The pheromone trail τ_{ij} is

$$\tau_{ij} = \begin{cases} \dfrac{\sum_{u \in \Omega_k(j)} \tau_{ui}}{|\Omega_k(j)|} & \text{if } \Omega_k(j) - \{i\} \ne 0, \\ 1 & \text{otherwise.} \end{cases} \tag{8}$$

The solution construction by an ant k is based on a pseudo random proportional rule given in the next equation:

(9) $$i=\begin{cases}\arg\max_{u\in\Omega_k(j)}\{\alpha\times\tau_{uj}+(1-\alpha)\times\eta_{uj}\}, & q\le q_0,\\ \text{explore } e & \text{otherwise,}\end{cases}$$

where q is a random variable uniformly distributed in [0, 1] and q_0 is a fixed variable having values between 0 and 1. If q is less than or equal to q_0, then the process is called exploitation otherwise exploration of new mappings is preferred. e is a random variable selected according to Roulette Wheel selection method using the random-proportional rule probability distribution [29]:

(10) $$P_{i,j}^k=\begin{cases}\dfrac{\alpha\times\tau_{ij}+(1-\alpha)\times\eta_{ij}}{\sum_{u\in\Omega_k(j)}(\alpha\times\tau_{ij}+(1-\alpha)\times\eta_{ij})}, & i\in\Omega_k(j),\end{cases}$$

(11) $$\Omega_k(j)=\left\{\begin{array}{l} i\in\{1,\dots,n\}\mid\left(\sum\limits_{u=1}^{m}\text{alloc}_{iu}=0\right)\wedge\left(\left(\sum\limits_{u=1}^{n}\left(\text{alloc}_{uj}\times R_{\text{cpu}}^{u}\right)+R_{\text{cpu}}^{i}\right)\le \text{TC}_{\text{cpu}}^{j}\right)\wedge\\ \left(\left(\sum\limits_{u=1}^{n}\left(\text{alloc}_{uj}\times R_{\text{mem}}^{u}\right)+R_{\text{mem}}^{i}\right)\le \text{TC}_{\text{mem}}^{j}\right).\end{array}\right\}$$

The local update of pheromone value is computed using

(12) $$\tau_{ij}=(1-\rho_l)\tau_{ij}(t-1)+\rho_l.\tau_0,$$

where $\rho_l\in\{0,1\}$ is the pheromone decay coefficient and τ_0 is the initial value of the pheromone.

To evaluate the fitness of the solutions obtained, we adapt the cost function given by Sait, Bala and El-Maleh [27] which is based on the fitness of a VM that got packed:

(13) $$\frac{R_i^{\text{cpu}}+R_i^{\text{mem}}}{\left(\text{TC}_{\text{cpu}}^{i}-\sum\limits_{k=1,k\ne i}^{n}R_k^{\text{cpu}}\right)+\left(\text{TC}_{\text{mem}}^{i}-\sum\limits_{k=1,k\ne i}^{n}R_k^{\text{mem}}\right)}.$$

The global pheromone update based on best solution is

(14) $$\tau_{ij}(t)=(1-\rho_g)\tau_{ij}(t-1)+\rho\Delta\tau_{ij}^{\text{best}},$$

where $\rho_g\in\{0,1\}$ is the evaporation rate, and

$$\Delta\tau_{ij}^{\text{best}}=\begin{cases}\mathit{ff}_{\text{SC}}(S^{\text{gb}}) \text{ if VM } i \text{ is placed in server } j,\\ 0 \quad \text{otherwise,}\end{cases}$$

where ff_{SC} is the fitness of the solution found by computing the average fitness of placed VMs which is actually computed based on VM fitness equation given in (13).

3.2. Firefly colony optimization algorithm

The Firefly Colony Optimization (FCO) algorithm is a swarm intelligence based metaheuristic approach which is based on ACO technique. FCO is inspired by the flashing behaviour of fireflies. The solution is constructed is using firefly state transition rule [17]:

(15) $$i=\begin{cases}\arg\max_{u\in\Omega_k(j)}\left\{\beta_{uj}{*}e^{-\gamma \mathrm{RW}_{uj}^{m}}\right\}, & q\le q_0,\\ \text{explore } e \quad \text{otherwise.}\end{cases}$$

Equation (15) can be rewritten as given in [17]:

(16) $$i=\begin{cases}\arg\max_{u\in\Omega_k(j)}\left\{\beta_{uj}{*}\eta_{uj}\right\}, & q\le q_0,\\ \text{explore } e \quad \text{otherwise,}\end{cases}$$

where the heuristic information η_{ij} is determined using the term $\frac{1}{\gamma \mathrm{RW}_{uj}^{m}}$, γ is the absorption coefficient of the light and it is initialized to 1. In Equation (16), if q is less than q_0 then a VM u with higher attractiveness is chosen from a set of eligible virtual machines. If q is greater than q_0 then the cumulative sum of the attractiveness of all eligible VMs are obtained and then the VM having the higher attractiveness than a generated random number is chosen to be the next VM for placement. The cumulative sum of the attractiveness is obtained by:

(17) $$\text{Attractivenessvector}=\text{cumsum}\left(\beta_{ij}^{k}\right),\ i\in\Omega_k(j),$$

(18) $$\beta_{ij}^{k}(t)=\begin{cases}\beta_{ij}*\eta_{ij}, & i\in\Omega_k(j),\\ 0 \quad \text{otherwise,}\end{cases}$$

(19) $$\beta_{ij}=\begin{cases}\dfrac{\sum_{u\in\Omega_k(j)}\beta_{ui}}{|\Omega_k(j)|} & \text{if } \Omega_k(j)-\{i\}\ne 0,\\ 1 \quad \text{otherwise.}\end{cases}$$

The local update of attractiveness is

(20) $$\beta_{ij}(t)=\alpha\left(\text{rand}-\frac{1}{2}\right)\beta_{ij}(t-1)+\beta_0,$$

where α is the attractiveness decay parameter, the initial value for β_0 is calculated using $\beta_0 = 1/[n\mathrm{RW}(S_0)]$, The global update of attractiveness is

(21) $$\beta_{ij}(t)=\alpha\left(\text{rand}-\frac{1}{2}\right)\beta_{ij}(t-1)+\Delta\beta_{ij}^{\text{best}},$$

where $\Delta\beta_{ij}(t)=\begin{cases}ff_{\mathrm{SC}}(S^{\mathrm{gb}}), & \text{if VM } i \text{ is placed in server } j,\\ 0 \quad \text{otherwise.}\end{cases}$

3.3. Cuckoo search

The Cuckoo Search Optimization (CSO) algorithm [26] is a metaheuristic approach inspired by the aggressive reproduction strategy of the cuckoos.

4. Proposed bioinspired hybrid optimization

The following sections present the proposed hybrid methods.

4.1. Fuzzy ACS-Cuckoo optimization

The fuzzy rules generated to decide next VM i for the current server j are as follows:

If β_{ij} is *low* and η_{ij} is *low* then the efficacy e_{ij} of choosing VM i is *very very low*.

If β_{ij} is *medium* and η_{ij} is *low* then the efficacy e_{ij} of choosing VM i is *very low*.

If β_{ij} is *high* and η_{ij} is *low* then the efficacy e_{ij} of choosing VM i is *low*.

If β_{ij} is *low* and η_{ij} is *medium* then the efficacy e_{ij} of choosing VM i is *low*.

If β_{ij} is *medium* and η_{ij} is *medium* then the efficacy e_{ij} of choosing VM i is *medium*.

If β_{ij} is *high* and η_{ij} is *medium* then the efficacy e_{ij} of choosing VM i is *high*.

If β_{ij} is *low* and η_{ij} is *high* then the efficacy e_{ij} of choosing VM i is *high*.

If β_{ij} is *medium* and η_{ij} is *high* then the efficacy e_{ij} of choosing VM i is *very high*.

If β_{ij} is *high* and η_{ij} is *high* then the efficacy e_{ij} of choosing VM i is *very very high*.

This method uses the minimum operation for fuzzy implication and max-min operator for the composition. Finally we obtain e_{ij}^{k} as the maximum efficacy for each virtual machine i. We use the following mechanism given in (21) to decide VM i for server j:

$$I=\begin{cases}\text{Fuzzy strategy, } q\le q_0 \text{ (exploitation)},\\ \text{Fuzzy probable strategy, } q>q_0 \text{ (exploration)}.\end{cases} \tag{22}$$

The output of each strategy is a crisp number specifying the next virtual machine to place in the server.

Fuzzy Strategy

The fuzzy strategy is introduced to implement the exploitation process:

$$\left[e_{u^{*}j}\right]=\sup_{u\in\Omega_k(j)}\left\{e_{uj}\right\}, \tag{23}$$

where $i = u^{*}$.

Fuzzy Probable Strategy (FPS)

The fuzzy probable strategy is introduced to implement the exploration process of fireflies:

$$\tilde{\beta}_{ij}^{k}=\frac{e_{ij}^{k}}{\sum_{u\in\Omega_k(j)} e_{uj}^{k}}. \tag{24}$$

Algorithm

The pseudocode of the fuzzy ACS-Cuckoo technique is given in Tables 1-3. The generation of instances is as given by G a o et al. [20].

Table 1. Fuzzy ACS-Cuckoo

```
%%%%Initialization
Initialize Number of Physical Machines (PMs), Number of Virtual Machines (VMs)
Set  List of physical machines and their current usage
Set  List of virtual machines and their requirements demand
Set maxIterations
Initialize the pheromone matrix $\tau_{ij}$, Number of Ants (NA)
Generate server consolidation problem instances using procedure given in Table 2
  Repeat
  Step 1. For each ant $k$ = 1: Number of Ants (NA) do
  Step 2.       S=constructSolution( )
  Step 3.    Update pheromones for the local best solutions using local update rule given in  (12)
  Step 4.  End For
  Step 5. Determine the objective function values using (2)
  Step 6. Apply cuckoo search procedure given in Table 3 to obtain new optimal solutions
  Step 7. Update Pheromones using  global updating rule given in  (14)
  Step 8. Until the maximum number of iterations is reached
  Step 9. Output global best solution and its fitness value

%%% constructSolution( )
 Step 1. Repeat
 Step 2.  Release a new server from the set of physical servers
 Step 3.   Repeat
 Step 4.  For each remaining VM that qualify for  inclusion in the current server
 Step 5.  Calculate the heuristic information using (7)
 Step 6.  Calculate the probability using (10)
 Step 7.  End For
 Step 8. Choose a VM for placement using Fuzzy state transition rule using (23)-(24)
 Step 9. Until no remaining VM fits in the server anymore
 Step 10. Until all VMs are assigned
```

Table 2. Cuckoo Search Algorithm [27]

```
 Step 1. Calculate the fitness of the solutions using fuzzy fitness procedure
 Step 2. Rank and Partition the solutions into top and bottom nests
 Step 3. For each bottom nests do
 Step 4. Rank and Partition the servers into top and bottom group
 Step 5. Delete $x$ number of servers from the bottom group
 Step 6. Sort the deleted VMs in bottom group servers using one of the sorting methods
 Step 7. Reinsert the VMs into the nest using First Fit decreasing heuristic
 Step 8. Partition the resulting solutions into top and bottom groups
 Step 9. End For
 Step 10. For each top nests do
 Step 11.    Rank and Partition the servers in the nest into top and bottom group
 Step 12. Delete 25%  number of servers (bottom group)
 Step 13. Sort the deleted VMs in bottom group servers using one of the sorting methods
 Step 14. Reinsert the VMs into the nest using First Fit decreasing heuristic and store the nest as
 best solution
 Step 15. Compare the solution with any of the randomly chosen existing solution
 Step 16. If fitness function of new solution is better than existing solution, then remove existing
 solution and place new solution
 Step 17. End If
 Step 18. End For
 Step 19. Store the best nest seen so far
 Step 20. End Repeat
```

Table 3. First fit heuristic

```
Step 1. Sort the VMs demand requirements in decreasing order using any of the multidimensional
sorting methods given in Maruyuma [8].
Step 2. For VM=1: number of virtual machines do
Step 3.   For j=1: number of servers do
Step 4.     If demand(VM)<=capacity(j) then
Step 5.       place virtual machine VM to server j
Step 6.       reduce the server capacity
Step 7.       break
Step 8.     End If
Step 9.   End for
Step 10.    If a virtual machine VM did not fit any of the available servers then choose a new
server and place it
Step 11.    End If
End For
```

4.2. Fuzzy Firefly-Cuckoo Optimization Algorithm

Here we apply cuckoo search for the solutions obtained using firefly colony approach and the procedure for applying cuckoo search is same as described in Section 4.1, Table 2. The preliminaries of firefly colony for VM placement are given in Section 3.2. Firefly colony solution construction process is given in (16) is replaced with the same fuzzy concepts given in (22)-(24) for fuzzy firefly colony approach, local pheromone update in (20) and global pheromone update in (21).

4.3. Computational experimental study

The VM requirement instances are generated using the procedure given in [20]. To support the worst VM placement scenario, the number of servers is set to the number of VMs, in which only one VM is assigned per server. The experimental results shown are for 300 VMs. For the proposed ACO-Cuckoo search algorithm, the parameters are set as follows:

q_0=0.8, NA=10, M=100, α=0.45, $\rho_l = \rho_g = 0.35$, $\tau_{pj} = \tau_{mj}$ =90%, η=0.0001.

We performed 20 runs and each run is repeated for 100 iterations and the final results reported are average of 20 runs. In the results table LB is the theoretical lower bound on the number of servers that can be used for placement which is

$$\text{LB} = \max\left\{\left\lceil\left(\sum_{i=1}^{n} R^{i}_{\text{cpu}}\right)\Big/\text{TC}^{j}_{\text{cpu}}\right\rceil, \left\lceil\left(\sum_{i=1}^{n} R^{i}_{\text{mem}}\right)\Big/\text{TC}^{j}_{\text{mem}}\right\rceil\right\}. \tag{25}$$

Table 4. Server consolidation results for VM requirements with reference values as 25% and 45% and probability values as –0.754 and –0.755

Algorithm	$R_p = R_m = 25\%$			$R_p = R_m = 45\%$		
	–0.754			–0.755		
	No of servers (m)	m/LB	Time, s	No of servers (m)	m/LB	Time, s
Fuzzy Firefly Colony-Cuckoo	95	1.05	7.15	192	1.20	8.54
Fuzzy ACS-Cuckoo	95	1.05	7.16	191	1.20	8.56
Firefly Colony	96	1.06	5.28	193	1.20	6.35
ACS	97	1.07	5.31	194	1.21	6.47
MMAS	101	1.12	5.26	195	1.21	6.53
FFD	125	1.38	8.34	218	1.36	24.61

Table 5. Server consolidation results for VM requirements with reference values as 25% and 45% and probability values as –0.348 and –0.374

Algorithm	$\overline{R_p} = \overline{R_m} = 25\%$			$\overline{R_p} = \overline{R_m} = 45\%$		
	–0.348			–0.374		
	No of servers (m)	m/LB	Time, s	No of servers (m)	m/LB	Time, s
Fuzzy Firefly Colony-Cuckoo	94	1.04	7.14	188	1.18	8.51
Fuzzy ACS-Cuckoo	94	1.04	7.16	189	1.18	8.53
Firefly Colony	95	1.05	5.28	190	1.18	6.32
ACS	96	1.06	5.32	191	1.19	6.28
MMAS	98	1.08	5.21	192	1.20	6.52
FFD	121	1.34	8.33	207	1.29	24.69

Table 6. Server consolidation results for VM requirements with reference values as 25% and 45% and probability values as –0.072 and –0.052

Algorithm	$\overline{R_p} = \overline{R_m} = 25\%$			$\overline{R_p} = \overline{R_m} = 45\%$		
	–0.072			–0.052		
	No of servers (m)	m/LB	Time, s	No of servers (m)	m/LB	Time, s
Fuzzy Firefly Colony-Cuckoo	93	1.03	7.11	183	1.14	8.46
Fuzzy ACS-Cuckoo	93	1.03	7.13	183	1.14	8.48
Firefly Colony	94	1.04	5.34	184	1.15	6.25
ACS	95	1.05	5.31	185	1.15	6.27
MMAS	97	1.07	5.22	187	1.16	6.49
FFD	117	1.30	8.24	199	1.24	24.61

Table 7. Server consolidation results for VM requirements with reference values as 25% and 45% and probability values as 0.371 and 0.398

Algorithm	$\overline{R_p} = \overline{R_m} = 25\%$			$\overline{R_p} = \overline{R_m} = 45\%$		
	0.371			0.398		
	No of servers (m)	m/LB	Time, s	No of servers (m)	m/LB	Time, s
Fuzzy Firefly Colony-Cuckoo	93	1.03	7.08	179	1.12	8.42
Fuzzy ACS-Cuckoo	93	1.03	7.12	180	1.12	8.43
Firefly Colony	93	1.03	5.31	183	1.14	6.31
ACS	94	1.04	5.29	184	1.15	6.24
MMAS	96	1.06	5.21	185	1.15	6.47
FFD	112	1.24	8.21	195	1.21	24.54

From the results shown in Fig. 1 we observe that our proposed ACO-Cuckoo and Firefly-Cuckoo algorithms have better server consolidation ratio compared to other approaches considered.

Table 8. Server consolidation results for VM requirements with reference values as 25% and 45% and probability values as 0.775 and 0.751

Algorithm	$\overline{R_p} = \overline{R_m} = 25\%$			$\overline{R_p} = \overline{R_m} = 45\%$		
	0.755			0.751		
	No of servers (m)	m/LB	Time, s	No of servers (m)	m/LB	Time, s
Fuzzy Firefly Colony-Cuckoo	91	1.01	7.06	171	1.08	8.39
Fuzzy ACS-Cuckoo	91	1.01	7.09	172	1.08	8.41
Firefly Colony	92	1.02	5.27	175	1.09	6.21
ACS	93	1.03	5.28	176	1.10	6.23
MMAS	95	1.05	5.18	181	1.13	6.42
FFD	105	1.16	8.19	190	1.18	24.23

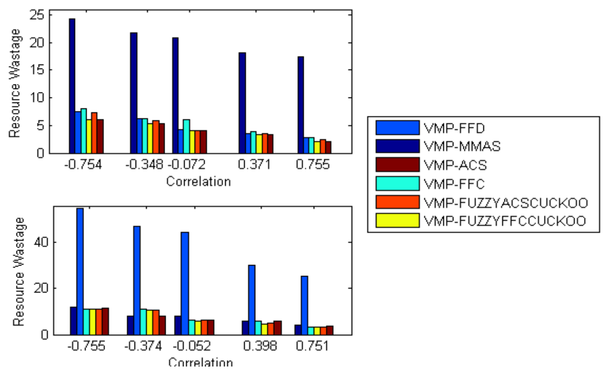

Fig 1. Number of servers utilized when $R_{\text{cpu}} = R_{\text{mem}} = 25\%$ and 45%

5. Virtual machine placement problem

The formulation of VM placement is presented in this section. Here it is considered as a multi-objective optimization problem.

5.1. Problem formulation

Here we formulate the optimization equations of the multi-objective virtual machine placement problem.

5.1.1. Power consumption modelling

The power consumed by the j-th server is [20]

$$P_j = \begin{cases} \left[\left(P_j^{\text{busy}} - P_j^{\text{idle}}\right) \times U_j^p\right] + P_j^{\text{idle}}, & U_j^C > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (26)$$

We have set $P_j^{\text{busy}} = 215$ W and $P_j^{\text{idle}} = 162$ W.

5.1.2. Resource wastage modelling

The resource wastage considered for multi-objective virtual machine placement problem is same as given in Section 2.2.

5.1.3. Objective functions

The minimization of power consumption and resource wastage are defined with next constraints:

(27) $$\text{Minimize} \sum_{j=1}^{m} P_j = \sum_{j=1}^{m} \left[y_j \times \left(\left(P_j^{\text{busy}} - P_j^{\text{idle}} \right) \times \sum_{i=1}^{n} \left(\text{alloc}_{ij} R_{\text{cpu}}^i \right) + P_j^{\text{idle}} \right) \right],$$

(28) $$\text{Minimize} \sum_{j=1}^{m} \text{RW}_j = \\ = \sum_{j=1}^{m} \left[y_j \times \frac{\left| \left(TC_{\text{cpu}}^j - \sum_{i=1}^{n} \left(\text{alloc}_{ij} R_{\text{cpu}}^i \right) \right) - \left(TC_{\text{mem}}^j - \sum_{i=1}^{n} \left(\text{alloc}_{ij} R_{\text{mem}}^i \right) \right) \right| + \in}{\sum_{i=1}^{n} \left(\text{alloc}_{ij} R_{\text{cpu}}^i \right) + \sum_{i=1}^{n} \left(\text{alloc}_{ij} R_{\text{mem}}^i \right)} \right],$$

subject to the constraints same as given in (3)-(6).

5.2. Proposed fuzzy ACO-Cuckoo and fuzzy Firefly-Cuckoo for multiobjective VM placement

The algorithms in Section 4 are reused with different objective functions for this section.

5.2.1. Fuzzy ACO-Cuckoo for multiobjective VM placement

The Fuzzy ant-colony for VM multiobjective placement has different heuristic function as given in the next three equations:

(29) $$\eta_{ij1} = \frac{1}{\varepsilon + \sum_{v=1}^{j} \left(P_v / P_v^{\max} \right)},$$

(30) $$\eta_{ij2} = \frac{1}{\varepsilon + \sum_{v=1}^{j} W_v}.$$

and the total desirability of each VM-PM mapping is

(31) $$\eta_{ij} = \eta_{ij1} + \eta_{ij2}.$$

The fuzzy solution construction process is same as given in Section 4.1. To evaluate the fitness of the obtained solutions, the fuzzy fitness function given by Sait, Bala and El-Maleh [27] is used for multiobjective VM placement problem.

5.2.2. Fuzzy Firefly-Cuckoo for multi-objective VM placement

The same procedures given in Tables 1-4 are used with different objective functions, heuristic equations and solution construction process. The local pheromone update as given in (20) and global pheromone update as given in (21).

Table 9. Comparison of the multiobjective hybrid bioinspired techniques with other algorithms for average power consumption and resource wastage of VM requirements with reference values as 25% and 45 % and probability values as –0.754 and –0.755

Algorithm	$\overline{R_p} = \overline{R_m} = 25\%$				$\overline{R_p} = \overline{R_m} = 45\%$			
	–0.754				–0.755			
	Power, W	Resource wastage	Fuzzy fitness ($\times 10^{-3}$)	Time, s	Power, W	Resource wastage	Fuzzy fitness, ($\times 10^{-3}$)	Time, s
Fuzzy Firefly Colony-Cuckoo	20410	6.10	917	7.23	30671	11.08	913	8.62
Fuzzy ACS-Cuckoo	20420	6.11	916	7.28	30680	11.09	912	8.61
Firefly Colony	20645	7.32	889	5.41	30985	11.21	901	6.23
ACS	20990	7.41	859	5.47	31315	11.42	832	6.41
MMAS	21910	7.96	852	5.70	31348	11.98	836	6.56
FFD	24818	24.26	716	8.54	34969	51.01	756	24.61

Table 10. Comparison of the multiobjective hybrid bioinspired techniques with other algorithms for average power consumption and resource wastage of VM requirements with reference values as 25% and 45 % and probability values as –0.348 and –0.374

Algorithm	$\overline{R_p} = \overline{R_m} = 25\%$				$\overline{R_p} = \overline{R_m} = 45\%$			
	–0.348				–0.374			
	Power, W	Resource wastage	Fuzzy Fitness, ($\times 10^{-3}$)	Time, s	Power, W	Resource wastage	Fuzzy fitness, ($\times 10^{-3}$)	Time, s
Fuzzy Firefly Colony-Cuckoo	20289	5.27	920	7.21	30421	10.71	890	8.60
Fuzzy ACS-Cuckoo	20295	5.28	919	7.25	30480	10.83	918	8.60
Firefly Colony	20460	5.82	895	5.41	30786	10.92	906	6.19
ACS	21586	6.23	867	5.49	31175	11.10	838	6.89
MMAS	21643	6.15	859	5.73	31280	11.12	848	6.52
FFD	24680	21.78	721	8.12	34712	47.23	769	23.24

Table 11. Comparison of the multiobjective hybrid bioinspired techniques with other algorithms for average power consumption and resource wastage of VM requirements with reference values as 25% and 45 % and probability values as –0.072 and –0.052

Algorithm	$\overline{R_p} = \overline{R_m} = 25\%$				$\overline{R_p} = \overline{R_m} = 45\%$			
	–0.072				–0.052			
	Power, W	Resource wastage	Fuzzy fitness, ($\times 10^{-3}$)	Time, s	Power, W	Resource wastage	Fuzzy fitness, ($\times 10^{-3}$)	Time, s
FuzzyFirefly Colony-Cuckoo	18084	4.06	922	7.20	28154	6.06	926	8.96
Fuzzy ACS-Cuckoo	18098	4.10	923	7.21	28175	6.08	925	8.57
Firefly Colony	18264	4.12	904	5.41	28436	6.12	918	6.17
ACS	18453	4.19	876	5.43	28854	6.34	844	6.42
MMAS	19501	5.96	865	5.71	29428	7.86	851	6.51
FFD	24476	20.89	739	8.12	34511	44.31	771	23.15

Table 12. Comparison of the multiobjective hybrid bioinspired techniques with other algorithms for average power consumption and resource wastage of VM requirements with reference values as 25% and 45 % and probability values as 0.371 and 0.398

Algorithm	$\overline{R_p} = \overline{R_m} = 25\%$				$\overline{R_p} = \overline{R_m} = 45\%$			
	0.371				0.398			
	Power, W	Resource wastage	Fuzzy Fitness, ($\times 10^{-3}$)	Time, s	Power, W	Resource wastage	Fuzzy fitness, ($\times 10^{-3}$)	Time, s
Fuzzy Firefly Colony-Cuckoo	17847	3.39	934	7.19	27856	4.77	938	8.95
Fuzzy ACS-Cuckoo	17862	3.42	936	7.19	27912	4.81	936	8.56
Firefly Colony	18200	3.58	908	5.35	28200	5.75	921	6.21
ACS	18215	3.61	884	5.34	28365	5.86	849	6.34
MMAS	19016	3.94	872	5.68	29316	5.89	856	6.42
FFD	21871	18.23	742	7.86	33654	30.18	776	21.12

Table 13. Comparison of the multiobjective hybrid bioinspired techniques with other algorithms for average power consumption and resource wastage of VM requirements with reference values as 25% and 45 % and probability values as 0.755 and 0.751

Algorithm	$\overline{R_p} = \overline{R_m} = 25\%$				$\overline{R_p} = \overline{R_m} = 45\%$			
	0.755				0.751			
	Power, W	Resource wastage	Fuzzy fitness, ($\times 10^{-3}$)	Time, s	Power, W	Resource wastage	Fuzzy fitness, ($\times 10^{-3}$)	Time, s
Fuzzy Firefly Colony-Cuckoo	17304	2.11	954	7.17	26872	3.35	954	8.86
Fuzzy ACS-Cuckoo	17319	2.12	942	7.14	26891	3.39	942	8.51
Firefly Colony	17860	2.42	912	5.35	27250	3.43	926	6.21
ACS	17880	2.71	897	5.34	27342	3.56	857	6.34
MMAS	19048	2.86	886	5.51	28344	4.01	869	6.24
FFD	21280	17.51	754	7.73	33410	25.52	786	21.16

5.3. Computational experimental study

We present the experimental results on the employment of the hybrid multiobjective firefly colony algorithm to solve the VM placement problem.

The experimental platform and parameters used are similar to that presented in Section 4.3. The results obtained are recorded in Tables 9-13. The measures used for comparison are: the average power consumption (Power), average Resource Wastage (RW), average Fuzzy Fitness (FF) and CPU execution times. From the results shown in Fig. 2 we could observe that the hybrid approaches perform better than other single optimization algorithms considered.

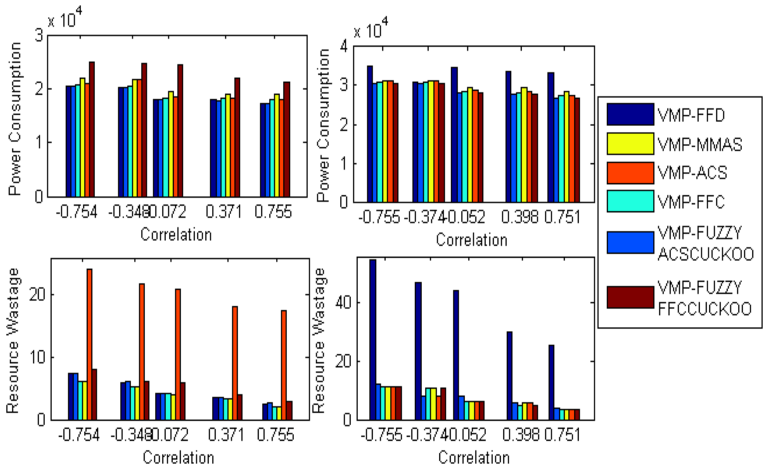

Fig 2. Power consumption and resource wastage of multiobjective techniques when $\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 25\%$ and 45%

6. Conclusion

In this paper we have addressed server consolidation and virtual machine placement problem. In server consolidation problem, we aim to pack as many VMs as possible in the server such that we achieve more resource utilization simultaneously minimizing the number of servers. For the multiobjective virtual machine placement problem, we try to find the optimal placement strategy simultaneously minimizing the power consumption and resource wastage. As a novel approach, we propose two fuzzy bioinspired hybrid optimization algorithms for the above mentioned problems based on the principles of cuckoo search, ant colony and firefly. In our approach, we propose to use cuckoo search to optimize the solutions obtained using Fuzzy ACO and firefly colony. The obtained results of hybrid algorithms are found to be better and more encouraging compared to ACO, Firefly colony, MMAS and FFD. As future research work, we would explore other possible optimization techniques for proposing new hybrid approaches to obtain better optimal solutions.

References

1. Mell, P., T. Grance. The NIST Definition of Cloud Computing, CSRC, US Department of Commerce. 2011.
2. Gupta, R., S. K. Bose, S. Sundarrajan, M. Chebiyam, A. Chakrabarti. A Two Stage Heuristic Algorithm for Solving the Server Consolidation Problem with Item-Item and Bin-Item Incompatibility Constraints. –Proc. of IEEE International Conference on Services Computing, Vol. **2**, 2008, pp. 39-46.
3. Wood, T., P. Shenoy, A. Venkataramani, M. Yousif. Sandpiper: Black-Box and Gray-Box Resource Management for Virtual Machines. – Computer Networks, Vol. **53**, 2009, No 17, pp. 2923-2938.

4. M i s h r a, M., A. S a h o o. On Theory of VM Placement: Anomalies in Existing Methodologies and Their Mitigation Using a Novel Vector Based Approach. – In: Proc. of IEEE International Conference on Cloud Computing (CLOUD'11), 2011, pp. 275-282.
5. B e l o g l a z o v, A., R. B u y y a. Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers. – In: Proc. of 8th International Workshop on Middleware for Grids, Clouds and e-Science, ACM, Vol. **4**, 2010.
6. L i, B., J. L i, J. H u a i, T. W o, Q. L i, L. Z h o n g. Enacloud: An Energy-Saving Application Live Placement Approach for Cloud Computing Environments. – In: Proc. of IEEE International Conference on Cloud Computing, 2009, pp. 17-24.
7. L i, X., Z. Q i a n, Z., R. C h i, B. Z h a n g, S. L u. Balancing Resource Utilization for Continuous Virtual Machine Requests in Clouds. – In: Proc. of 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), IEEE, 2012, pp. 266-273.
8. M a r u y a m a, K., S. K. C h a n g, D. T. T a n g. A General Packing Algorithm for Multidimensional Resource Requirements. – International Journal of Computer & Information Sciences, Vol. **6**, 1977, No 2, pp. 131-149.
9. P a n i g r a h y, R., K. T a l w a r, L. U y e d a, U. W i e d e r. Heuristics for Vector Bin Packing. Research. 2011.
microsoft.com
10. J u n g, G., K. R. J o s h i, M. A. H i l t u n e n, R. D. S c h l i c h t i n g, C. P u. Generating Adaptation Policies for Multi-Tier Applications in Consolidated Server Environments. – In: Proc. of International Conference on Autonomic Computing, 2008, pp. 23-32.
11. F a l k e n a u e r, E. A Hybrid Grouping Genetic Algorithm for Bin Packing. – Journal of Heuristics, Vol. **2**, 1996, No 1, pp. 5-30.
12. B r u g g e r, B., K. F. D o e r n e r, R. F. H a r t l, M. R e i m a n n. Antpacking – An Ant Colony Optimization Approach for the One-Dimensional Bin Packing Problem. – In: Proc. of Evolutionary Computation in Combinatorial Optimization. Springer, 2004, pp. 41-50.
13. R o h l f s h a g e n, P., J. A. B u l l i n a r i a. A Genetic Algorithm with Exon Shuffling Crossover for Hard Bin Packing Proble Ms. – In: Proc. of 9th Annual Conference on Genetic and Evolutionary Computation, ACM, 2007, pp. 1365-1371.
14. A g r a w a l, S., S. K. B o s e, S. S u n d a r r a j a n. Grouping Genetic Algorithm for Solving the Server Consolidation Problem with Conflicts. – In: Proc. of 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation, ACM, June 2009, pp. 1-8.
15. W i l c o x, D., A. M c N a b b, K. S e p p i. Solving Virtual Machine Packing with a Reordering Grouping Genetic Algorithm. – In: Proc. of 2011 IEEE Congress on Evolutionary Computation, 2011, pp. 362-369.
16. F e l l e r, E., L. R i l l i n g, C. M o r i n. Energy-Aware Ant Colony Based Workload Placement in Clouds. – In: Proc. of 12th International Conference on Grid Computing IEEE/ACM, 2011, pp. 26-33.
17. P e r u m a l, B., A. M u r u g a i y a n. A Firefly Colony and Its Fuzzy Approach for Server Consolidation and Virtual Machine Placement in Cloud Datacenters. – Advances in Fuzzy Systems, 2016.
18. W u, Y., M. T a n g, W. F r a s e r. A Simulated Annealing Algorithm for Energy Efficient Virtual Machine Placement. – In: Proc. of IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2012, pp. 1245-1250.
19. L u k e, S. Essentials of Metaheuristics. Lulu, 2009.
http://cs. gmu. edu/sean/book/metaheuristics/.
20. G a o, Y., H. G u a n, Z. Q i, Y. H o u, L. L i u. A Multi-Objective Ant Colony System Algorithm for Virtual Machine Placement in Cloud Computing. – Journal of Computer and System Sciences, Vol. **79**, 2013, No 8, pp. 1230-1242.
21. X u, J., J. A. F o r t e s. Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments. – In: Green Computing and Communications (GreenCom), IEEE/ACM International Conference on & International Conference on Cyber, Physical and Social Computing, 2010, pp. 179-188.
22. S u s e e l a, B. B. J., V. J e y a k r i s h n a n. A Multi-Objective Hybrid ACO-PSO Optimization Algorithm for Virtual Machine Placement in Cloud Computing. – Int. J. Res. Eng. Technol., Vol. **3**, 2014, No 4, pp. 474-476.

23. Z h a o, J., L. H u, Y. D i n g, G. X u, M. H u. A Heuristic Placement Selection of Live Virtual Machine Migration for Energy-Saving in Cloud Computing Environment. PloS One, Vol. **9**, 2014, No 9, e108275.
24. N g u y e n, M. N. P., V. S. L e, H. H. C. N g u y e n. – Energy Efficient Resource Allocation for Virtual Services Based on Heterogeneous Shared Hosting Platforms in Cloud Computing. – Cybernetics and Information Technologies, Vol. **17**, 2017, No 3, pp. 47-58.
25. S a n y a s i N a i d u, P., B. B h a g a t. Emphasis on Cloud Optimization and Security Gaps: A Literature Review. – Cybernetics and Information Technologies, Vol. **17**, 2017, No 3, pp. 165-185.
26. Y a n g, X. S., S. D e b. Cuckoo Search Via Lévy Flights. – In: Proc. of World Congress on IEEE Nature & Biologically Inspired Computing, NaBIC'09, 2009, pp. 210-214.
27. S a i t, S. M., A. B a l a, A. H. E l-M a l e h. Cuckoo Search Based Resource Optimization of Datacenters. – Applied Intelligence, 2015, pp.1-18.
28. D o r i g o, M., L. M. G a m b a r d e l l a. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. – IEEE Transactions on Evolutionary Computation, Vol. **1**, 1997, No 1, pp. 53-66.
29. M a n i e z z o, V. Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem. – INFORMS Journal On Computing, Vol. **11**, 1999, No 4, pp. 358-369.