

Hindawi Publishing Corporation
Advances in Fuzzy Systems
Volume 2016, Article ID 6734161, 15 pages
<http://dx.doi.org/10.1155/2016/6734161>



Research Article

A Firefly Colony and Its Fuzzy Approach for Server Consolidation and Virtual Machine Placement in Cloud Datacenters

Boominathan Perumal¹ and Aramudhan Murugaiyan²

¹Vellore Institute of Technology, Vellore 632014, India

²Perunthalaivar Kamarajar Institute of Engineering and Technology, Puducherry 609603, India

Correspondence should be addressed to Boominathan Perumal; boominathan.p@vit.ac.in

Received 26 November 2015; Revised 12 February 2016; Accepted 14 February 2016

Academic Editor: Ashok B. Kulkarni

Copyright © 2016 B. Perumal and A. Murugaiyan. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Managing cloud datacenters is the most prevailing challenging task ahead for the IT industries. The data centers are considered to be the main source for resource provisioning to the cloud users. Managing these resources to handle large number of virtual machine requests has created the need for heuristic optimization algorithms to provide the optimal placement strategies satisfying the objectives and constraints formulated. In this paper, we propose to apply firefly colony and fuzzy firefly colony optimization algorithms to solve two key issues of datacenters, namely, server consolidation and multiobjective virtual machine placement problem. The server consolidation aims to minimize the count of physical machines used and the virtual machine placement problem is to obtain optimal placement strategy with both minimum power consumption and resource wastage. The proposed techniques exhibit better performance than the heuristics and metaheuristic approaches considered in terms of server consolidation and finding optimal placement strategy.

1. Introduction

Cloud computing is the recent cutting-edge technology to provide computing services over the Internet. It characterizes a paradigm shift from computing as an artefact that is purchased to computing as a service delivered through Internet. Infrastructure as a service (IaaS) is the service model which delivers virtualized computing resources [1, 2]. Resource provisioning is majorly concerned with datacenters which act as the source of massive computing resource pools. Generally, most of the IT services are hosted on dedicated physical servers to handle the complex resource requirement of deployed services and to handle further peak demands of the services. In such a case, this raises a problem of server sprawl, where multiple, underutilized servers take up huge space and resources than required for their current workloads. As a result, this raises the operational costs and investment costs of the service provider. To handle this drawback of server sprawl, virtualization based server consolidation emerged,

to achieve more utilization of physical resources and reduce hardware costs and operational expenses. Virtualization is achieved by partitioning the available physical resources into a number of remote execution environments for virtual machines (VMs). This creates an opinion to the user as if each VM accommodates an individual operating system and henceforth a dedicated physical resource of its own. With an increasing trend towards virtualization based data centers becoming the host platform for wide range of applications, server consolidation and virtual machine placement have become a key research area in cloud computing environment.

Server consolidation runs multiple applications on the same host machine, thus mainly reducing the energy costs and increasing the resource utilization of the servers. It is modelled as a vector bin packing problem where the maximum number of virtual machines is hosted onto a single server such that the total number of physical machines is minimized and thus resource utilization is increased. The other key challenge is to define an optimal mapping strategy

to place a set of virtual machines on physical machines. For a fewer number of virtual and physical machines, an operator might be able to manage the placement of virtual machines. However, when the number of VMs and the number PMs increase, automated placement strategies are preferred. In that case the number of types of possible mapping to be assessed for a given set of virtual and physical machines becomes $(\text{number of physical servers})^{(\text{Number of virtual machines})}$ [3]. Having such a huge mapping strategy or solution space makes the virtual placement problem a NP-hard problem, where it is highly impossible to get an exact placement strategy (VM-PM mapping) with in practically acceptable time period [3]. Hence, there is a need for intelligent methods to fine tune the search for a VM-PM mapping solution to obtain near-optimal placement strategies.

Gupta et al. [4] proposed a two-stage heuristic approach to achieve server consolidation focusing on bin-item incompatibility constraints. In their work, they considered the items to be packed as server itself and the bins for packing as the target servers. Most of the existing works consider VM consolidation as a variant of the bin packing problem and propose some improvements or extensions of simple greedy algorithms such as first fit decreasing (FFD) [5, 6], best fit [7, 8], best fit decreasing [9, 10], and others [6, 11, 12] for constructing the solution. While applying such heuristic approaches, the VM requests are to be sorted in specific order to place in physical machines. In single dimensional case, the sorting process is trivial, whereas for multidimensional placement problem, the sorting methods proposed by Maruyama et al. [13] can be used. In recent years, few efforts are taken to convert the multidimensional sizes considered as vectors into sizes of scalar type. Panigrahy et al. [14] proposed a novel method of geometric heuristics and presented a report on their findings related to various arrangements of vectors to produce the scalar (size). However, the study does not provide any rationale for when to use a particular heuristics or why one heuristics outperforms others. Wood et al. [15] proposed a Sandpiper system to detect hotspot and accordingly enable VM migrations. Mishra and Sahoo [16] proposed a novel vector algebra 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 Li et al. [17] and Jung et al. [18].

Several metaheuristic approaches are also widely used for VM placement. Falkenauer [19] proposed an enhanced approach of genetic algorithm called grouping genetic algorithm to handle the server consolidation problem. In the proposed approach, group-based encoding scheme with new mutation and crossover operation is used rather than individual encoding technique of traditional GA method. Brugger et al. [20] proposed ACO metaheuristic that had better performance than genetic algorithm for large problem instances. Rohlfschagen and Bullinaria [21] proposed another variant of GGA called exon shuffling genetic algorithm. A group genetic algorithm is proposed by Shubham et al. [22] to solve server consolidation problem as a vector packing problem with conflicts and has shown that they achieve better results compared to other approaches. Xu and Fortes [23] solved

VM placement as a multiobjective optimization problem. As a further enhancement to grouping genetic algorithm, a reordering grouping genetic algorithm is given by Wilcox et al. [24]. Feller et al. [25] used another version of ACO to address VM consolidation and obtained encouraging results rather than FFD. However, their evaluation was simplified to one-dimensional resource as they varied only the number of cores demanded by VM and retained resource demands unchanged. Mills et al. [26] proposed methods for initial VM placements and compared against 18 initial placement algorithms. Wu et al. in [27] proposed simulated annealing technique for VM placement. Luke [28] designed new mutation and crossover operations where swap, move, and remove are considered for mutation and these operations are applied to the steady state genetic algorithm for VM placement.

Adamuthe et al. [29] addressed VM placement as a multiobjective optimization problem and compared the results of genetic algorithms, nondominated sorting genetic algorithm-(NSGA-) I and NSGA-II. Finally they concluded that NSGA-II provides diversified good solutions. An improved particle swarm optimization approach for virtual machine placement is proposed by Wang et al. [30]. A multiobjective ant colony system for virtual machine placement problem is proposed by Gao et al. [31]. In this work, they proposed VMPACS that optimizes the datacenter with both minimum power consumption and resource wastage simultaneously. A novel multiobjective memetic algorithm is also proposed to solve virtual machine placement problem [32]. Ant colony optimization is used to solve multiobjective optimization problem to optimize total processing resource wastage and memory resource wastage [33]. A novel family genetic algorithm is proposed to address the problem of VM placement with less energy consumption and VM migrations [34]. In addition, Tang and Pan [35] provided another approach using hybrid genetic algorithm addressing virtual machine placement with an objective of reducing energy consumption.

Here, we propose to apply a firefly colony (FCO) optimization algorithm and fuzzy firefly colony (FFCO) optimization algorithm for server consolidation and multiobjective virtual machine placement (VMP) problem. The firefly colony [36] approaches used are seen as a variant of firefly which is based on ant colony optimization (ACO) algorithm. In this firefly approach, the fireflies are taken as collaborating learning agents like ants in ACO. In contrast to the standard firefly algorithm, firefly colony algorithm is a distributed, autocatalytic, and constructive greedy metaheuristic. In this approach, it is assumed that the phosphorescent substance released by the fireflies is absorbed by the paths and, in turn, they glow. The decision-making is based on the probabilistic choice, which is biased by the concentration of the phosphorescent glowing in its path. This in turn has an autocatalytic effect that the path chosen by the fireflies will have higher probability of getting chosen again by other fireflies in the upcoming visits. It is also assumed that while returning also the same path would be chosen due to increase in brightness on the path. The reason behind choosing this approach for virtual machine placement is as follows [36]. Firstly, compared to standard firefly, this firefly colony algorithm works fast as pairwise assessment of flies is eluded in this

approach. Secondly, randomization capacity of the basic FA is gradually decreased as it reaches the optima, and hence the performance of the FCO is improved. Thirdly, as it is the hybridization of greedy metaheuristic algorithm and the exploitation and exploration process of firefly algorithm, it turns out to be a prevailing approach for server consolidation and virtual machine placement problem [36]. In server consolidation, we aim to minimize the number of physical servers and in VM placement we aim to simultaneously minimize the power consumption and resource wastage while constructing the solution for placement. From the experimental results conducted, we could show that our proposed approach outperforms ant colony system, max-min ant system, and first fit decreasing heuristics.

The rest of the paper is organized as follows. In Section 2, we present the basic firefly algorithm. In Section 3, we present firefly colony algorithm, fuzzy firefly colony algorithm for server consolidation problem with the experimental results obtained. In Section 4, the multiobjective virtual machine placement problem is presented with the experimental results. Section 5 concludes the paper followed by references.

2. Basic Firefly Optimization Technique

The firefly algorithm (FFA) developed by Yang [37–39] is a swarm intelligence based metaheuristic approach, inspired by the flashing behaviour of fireflies. Fireflies are simple insects living in groups. The firefly flashes act as a signal system to attract (communication) other files. The firefly algorithm is based on these flashing patterns and the behaviour of fireflies. The characteristics of standard firefly algorithm are [37] as follows:

- (1) The fireflies are unisex and they get attracted to other fireflies regardless of their sex.
- (2) The less bright fireflies get attracted to brighter ones. The attractiveness and brightness of the firefly decrease as the distance increases and the fireflies start moving randomly if they do not find brighter fireflies in their path.
- (3) The brightness of a firefly depends on the setting of the objective function.

The two important factors of firefly algorithm are the light intensity and the attractiveness. We know that the light intensity I decreases as the distance r increases in terms of $I \propto 1/r^2$ obeying inverse square law [37]. At the same time as the air absorbs light, it becomes feeble as the distance increases. This makes the fireflies visible only for certain distance. Basically, the light intensity $I(x)$ varies with the distance r monotonically and exponentially as given in [37]

$$I(r) = I_0 e^{-\gamma r^2}. \quad (1)$$

To decrease the function monotonically at a slower rate, the approximation given by Yang [37] is

$$I(r) = \frac{I_0}{1 + \gamma r^2}. \quad (2)$$

The difference in attractiveness β as the distance r changes is given as [37]

$$\beta = \beta_0 e^{-\gamma r^2}, \quad (3)$$

where β_0 is the attractiveness at $r = 0$. It is noted that when $m > 0$, the exponent γr^2 can be replaced by γr^m . As it is often faster to calculate $1/1 + r^2$ than an exponential function, the above function, if necessary, can conveniently be replaced by

$$\beta = \frac{\beta_0}{1 + \gamma r^2}. \quad (4)$$

The distance between any two fireflies is given as [37]

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^d (x_{ik} - x_{jk})^2}. \quad (5)$$

The movement of firefly i towards firefly j is determined as [37]

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2} (x_j^t - x_i^t) + \alpha_i \left(\text{rand} - \frac{1}{2} \right). \quad (6)$$

3. Firefly Colony Approach for Server Consolidation

Here we present the problem formulation and the proposed firefly colony algorithm for server consolidation problem. Server consolidation is a variant of vector packing problem. We consider CPU and memory as two major dimensions of resource utilization for server consolidation problem.

3.1. Problem Formulation. Suppose we are given n virtual machines (VMs) $i \in I$ to place in m physical machines (PMs) $j \in J$; we assume that no virtual machine requires more capacity than can be provided by a single server. Let $[R_{\text{cpu}}^i \ R_{\text{mem}}^i]$ be the processing unit demand and memory demand of each VM. Let $[TC_{\text{cpu}}^j \ TC_{\text{mem}}^j]$ be the processing unit capacity and memory capacity associated with each physical machine. A threshold of 90% is set for $[TC_{\text{cpu}}^j \ TC_{\text{mem}}^j]$ to avoid physical machine resource utilization reaching 100% as it may lead to severe performance degradation when it is fully utilized [31]. Two decision variables are 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 . A binary variable $y_j \in \{0, 1\}$ indicates whether a server is in use or not.

3.2. Resource Wastage Modelling. The possible cost of wasted resources is determined using [31]

$$\text{RW}_j = \frac{|L_j^{\text{cpu}} - L_j^{\text{mem}}| + \varepsilon}{|U_j^{\text{cpu}} + U_j^{\text{mem}}|}, \quad (7)$$

where RW_j is the resource wastage of the j th server and U_j^{mem} and U_j^{cpu} are the normalized memory usage and CPU resource usage, respectively. In other words, it is the ratio of used resource to total available resources. L_j^{mem} and L_j^{cpu} are the normalized remaining memory and CPU usage, respectively. ε is a small positive constant set to 0.0001 [31].

```

Step 1. Initialize Number of physical machines (PMs), Number of virtual machines (VMs)
Step 2. Set List of physical machines and their current usage
Step 3. Generate VM request demands using the procedure given in Algorithm 2.
Step 4. Set Light absorption coefficient  $\gamma$  and maxIterations
Step 5. Initialize the attractiveness matrix  $\beta_0$ 
Step 6. Repeat
Step 7. For each firefly  $k = 1 : NF$  (Number of Fireflies)
Step 8. Repeat
Step 9. For each physical server  $j = 1 : m$  do
Step 10. Issue a new server  $j$  from the set of random ordered physical servers
Step 11. Determine the set of eligible virtual machines  $\Omega_k(j)$  using (19)
Step 12. Repeat
Step 13. For each residual VM that is eligible to be placed in the currently chosen server
Step 14. Calculate the heuristic information of the VM-PM mapping using (13)
Step 15. Calculate the attractiveness of the VM-PM mapping using (16)-(17)
Step 16. End For
Step 17. Decide the next VM to place in server by applying pseudo random proportional rule (Equation (14))
Step 18. Apply local attractiveness updating rule according to (21)
Step 19. Until no remaining VM can be placed in the server anymore
Step 20. Until all VMs are placed in the server
Step 21. End For
Step 22. Find iteration best firefly solution ( $S^{ib}$ )
Step 23. if  $ff_{sc}(S^{ib}) > globalbest$  then
Step 24.  $S^{gb} \leftarrow S^{ib}$ 
Step 25.  $globalbest \leftarrow ff_{sc}(S^{ib})$ 
Step 26. End if
Step 27. Apply global attractiveness update rule according to (22)
Step 28. Until the maximum iteration is reached

```

ALGORITHM 1: Firefly colony algorithm for server consolidation.

3.3. *Objective Function.* The objective is to minimize the number of servers used while no capacity constraint is violated. The minimization function is formulated as follows [40]:

$$\text{Minimize } \sum_{j=1}^m y_j \quad (8)$$

subject to constraints:

$$\sum_{j=1}^m \text{alloc}_{ij} = 1 \quad \forall i \in I, \quad (9)$$

$$\sum_{i=1}^n R_{cpu}^i \cdot \text{alloc}_{ij} \leq TC_{cpu}^j \cdot y_j \quad \forall j \in J, \quad (10)$$

$$\sum_{i=1}^n R_{mem}^i \cdot \text{alloc}_{ij} \leq TC_{mem}^j \cdot y_j \quad \forall j \in J, \quad (11)$$

$$y_j, \text{alloc}_{ij} \in \{0, 1\} \quad \forall i \in I, \forall j \in J. \quad (12)$$

Constraint (9) assigns VM i to only one of the servers. The capacity constraint of the server is specified in constraint (10) and (11). The domain of the binary decision variables used is given in (12).

```

for  $i = 1 : n$ 
   $R_{cpu}^i = \text{rand}(2\overline{R_{cpu}})$ 
   $R_{mem}^i = \text{rand}(\overline{R_{mem}})$ 
   $r = \text{rand}(1.0)$ 
  if  $(r < p \wedge R_{cpu}^i \geq \overline{R_{cpu}}) \vee (r \geq p \wedge R_{cpu}^i < \overline{R_{cpu}})$  then
     $R_{mem}^i = R_{mem}^i + \overline{R_{mem}}$ 
  End if
End for

```

ALGORITHM 2: Generate instances procedure [31].

3.4. *Firefly Colony Optimization Algorithm.* The firefly colony optimization algorithm [36] is a swarm based metaheuristic approach based on the flashing patterns and behaviour of the fireflies. The procedure of the firefly colony approach for virtual machine placement is given in Algorithm 1. In the initialization phase, the initial attractiveness matrix is set to β_0 as given in Section 3.4.3. In the iterative process, each firefly constructs solution incrementally using a greedy stochastic process where it probabilistically assigns VM to the server. The stochastic policy is indicated by attractiveness trail and heuristic information. Finally the best so far solution is used to update the attractiveness values.

3.4.1. Procedure for Firefly Solution Construction. Every firefly initiates the solution construction process with a set of all VMs and randomly arranged set of available servers. For each server chosen, the firefly starts allocating the VMs one by one. The choice of the next VM i to place in current server j is based on some probability which is proportional to the attractiveness. In other words, the higher is the attractiveness of VM i among all eligible virtual machines $\Omega_k(j)$ to pack in current server j ; then the larger is its probability to be chosen. Once a particular server reaches the state where no more VMs can be added, then the server is closed and a new server is hosted for packing VMs. The process continues until there are no more VMs for each server and no more servers left for packing VMs. In this process of placement, the packing arrangements of the VMs from the server 1 to server j are known dynamically based on the current state of the firefly; hence, this information is used to calculate the resource wastage during the process of placing VM u in server j using the equation given as follows:

$$RW_{uj} = \sum_{v=1}^j RW_v. \quad (13)$$

The pseudo random proportional rule used for solution construction is given in (14) which includes both exploitation and exploration process:

$$i = \begin{cases} \operatorname{argmax}_{u \in \Omega_k(j)} \{\beta_{uj} * e^{-\gamma RW_{uj}^m}\}, & q \leq q_0; \\ \text{explore } e, & \text{otherwise,} \end{cases} \quad (14)$$

where γ is the absorption coefficient of the light initialized to 1 and β_{uj} is the attractiveness trail defined as the constructive factor of assigning VM u to server j having set of VMs already placed in it. The virtual u belongs to a set of eligible virtual machines.

Equation (14) can be rewritten as

$$i = \begin{cases} \operatorname{argmax}_{u \in \Omega_k(j)} \{\beta_{uj} * \eta_{uj}\}, & q \leq q_0; \\ \text{explore } e, & \text{otherwise,} \end{cases} \quad (15)$$

where q is a random variable uniformly distributed in $[0, 1]$ and q_0 is a fixed variable having a value between 0 and 1. If q is less than q_0 , then the process is called exploitation where VM u with higher attractiveness is chosen from a set of eligible virtual machines. If q is greater than q_0 , then the process is called exploration where we find the aggregation of the attractiveness of all eligible VMs using (16), and then we choose the VM having the higher attractiveness than a random number:

$$\text{Attractiveness_Array} = \operatorname{cumulative\,sum}(\beta_{ij}^k), \quad (16)$$

$$i \in \Omega_k(j),$$

where

$$\beta_{ij}^k(t) = \begin{cases} \beta_{ij} * \eta_{ij}, & i \in \Omega_k(j); \\ 0, & \text{otherwise,} \end{cases} \quad (17)$$

where $\beta_{ij}^k(t)$ is the attractiveness at iteration t for placing VM i in server j along with other virtual machines packed as a partial solution. The attractiveness value β_{ij} is determined as

$$\beta_{ij} = \begin{cases} \frac{\sum_{u \in \Omega_k(j)} \beta_{ui}}{|\Omega_k(j)|}, & \text{if } \Omega_k(j) - \{i\} \neq \emptyset; \\ 1, & \text{otherwise.} \end{cases} \quad (18)$$

The set of eligible virtual machines which are determined using (19) as given by Gao et al. [31] is

$$\Omega_k(j) = \left\{ i \in \{1, \dots, n\} \mid \left(\sum_{u=1}^m \operatorname{alloc}_{iu} = 0 \right) \wedge \left(\left(\sum_{u=1}^n (\operatorname{alloc}_{uj} \times R_{\text{cpu}}^u) + R_{\text{cpu}}^i \right) \leq TC_{\text{cpu}}^j \right) \wedge \left(\left(\sum_{u=1}^n (\operatorname{alloc}_{uj} \times R_{\text{mem}}^u) + R_{\text{mem}}^i \right) \leq TC_{\text{mem}}^j \right) \right\}. \quad (19)$$

The fitness function used for evaluating the constructed solution is [40]

$$\begin{aligned} \text{ff}_{\text{SC}(S)} &= \frac{\sum_{j=1}^m \left(\sum_{i=1}^n (\operatorname{alloc}_{ij} \cdot R_{\text{cpu}}^i) / (TC_{\text{cpu}}^j \cdot y_j) \right)^k}{m} \\ &+ \frac{\sum_{j=1}^m \left(\sum_{i=1}^n (\operatorname{alloc}_{ij} \cdot R_{\text{mem}}^i) / (TC_{\text{mem}}^j \cdot y_j) \right)^k}{m}, \end{aligned} \quad (20)$$

where k is assigned with value of 2 as per the literature given by Falkenauer [19].

3.4.2. Fitness Function Evaluation. The quality of the constructed solution is assessed using (20).

3.4.3. Attractiveness Updating. Another important part of firefly algorithm is the updating of attractiveness trails. The attractiveness trails can either increase as phosphorescent substance is emitted or decrease as the substance evaporates. The emission of phosphorescent substance is based on the fact that the information contained in some good solutions should be shown in the attractiveness trails and the movement included in these good solutions will be biased by other flies which are constructing the subsequent solutions. However, the phosphorescent substance evaporation favours the exploration of new areas of search space as it implements a useful form of forgetting and also it avoids too rapid form of convergence of the algorithm towards a suboptimal region. In this approach, the attractiveness update is done in two stages: after a firefly k finds a feasible solution to the VM placement problem; a local updating is performed according to the following equation [36]:

$$\beta_{ij}(t+1) = \alpha \left(\operatorname{rand} - \frac{1}{2} \right) \beta_{ij}(t) + \beta_0, \quad (21)$$

where α is the attractiveness decay parameter, the initial β_0 values are calculated using $1/[n \cdot \text{ff}_{\text{SC}}(S_0)]$, where n is the number of VMs, and S_0 is the initial solution obtained using FFD heuristic. Once all the fireflies construct the

solution, the best so far solution is used to globally modify the attractiveness. The global update rule is given as

$$\beta_{ij}(t+1) = \alpha \left(\text{rand} - \frac{1}{2} \right) \beta_{ij}(t) + \Delta \beta_{ij}(t), \quad (22)$$

where

$$\Delta \beta_{ij}(t) = \begin{cases} \text{ff}_{\text{SC}}(S^{\text{gb}}), & \text{if VM } i \text{ placed in server } j \in \text{global best solution;} \\ 0, & \text{otherwise.} \end{cases} \quad (23)$$

3.5. Fuzzy Firefly Colony Approach for Server Consolidation.

A new variant of firefly colony named fuzzy firefly colony approach is used to handle the uncertainties that could exist when fireflies makes a choice to choose next virtual machine for placement in current server. In this variant, the control strategies of the fireflies are established using fuzzy rules [41–47]. The fuzzy firefly colony algorithm follows same placement procedure given in Algorithm 1 except that step 17 in the procedure is modified to include fuzzy strategy and fuzzy probable strategy for choosing the next virtual machine for placement. For the placement problem, chosen a random server, the next virtual machine i to place in it depends on the attractiveness trail and the desirability (heuristic information) of placing the virtual machine i in current server j . There is a possibility that the fireflies can remark the desirability of placing the virtual machine and the attractiveness trail as fuzzy values (low, medium, or high). Hence, when choosing the next virtual machine, the fireflies will have greater or lesser efficacy on choosing the virtual machine, based on their attractiveness and the desirability. These efficacies can also be described by the appropriate fuzzy sets (very very low, very low, low, medium, high, very high, or very very high). The fuzzy sets of all these state variables are defined as triangular membership functions.

The approximate reasoning to determine the k th firefly efficacy in choosing the next VM i for the current server j consists of following rules for each eligible virtual machine:

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.

The fuzzy rule contains both attractiveness and heuristic information in the antecedent part and the efficacy of choosing the next VM in the consequent part. 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 . Here we introduce two strategies, namely, fuzzy strategy and fuzzy probable strategy, to implement the exploitation and exploration process of deciding the next VM i to place in current server j [45]

$$I = \begin{cases} \text{Fuzzy strategy,} & q \leq q_0 \text{ (exploitation);} \\ \text{Fuzzy probable strategy,} & q > q_0 \text{ (exploration).} \end{cases} \quad (24)$$

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 ($q \leq q_0$) where, among all the eligible virtual machines, the virtual machine u having the maximum efficacy is chosen to be the next VM for placement. In other words, maximum defuzzification method is used to produce the output fuzzy sets as fuzzy singletons [45], where $i = u^*$ such that

$$[e_{u^*j}^k] = \sup_{u \in \Omega_k(j)} \{e_{uj}^k\}. \quad (25)$$

Fuzzy Probable Strategy (FPS). The fuzzy probable strategy is introduced to implement the exploration process of fireflies. In contrast to the exploitation process, here we consider that the VM i with maximum efficacy is not essential to be selected as the next VM, but it has the highest chance to be selected; that is, the next VM can be probably i [45]. The fuzzy

TABLE 1: Server consolidation results for VM requirements with reference values as 25% and 45% and correlation coefficients as -0.754 and -0.755 .

Reference value	$\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 25\%$			$\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 45\%$		
Correlation coefficient	-0.754			-0.755		
Algorithm	Number of physical machines used (m)	Average consolidation ratio	CPU execution time (S)	Number of physical machines used (m)	Average consolidation ratio	CPU execution time (S)
Fuzzy firefly colony	95	1.05	5.31	192	1.20	6.39
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

probability (attractiveness) $\widetilde{\beta}_{ij}^k$ for the virtual machine i to be chosen is equivalent to the

$$\widetilde{\beta}_{ij}^k = e_{ij}^k \cdot \widetilde{F}_{ij}^k. \quad (26)$$

To obtain \widetilde{F}_{ij}^k a random number $0 \leq r_u \leq 1$ is generated for each eligible virtual machine u belonging to $\Omega_k(j)$. Then

$$\widetilde{F}_{ij}^k = \frac{r_i}{\sum_{u=1}^n r_u}. \quad (27)$$

Like Fuzzy strategy, maximum defuzzification method is applied on fuzzy probability to choose the next virtual machine i among u eligible virtual machines for current server j .

3.6. Experimental Results. The VM requirements instances are generated using the procedure given in Algorithm 2 [31]. Reference-based VM resource demands are generated where Ref = $z\%$ means each randomly generated VM resource demand falls in the interval $[0, 2z]$. It is to be noted that we can choose any z value as reference value. For example, if we choose Ref = 10%, 15%, 20%, 25%, and 45% on average we get 90/10, 90/15, 90/20, 90/25, and 90/45 VMs per server.

In Algorithm 2 we consider CPU and memory as two dimensions of VM requirement. $\overline{R}_{\text{cpu}}$ is the reference CPU utilization, $\overline{R}_{\text{mem}}$ is the reference memory utilization, and the probability p is the reference value. For our experimental study, we have used two reference values and five probabilities as followed by Gao et al. [31]. We set both $\overline{R}_{\text{cpu}}$ and $\overline{R}_{\text{mem}}$ to 25% and then to 45%. The distributions of CPU and memory utilizations are in the range $[0, 50\%]$ when $\overline{R}_{\text{cpu}}$ and $\overline{R}_{\text{mem}}$ are set to 25% and $[0, 90\%]$ when $\overline{R}_{\text{cpu}}$ and $\overline{R}_{\text{mem}}$ are set to 45%. The linear correlation between the input dimensions is controlled to a certain limit by changing the probability p . For $\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 25\%$, the probability values of 0.00, 0.25, 0.50, 0.75, and 1.0 are used. The average correlation coefficients for the stated probabilities are -0.754 , -0.348 , -0.072 , 0.371 , and 0.755 and these correspond to strong-negative, weak-negative, no-correlation, weak-positive, and strong-positive correlations, respectively [31]. These five probability values

are chosen to show that as the correlation between CPU and memory increases, the average power consumption and resource wastage are minimized. We similarly set p for $\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 45\%$ and the correlation coefficients given are -0.755 , -0.374 , -0.052 , 0.398 , and 0.751 . For each of the experiments conducted, an upper bound of 90% is imposed as the threshold value for the resource utilization of the single server [31]. Using the above procedure and parameter settings, the instances were generated for specified number of VMs. The programs for the proposed algorithm and all other variants were coded in MATLAB and ran on an Intel Pentium® Dual-Core processor with 2.50 GHz CPU and 8 GB RAM. To support the worst VM placement scenario, the number of servers was set to the number of VMs, in which only one VM is assigned per server [31].

In Tables 1–5, the performance of the proposed multiobjective firefly algorithm and fuzzy firefly algorithm for placing 300 VMs is compared to that of ant colony system (ACS), max-min ant system (MMAS), and first fit decreasing (FFD) methods of VM placement. For the proposed firefly algorithms, we set gamma value to be 0.2. From the literature, it is noted that on an average 10 to 20 runs are performed for each instance generated varying the probability and the reference value. For our experimental study, we performed 20 runs and each run is repeated for 100 iterations. The final results reported are average of 20 runs. The performance measures reported are the average number of physical machines used (m), average consolidation ratio ($m/\text{lower bound}$), and CPU execution time in seconds. The lower bound of the minimum number of servers that can be used for placement [40, 48] is calculated using

Lower Bound

$$= \max \left\{ \left\lceil \frac{\left(\sum_{i=1}^n R_{\text{cpu}}^i \right)}{TC_{\text{cpu}}} \right\rceil, \left\lceil \frac{\left(\sum_{i=1}^n R_{\text{mem}}^i \right)}{TC_{\text{mem}}} \right\rceil \right\}. \quad (28)$$

When the average number of servers used for placement reaches the lower bound then the server consolidation ratio ($m/\text{lower bound}$ (LB)) gets closer to 1 [48].

TABLE 2: Server consolidation results for VM requirements with reference values as 25% and 45% and correlation coefficients as -0.348 and -0.374 .

Reference value	$\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 25\%$			$\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 45\%$		
Correlation coefficient	-0.348			-0.374		
Algorithm	Number of physical machines used (m)	Average consolidation ratio	CPU execution time (S)	Number of physical machines used (m)	Average consolidation ratio	CPU execution time (S)
Fuzzy firefly colony	94	1.04	5.29	189	1.18	6.37
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	24.59

TABLE 3: Server consolidation results for VM requirements with reference values as 25% and 45% and correlation coefficients as -0.072 and -0.052 .

Reference value	$\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 25\%$			$\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 45\%$		
Correlation coefficient	-0.072			-0.052		
Algorithm	Number of physical machines used (m)	Average consolidation ratio	CPU execution time (S)	Number of physical machines used (m)	Average consolidation ratio	CPU execution time (S)
Fuzzy firefly colony	93	1.03	5.28	183	1.14	6.22
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 4: Server consolidation results for VM requirements with reference values as 25% and 45% and correlation coefficients as 0.371 and 0.398 .

Reference value	$\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 25\%$			$\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 45\%$		
Correlation coefficient	0.371			0.398		
Algorithm	Number of physical machines used (m)	Average consolidation ratio	CPU execution time (S)	Number of physical machines used (m)	Average consolidation ratio	CPU execution time (S)
Fuzzy firefly colony	92	1.02	5.27	180	1.12	6.23
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

TABLE 5: Server consolidation results for VM requirements with reference values as 25% and 45% and correlation coefficients as 0.775 and 0.751 .

Reference value	$\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 25\%$			$\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 45\%$		
Correlation coefficient	0.755			0.751		
Algorithm	Number of physical machines used (m)	Average consolidation ratio	CPU execution time (S)	Number of physical machines used (m)	Average consolidation ratio	CPU execution time (S)
Fuzzy firefly colony	91	1.01	5.25	173	1.08	6.22
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

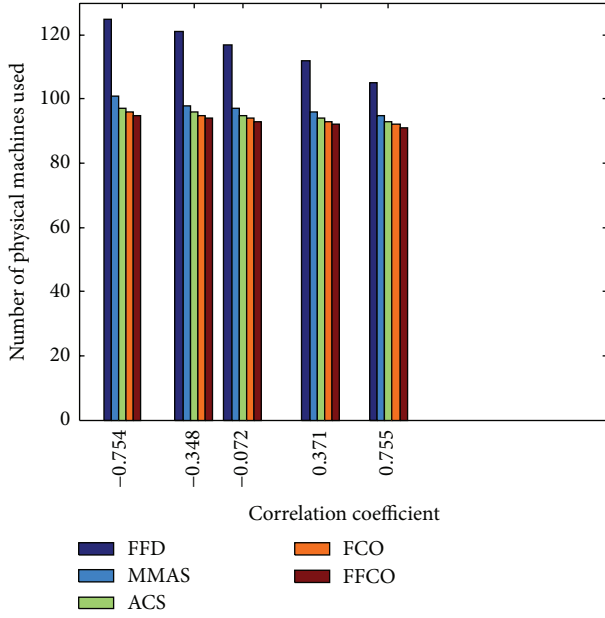


FIGURE 1: Number of servers utilized when $\overline{R}_{cpu} = \overline{R}_{mem} = 25\%$.

From the results shown in Figures 1 and 2 we could observe that our proposed firefly colony approach utilizes minimum number of servers compared to other deterministic and nondeterministic approaches.

4. Multiobjective Firefly Colony Approach for Virtual Machine Placement

In this section we present the problem formulation and the implementation of firefly colony and fuzzy firefly colony

for solving the multiobjective virtual machine placement problem. The multiobjectives considered are the power consumption and resource wastage which are to be minimized simultaneously.

4.1. Power Consumption and Resource Wastage Modelling.

The power consumption of the j th server is defined as a function of the CPU utilization [31]:

$$P_j = \begin{cases} [(P_j^{\text{busy}} - P_j^{\text{idle}}) \times U_j^p] + P_j^{\text{idle}}; & U_j^c > 0; \\ 0; & \text{otherwise,} \end{cases} \quad (29)$$

where P_j^{idle} and P_j^{busy} are the average power values when the j th server is idle and fully utilized, respectively. The power consumption in idle state is not a part of the total energy consumed by the CPU. We have set $P_j^{\text{busy}} = 215$ Watts and $P_j^{\text{idle}} = 162$ Watts for homogeneous physical servers [31]. The resource wastage considered for multiobjective virtual machine placement problem is same as specified in server consolidation problem given in (7).

4.2. Objective Functions. Formally, the problem of VM placement is formulated as follows. Given a set of n virtual machines, m physical servers, and the physical servers hosting n_i VMs, the VM placement algorithms generate possible mapping solutions for n VMs on the m physical servers under a specified set of constraints. The objective functions to be minimized are the total power consumption and resource wastage and these objective functions are modelled as follows [31]:

$$\text{Minimize } \sum_{j=1}^m P_j = \sum_{j=1}^m \left[y_j \times \left((P_j^{\text{busy}} - P_j^{\text{idle}}) \times \sum_{i=1}^n (\text{alloc}_{ij} \cdot R_{\text{cpu}}^i) + P_j^{\text{idle}} \right) \right], \quad (30)$$

$$\text{Minimize } \sum_{j=1}^m RW_j = \sum_{j=1}^m \left[y_j \times \frac{|(TC_{\text{cpu}}^j - \sum_{i=1}^n (\text{alloc}_{ij} \cdot R_{\text{cpu}}^i)) - (TC_{\text{mem}}^j - \sum_{i=1}^n (\text{alloc}_{ij} \cdot R_{\text{mem}}^i))| + \epsilon}{\sum_{i=1}^n (\text{alloc}_{ij} \cdot R_{\text{cpu}}^i) + \sum_{i=1}^n (\text{alloc}_{ij} \cdot R_{\text{mem}}^i)} \right] \quad (31)$$

subject to

$$\begin{aligned} \sum_{j=1}^m \text{alloc}_{ij} &= 1 \quad \forall i \in I, \\ \sum_{i=1}^n R_{\text{cpu}}^i \cdot \text{alloc}_{ij} &\leq TC_{\text{cpu}}^i \cdot y_j \quad \forall j \in J, \\ \sum_{i=1}^n R_{\text{mem}}^i \cdot \text{alloc}_{ij} &\leq TC_{\text{mem}}^j \cdot y_j \quad \forall j \in J, \\ y_j, \text{alloc}_{ij} &\in \{0, 1\} \quad \forall i \in I, \forall j \in J. \end{aligned} \quad (32)$$

The objective function given in (30) is to minimize the power consumption and the objective function in (31) focuses on reducing the resource wastage. As we consider the VMP problem as a multiobjective combinatorial optimization problem the optimization strategy focuses on simultaneously minimizing the power consumption and the resource wastage under the given set of constraints.

4.3. Firefly Colony Procedure for Multiobjective Virtual Machine Placement. The initial attractiveness matrix for the multiobjective virtual machine placement problem is determined using $\beta_0 = 1/[n \cdot (P'(S_0) + RW(S_0))]$, where n is the number of VMs and S_0 is the solution generated by FFD

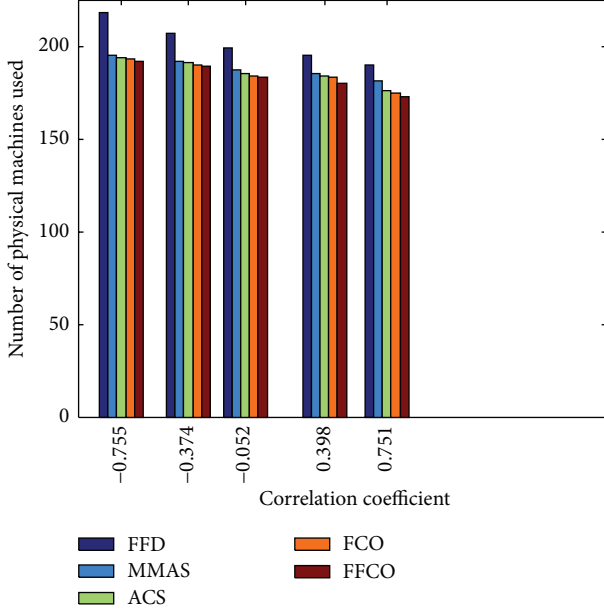


FIGURE 2: Number of servers utilized when $\overline{R_{cpu}} = \overline{R_{mem}} = 45\%$.

heuristic. $RW(S_0)$ is the resource wastage of the solution S_0 (see (7)). $P'(S_0)$ is the normalized power consumption of the solution S_0 which is determined as [31]

$$P'(S_0) = \sum_{j=1}^m \left(\frac{P_j}{P_j^{\max}} \right). \quad (33)$$

As given in Section 3.4.1, the packing arrangements of the VMs from the server 1 to server j are known dynamically based on the current state of the firefly; hence this information is used to calculate the normalized power consumption and resource wastage during the process of multiobjective VM placement using

$$P'_{uj} = \sum_{v=1}^j \left(\frac{P_v}{P_v^{\max}} \right). \quad (34)$$

The pseudo random proportional rule in (14) is rewritten for multiobjective solution construction (see (35)) which includes both power consumption and resource wastage information:

$$i = \begin{cases} \operatorname{argmax}_{u \in \Omega_k(j)} \left\{ \beta_{uj} * e^{-\gamma(P'_{uj} + RW_{uj})^m} \right\}, & q \leq q_0; \\ \operatorname{explore} e, & \text{otherwise,} \end{cases} \quad (35)$$

where γ is the absorption coefficient of the light initialized to 1, β_{uj} is the attractiveness trail defined as the favourability of assigning VM u to server j having set of VMs already placed in it. The virtual u belongs to the set of eligible virtual machines. RW_{uj} is determined using (13). The procedure for multiobjective firefly colony and fuzzy fire colony approach is same as the procedure given in Section 3.4 except for the objective functions, fuzzy strategy, and the fitness evaluation method used. The objective functions are given above in

Section 4.2. The fuzzy strategy to decide the next virtual machine is given in Section 3.5. The fuzzy fitness evaluation given by Sait et al. [48] is used here for assessing the quality of the solutions. The fuzzy fitness evaluation method combines both the power consumption and resource wastage objectives into one objective to be maximized [48].

4.4. *Fuzzy Fitness Evaluation* [48]. The fuzzy rule used to evaluate the solution is given as follows [48].

“If solution has *low* power consumption and *small* resource wastage then it is a good solution.”

A solution with highest membership in the fuzzy sets of $\{low, small\}$ is chosen to be best quality solution. The limits of the fuzzy sets for the power consumption and resource wastage are determined as given by Sait et al. [48]. The lower limit of the power consumption is presumed to be the power consumed by the solution having the minimum number of servers [48]:

$$\begin{aligned} \text{powerconsumption}_{\text{lower}} &= \text{LB}P_j^{\text{idle}} + (P_j^{\text{busy}} - P_j^{\text{idle}}) \left(\sum_{i=1}^n R_{\text{cpu}}^i \right). \end{aligned} \quad (36)$$

LB is the theoretical lower bound given in (28).

The upper limit of the power consumption is the power consumed by the worst case solution where each VM occupies one server [48]:

$$\begin{aligned} \text{powerconsumption}_{\text{upper}} &= \text{VM}_{\text{count}} P_j^{\text{idle}} + (P_j^{\text{busy}} - P_j^{\text{idle}}) \left(\sum_{i=1}^n R_{\text{cpu}}^i \right). \end{aligned} \quad (37)$$

VM_{count} is the number of VMs. The lower limit of resource wastage is presumed to be the wastage which results when all VMs are placed in a single large server [48]:

$$\text{resourcewastage}_{\text{lower}} = \frac{\left| \sum_{i=1}^n R_{\text{cpu}}^i - \sum_{i=1}^n R_{\text{mem}}^i \right| + \varepsilon}{\sum_{i=1}^n R_{\text{cpu}}^i + \sum_{i=1}^n R_{\text{mem}}^i}. \quad (38)$$

The upper limit on resource wastage is the wastage resulting due to worst VM placement scenario, that is, one VM per server [48]:

$$\text{resourcewastage}_{\text{upper}} = \sum_{i=1}^n \left\{ \frac{\left| R_{\text{cpu}}^i - R_{\text{mem}}^i \right| + \varepsilon}{\left| R_{\text{cpu}}^i - R_{\text{mem}}^i \right|} \right\}. \quad (39)$$

Given a solution s , the membership functions of the solution s in the fuzzy set *low* and *small* are determined as follows [48].

The membership of the solution s in the set $\{low\}$ is denoted as

$$\begin{aligned} \mu_{\text{spower}} &= \frac{\text{powerconsumption}_{\text{upper}} - \text{powerconsumption}_s}{\text{powerconsumption}_{\text{upper}} - \text{powerconsumption}_{\text{lower}}}. \end{aligned} \quad (40)$$

TABLE 6: Comparison of the multiobjective firefly and fuzzy firefly technique with other algorithms for average power consumption and resource wastage of VM requirements with reference values as 25% and 45% and correlation coefficients as -0.754 and -0.755 .

Reference value	$\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 25\%$				$\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 45\%$			
Correlation coefficient	-0.754				-0.755			
Algorithm	Power (W)	Resource wastage	Fuzzy fitness (10^{-3})	CPU execution time (S)	Power (W)	Resource wastage	Fuzzy fitness (10^{-3})	CPU execution time (S)
Fuzzy firefly colony	20430	6.12	905	5.43	30701	11.12	911	6.21
Firefly colony	20645	7.32	889	5.41	30985	11.21	901	6.23
ACO	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

The membership of the solution s in the set $\{small\}$ is denoted as

$$\mu_{\text{swastage}} = \frac{\text{resourcewastage}_{\text{upper}} - \text{resourcewastage}_s}{\text{resourcewastage}_{\text{upper}} - \text{resourcewastage}_{\text{lower}}} \quad (41)$$

The weights of each objective are calculated as [49]

$$\overline{w}_{\text{swastage}} = \frac{\overline{\mu}_{\text{swastage}}}{\overline{\mu}_{\text{swastage}} + \overline{\mu}_{\text{spower}}}, \quad (42)$$

$$\overline{w}_{\text{spower}} = \frac{\overline{\mu}_{\text{spower}}}{\overline{\mu}_{\text{spower}} + \overline{\mu}_{\text{swastage}}},$$

where

$$\overline{\mu}_{\text{swastage}} = 1 - \mu_{\text{swastage}}, \quad (43)$$

$$\overline{\mu}_{\text{spower}} = 1 - \mu_{\text{spower}}.$$

Using the weights given in ((42)-(43)), the complement of the overall degree of membership of solution s in the fuzzy set $\{low, small\}$ is determined as given by Sait et al. [48]:

$$\overline{\mu}_s = (\overline{w}_{\text{swastage}})(\overline{\mu}_{\text{swastage}}) + (\overline{w}_{\text{spower}})(\overline{\mu}_{\text{spower}}). \quad (44)$$

At last, the fuzzy fitness evaluation of solution s is determined as [48]

$$\mu_s = 1 - \overline{\mu}_s. \quad (45)$$

The solution which has maximum fuzzy fitness evaluation is considered to be the best solution.

4.5. Computational Experimental Study. We present the experimental results of the multiobjective FCO and FFCO algorithm to solve the VM placement problem. The experimental set up is same as discussed in Section 3.6. The results of heuristics and metaheuristic approaches considered are recorded in Tables 6–10.

From the results shown in Figures 3–6, we could observe that among the methods considered our proposed firefly colony approach has minimum power consumption and resource wastage.

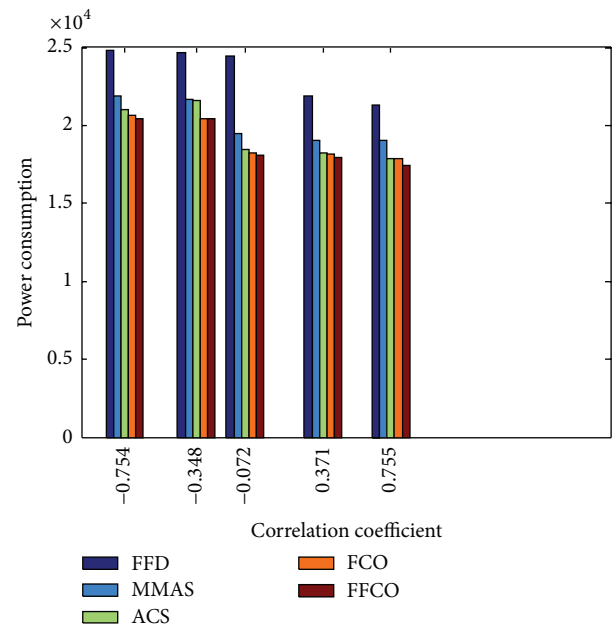


FIGURE 3: Power consumption of multiobjective techniques when $\overline{R}_{\text{cpu}} = \overline{R}_{\text{mem}} = 25\%$.

5. Conclusion

In this paper we propose to apply firefly colony and fuzzy firefly colony algorithms for server consolidation and virtual machine placement problem. The objective of server consolidation is to minimize the number of physical servers so as to minimize resource wastage. The multiobjective virtual machine placement problem aims to find an optimal solution of VM placement with minimum power consumption and resource wastage. Both the objectives are considered simultaneously during the solution construction process. The advantages of choosing firefly colony for the placement problem is that, unlike the standard firefly algorithms, the firefly colony approach is fast and decreases much of randomization during the search for optimal solution, thus leading to good performance. The fuzzy firefly colony approach is proposed with fuzzy transition probability rule to handle the exploring behaviour of fireflies with uncertainty. The obtained results

TABLE 7: Comparison of the multiobjective firefly and fuzzy firefly technique with other algorithms for average power consumption and resource wastage of VM requirements with reference values as 25% and 45% and correlation coefficients as -0.348 and -0.374 .

Reference value		$\overline{R}_{cpu} = \overline{R}_{mem} = 25\%$			$\overline{R}_{cpu} = \overline{R}_{mem} = 45\%$			
Correlation coefficient		-0.348			-0.374			
Algorithm	Power (W)	Resource wastage	Fuzzy fitness (10^{-3})	CPU execution time (S)	Power (W)	Resource wastage	Fuzzy fitness (10^{-3})	CPU execution time (S)
Fuzzy firefly colony	20402	5.29	908	5.25	30554	10.87	918	6.20
Firefly colony	20460	5.82	895	5.41	30786	10.92	906	6.19
ACO	21586	6.23	867	5.49	31175	8.12	838	6.89
MMAS	21643	6.15	859	5.73	31280	8.10	848	6.52
FFD	24680	21.78	721	8.12	34712	47.23	769	23.24

TABLE 8: Comparison of the multiobjective firefly and fuzzy firefly technique with other algorithms for average power consumption and resource wastage of VM requirements with reference values as 25% and 45% and correlation coefficients as -0.072 and -0.052 .

Reference value		$\overline{R}_{cpu} = \overline{R}_{mem} = 25\%$			$\overline{R}_{cpu} = \overline{R}_{mem} = 45\%$			
Correlation coefficient		-0.072			-0.052			
Algorithm	Power (W)	Resource wastage	Fuzzy fitness (10^{-3})	CPU execution time (S)	Power (W)	Resource wastage	Fuzzy fitness (10^{-3})	CPU execution time (S)
Fuzzy firefly colony	18105	4.11	911	5.23	28190	6.09	922	6.18
Firefly colony	18264	4.12	904	5.41	28436	6.12	918	6.17
ACO	18453	4.12	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 9: Comparison of the multiobjective firefly and fuzzy firefly technique with other algorithms for average power consumption and resource wastage of VM requirements with reference values as 25% and 45% and correlation coefficients as 0.371 and 0.398 .

Reference value		$\overline{R}_{cpu} = \overline{R}_{mem} = 25\%$			$\overline{R}_{cpu} = \overline{R}_{mem} = 45\%$			
Correlation coefficient		0.371			0.398			
Algorithm	Power (W)	Resource wastage	Fuzzy fitness (10^{-3})	CPU execution time (S)	Power (W)	Resource wastage	Fuzzy fitness (10^{-3})	CPU execution time (S)
Fuzzy firefly colony	17980	3.46	916	5.31	27980	4.86	934	6.25
Firefly colony	18200	3.58	908	5.35	28200	5.75	921	6.21
ACO	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 10: Comparison of the multiobjective firefly and fuzzy firefly technique with other algorithms for average power consumption and resource wastage of VM requirements with reference values as 25% and 45% and correlation coefficients as 0.755 and 0.751 .

Reference value		$\overline{R}_{cpu} = \overline{R}_{mem} = 25\%$			$\overline{R}_{cpu} = \overline{R}_{mem} = 45\%$			
Correlation coefficient		0.755			0.751			
Algorithm	Power (W)	Resource wastage	Fuzzy fitness (10^{-3})	CPU execution time (S)	Power (W)	Resource wastage	Fuzzy fitness (10^{-3})	CPU execution time (S)
Fuzzy firefly colony	17420	2.14	924	5.26	27083	3.14	938	6.24
Firefly colony	17860	2.42	912	5.28	27250	3.43	926	6.06
ACO	17880	2.71	897	5.31	27342	3.56	857	6.11
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

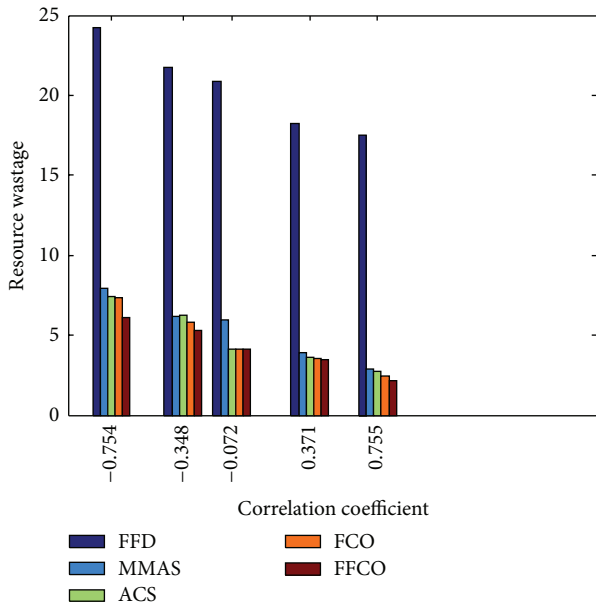


FIGURE 4: Resource wastage of multiobjective techniques when $\overline{R}_{cpu} = \overline{R}_{mem} = 25\%$.

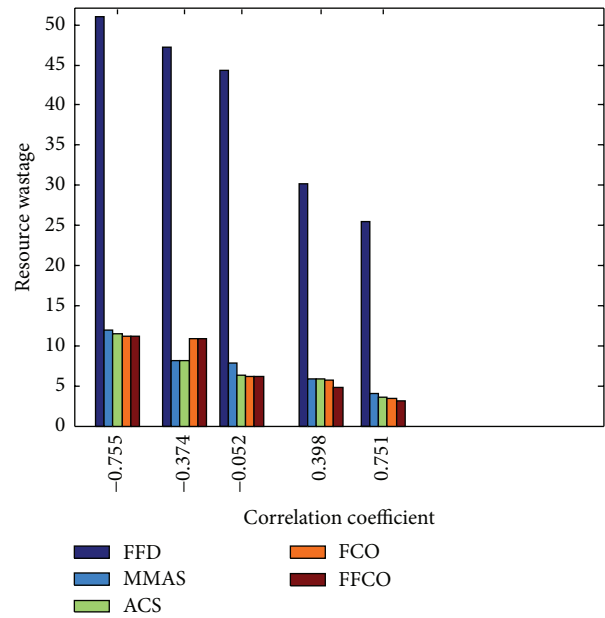


FIGURE 6: Resource wastage of multiobjective techniques when $\overline{R}_{cpu} = \overline{R}_{mem} = 45\%$.

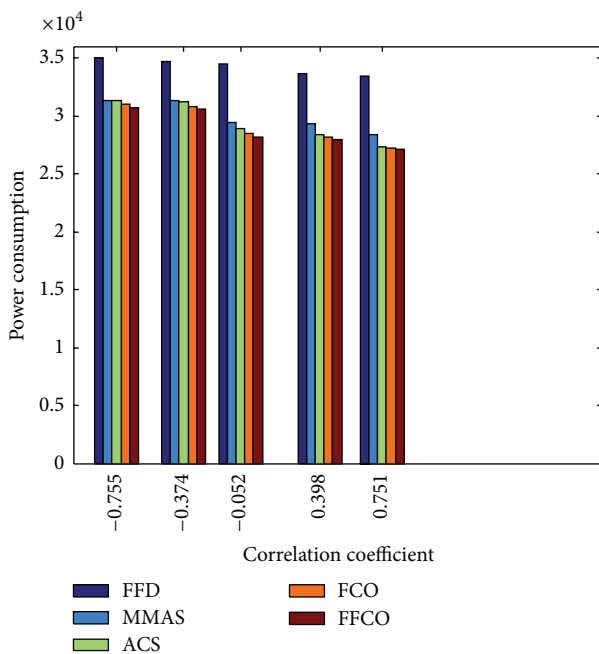


FIGURE 5: Power consumption of multiobjective techniques when $\overline{R}_{cpu} = \overline{R}_{mem} = 45\%$.

of firefly colony approaches are found to be better and encouraging compared to FFD deterministic heuristics and other few metaheuristic approaches considered.

Competing Interests


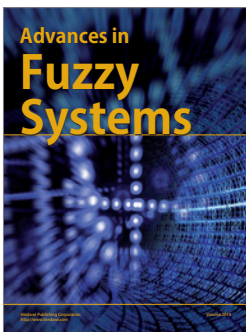
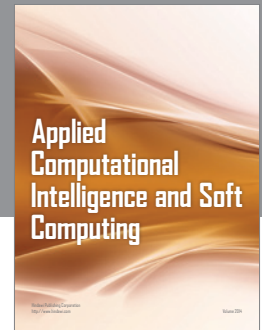
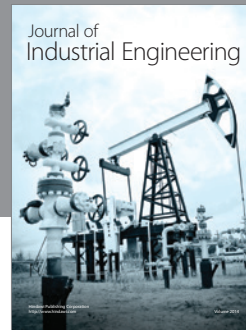
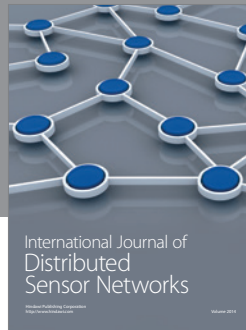
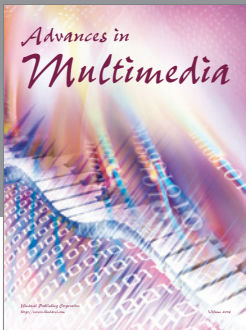
The authors declare that they have no competing interests.

References

- [1] P. Mell and T. Grance, "The nist definition of cloud computing (draft)," *NIST Special Publication*, vol. 800, no. 145, 2011.
- [2] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: a performance evaluation," in *Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings*, vol. 5931 of *Lecture Notes in Computer Science*, pp. 254-265, Springer, Berlin, Germany, 2009.
- [3] H. Chris, B. Mckee, R. Gardner, and B. J. Watson, "Autonomic virtual machine placement in the data center," HP Laboratories HPL-2007-189, 2008.
- [4] R. Gupta, S. K. Bose, S. Sundarajan, M. Chebiyam, and A. Chakrabarti, "A two stage heuristic algorithm for solving the server consolidation problem with item-item and bin-item incompatibility constraints," in *Proceedings of the IEEE International Conference on Services Computing (SCC '08)*, pp. 39-46, IEEE, Honolulu, Hawaii, USA, July 2008.
- [5] A. Anand, J. Lakshmi, and S. K. Nandy, "Virtual machine placement optimization supporting performance SLAs," in *Proceedings of the 5th IEEE International Conference on Cloud Computing Technology and Science (CloudCom '13)*, pp. 298-305, Bristol, UK, December 2013.
- [6] H. Jin, D. Pan, J. Xu, and N. Pissinou, "Efficient VM placement with multiple deterministic and stochastic resources in data centers," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM '12)*, pp. 2505-2510, IEEE, Anaheim, Calif, USA, December 2012.
- [7] J. Dong, H. Wang, X. Jin, Y. Li, P. Zhang, and S. Cheng, "Virtual machine placement for improving energy efficiency and network performance in IaaS cloud," in *Proceedings of the 33rd IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW '13)*, pp. 238-243, IEEE, Philadelphia, Pa, USA, July 2013.

- [8] S. Fang, R. Kanagavelu, B.-S. Lee, C. H. Foh, and K. M. M. Aung, "Power-efficient virtual machine placement and migration in data centers," in *Proceedings of the IEEE Green Computing and Communications (GreenCom '13), and IEEE International Conference on Internet of Things, and IEEE Cyber, Physical and Social Computing (iThings/CPSCoM '13)*, pp. 1408–1413, Beijing, China, August 2013.
- [9] T. Ferreto, C. A. F. De Rose, and H.-U. Heiss, "Maximum migration time guarantees in dynamic server consolidation for virtualized data centers," in *Euro-Par 2011 Parallel Processing*, vol. 6852 of *Lecture Notes in Computer Science*, pp. 443–454, Springer, Berlin, Germany, 2011.
- [10] D. Dong and J. Herbert, "Energy efficient VM placement supported by data analytic service," in *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid '13)*, pp. 648–655, Delft, The Netherlands, May 2013.
- [11] I. S. Moreno, R. Yang, J. Xu, and T. Wo, "Improved energy-efficiency in cloud datacenters with interference-aware virtual machine placement," in *Proceedings of the 11th IEEE International Symposium on Autonomous Decentralized Systems (ISADS '13)*, pp. 1–8, IEEE, Mexico City, Mexico, March 2013.
- [12] W. Shi and B. Hong, "Towards profitable virtual machine placement in the data center," in *Proceedings of the 4th IEEE/ACM International Conference on Cloud and Utility Computing (UCC '11)*, pp. 138–145, New South Wales, Australia, December 2011.
- [13] K. Maruyama, S. K. Chang, and D. T. Tang, "A general packing algorithm for multidimensional resource requirements," *International Journal of Computer and Information Sciences*, vol. 6, no. 2, pp. 131–149, 1977.
- [14] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder, "Heuristics for vector bin packing," research.microsoft.com, 2011.
- [15] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks*, vol. 53, no. 17, pp. 2923–2938, 2009.
- [16] M. Mishra and A. Sahoo, "On theory of vm placement: anomalies in existing methodologies and their mitigation using a novel vector based approach," in *Proceedings of the IEEE 4th International Conference on Cloud Computing (CLOUD '11)*, pp. 275–282, IEEE, Washington, DC, USA, July 2011.
- [17] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "EnaCloud: an energy-saving application live placement approach for cloud computing environments," in *Proceedings of the IEEE International Conference on Cloud Computing (CLOUD '09)*, pp. 17–24, IEEE, Bangalore, India, September 2009.
- [18] G. Jung, K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, and C. Pu, "Generating adaptation policies for multi-tier applications in consolidated server environments," in *Proceedings of the 5th International Conference on Autonomic Computing (ICAC '08)*, pp. 23–32, IEEE, Chicago, Ill, USA, June 2008.
- [19] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of Heuristics*, vol. 2, no. 1, pp. 5–30, 1996.
- [20] B. Brugger, K. F. Doerner, R. F. Hartl, and M. Reimann, "AntPacking—an ant colony optimization approach for the one-dimensional bin packing problem," in *Evolutionary Computation in Combinatorial Optimization*, vol. 3004 of *Lecture Notes in Computer Science*, pp. 41–50, Springer, Berlin, Germany, 2004.
- [21] P. Rohlfshagen and J. A. Bullinaria, "A genetic algorithm with exon shuffling crossover for hard bin packing problems," in *Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference (GECCO '07)*, pp. 1365–1371, London, UK, July 2007.
- [22] A. Shubham, S. K. Bose, and S. Sundarajan, "Grouping genetic algorithm for solving the serverconsolidation problem with conflicts," in *Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pp. 1–8, ACM, 2009.
- [23] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Proceedings of the IEEE/ACM Int'l Conference on Green Computing and Communications (GreenCom '10) & Int'l Conference on Cyber, Physical and Social Computing (CPSCoM '10)*, pp. 179–188, IEEE, Hangzhou, China, December 2010.
- [24] D. Wilcox, A. McNabb, and K. Seppi, "Solving virtual machine packing with a reordering grouping genetic algorithm," in *Proceedings of the IEEE Congress of Evolutionary Computation (CEC '11)*, pp. 362–369, IEEE, New Orleans, La, USA, June 2011.
- [25] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *Proceedings of the 12th IEEE/ACM International Conference on Grid Computing (Grid '11)*, pp. 26–33, Lyon, France, September 2011.
- [26] K. Mills, J. Filliben, and C. Dabrowski, "Comparing VM-placement algorithms for on-demand clouds," in *Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom '11)*, pp. 91–98, IEEE, Athens, Greece, December 2011.
- [27] Y. Wu, M. Tang, and W. Fraser, "A simulated annealing algorithm for energy efficient virtual machine placement," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC '12)*, pp. 1245–1250, IEEE, Seoul, South Korea, October 2012.
- [28] S. Luke, *Essentials of Metaheuristics*, Lulu, 2nd edition, 2013, <https://cs.gmu.edu/~sean/book/metaheuristics/>.
- [29] A. C. Adamuthe, R. M. Pandharpatte, and G. T. Thampi, "Multiobjective virtual machine placement in cloud environment," in *Proceedings of the International Conference on Cloud and Ubiquitous Computing and Emerging Technologies (CUBE '13)*, pp. 8–13, IEEE, Pune, India, November 2013.
- [30] S. Wang, Z. Liu, Z. Zheng, Q. Sun, and F. Yang, "Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers," in *Proceedings of the 19th IEEE International Conference on Parallel and Distributed Systems (ICPADS '13)*, pp. 102–109, IEEE, Seoul, South Korea, December 2013.
- [31] 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.
- [32] F. L. López Pires, E. Melgarejo, and B. Barán, "Virtual machine placement. A multi-objective approach," in *Proceedings of the 39th Latin American Computing Conference (CLEI '13)*, pp. 1–8, IEEE, Naiguata, Venezuela, October 2013.
- [33] M. A. Tawfeek, A. B. El-Sisi, A. E. Keshk, and F. A. Torkey, "Virtual machine placement based on ant colony optimization for minimizing resource wastage," in *Advanced Machine Learning Technologies and Applications*, pp. 153–164, Springer, Basel, Switzerland, 2014.
- [34] C. T. Joseph, K. B. Chandrasekaran, and R. Cyriaca, "A novel family genetic approach for virtual machine allocation," in *Proceedings of the International Conference on Information and Communication Technologies*, Kochi, India, December 2014.

- [35] M. Tang and S. Pan, "A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers," *Neural Processing Letters*, vol. 41, no. 2, pp. 211–221, 2015.
- [36] A. Layeb and Z. Benayad, "A novel firefly algorithm based ant colony optimization for solving combinatorial optimization problems," *International Journal of Computer Science and Applications*, vol. 11, no. 2, article 19, 2014.
- [37] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications*, vol. 5792 of *Lecture Notes in Computer Science*, pp. 169–178, Springer, Berlin, Germany, 2009.
- [38] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.
- [39] X. S. Yang and X. He, "Firefly algorithm: recent advances and applications," *International Journal of Swarm Intelligence*, vol. 1, no. 1, pp. 36–50, 2013.
- [40] Y. Gao, H. Guan, Z. Qi, and B. Wang, "An ant colony system algorithm for the problem of server consolidation in virtualized data centers," *Journal of Computational Information Systems*, vol. 8, no. 16, pp. 6631–6640, 2012.
- [41] A. A. Alsawy and H. A. Hefny, "Fuzzy-based ant colony optimization algorithm," in *Proceedings of the 2nd International Conference on Computer Technology and Development (ICCTD '10)*, pp. 530–534, IEEE, Cairo, Egypt, November 2010.
- [42] W. Elloumi, N. Baklouti, A. Abraham, and A. M. Alimi, "The multi-objective hybridization of particle swarm optimization and fuzzy ant colony optimization," *Journal of Intelligent & Fuzzy Systems. Applications in Engineering and Technology*, vol. 27, no. 1, pp. 515–525, 2014.
- [43] L. Gacogne and S. Sandri, "A study on ant colony systems with fuzzy pheromone dispersion," in *Proceedings of the International Conference on Information Processing and Management of Uncertainty (IPMU '08)*, vol. 8, p. 812, Malaga, Spain, 2008.
- [44] J. Van Ast, R. Babuška, and B. De Schutter, "Fuzzy ant colony optimization for optimal control," in *Proceedings of the American Control Conference (ACC '09)*, pp. 1003–1008, IEEE, St. Louis, Mo, USA, June 2009.
- [45] C.-W. Tao, J.-S. Taur, J.-T. Jeng, and W.-Y. Wang, "A novel fuzzy ant colony system for parameter determination of fuzzy controllers," *International Journal of Fuzzy Systems*, vol. 11, no. 4, pp. 298–307, 2009.
- [46] S. J. Narayanan, I. Paramasivam, R. B. Bhatt, and M. Khalid, "A study on the approximation of clustered data to parameterized family of fuzzy membership functions for the induction of fuzzy decision trees," *Cybernetics and Information Technologies*, vol. 15, no. 2, pp. 75–96, 2015.
- [47] R. B. Bhatt, S. J. Narayanan, I. Paramasivam, and M. Khalid, "Approximating fuzzy membership functions from clustered raw data," in *Proceedings of the Annual IEEE India Conference (INDICON '12)*, pp. 487–492, IEEE, Kochi, India, December 2012.
- [48] S. M. Sait, A. Bala, and A. H. El-Maleh, "Cuckoo search based resource optimization of datacenters," *Applied Intelligence*, vol. 44, no. 3, pp. 489–506, 2016.
- [49] J. A. Khan and S. M. Sait, "Fuzzy aggregating functions for multiobjective VLSI placement," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '02)*, vol. 2, pp. 831–836, IEEE, Honolulu, Hawaii, USA, May 2002.



Hindawi

Submit your manuscripts at
<http://www.hindawi.com>

