Multiagent and Grid Systems – An International Journal 12 (2016) 167–198
DOI 10.3233/MGS-160250
IOS Press

Energy-efficient virtual machine placement using enhanced firefly algorithm

Esha Barlaskar[a,*], Yumnam Jayanta Singh[a] and Biju Issac[b]
[a]*Department of Computer Science and Engineering and Information Technology, School of Technology, Assam Don Bosco University, Assam, India*
[b]*School of Computing, Teesside University, Middlesbrough, UK*

Received 23 February 2016
Accepted 7 July 2016

Abstract. The consolidation of the virtual machines (VMs) helps to optimise the usage of resources and hence reduces the energy consumption in a cloud data centre. VM placement plays an important part in the consolidation of the VMs. The researchers have developed various algorithms for VM placement considering the optimised energy consumption. However, these algorithms lack the use of exploitation mechanism efficiently. This paper addresses VM placement issues by proposing two meta-heuristic algorithms namely, the enhanced modified firefly algorithm (MFF) and the hierarchical cluster based modified firefly algorithm (HCMFF), presenting the comparative analysis relating to energy optimisation. The comparisons are made against the existing honeybee (HB) algorithm, honeybee cluster based technique (HCT) and the energy consumption results of all the participating algorithms confirm that the proposed HCMFF is more efficient than the other algorithms. The simulation study shows that HCMFF consumes 12% less energy than honeybee algorithm, 6% less than HCT algorithm and 2% less than original firefly. The usage of the appropriate algorithm can help in efficient usage of energy in cloud computing.

Keywords: Energy efficiency, virtual machine placement, hierarchical clustering, modified firefly algorithm

1. Introduction

The cloud being the fastest growing service providers impose increased the cost of maintenance and energy demand. To minimise the energy consumption in a cloud data center, the virtualization technology is considered [1]. Virtualization technology supports the data centers to run with fewer physical servers, optimising the usages of server and hence reduces the cost of the hardware and operation. However, it brings new challenges for the management of Virtual Machines (VMs), which must be provisioned and managed productively and hence, must pave the way for optimising the energy and performance trade-off. Proper allocation of VMs reduces the energy consumption and minimises the Service level agreements (SLAs). In clouds, dynamic VM consolidation is important since present-day service applications frequently experience variable workloads. When an application increases its demand, it results in an unexpected rise of the resource usage, which may lead to performance degradation if VM consolidation is not constrained. Many a time the application may encounter increased response

*Corresponding author: Esha Barlaskar, Department of Computer Science and Engineering and Information Technology, School of Technology, Assam Don Bosco University, Assam, India. E-mail: eshabarlaskar@gmail.com.

ISSN 1574-1702/16/$35.00

times, timeouts or failures if the application's resource requirement is not met. One of the important agreements in SLAs made between cloud providers, and their users are to provide quality of service. For meeting the quality of service in SLAs, the performance degradation is a major concern, which is further explained in this paper. The dynamic VM consolidation problem has four sub-problems:

(a) To determine when a host is considered as being overloaded (host overloading detection).
(b) To determine when a host is considered as being underloaded.
(c) To determine which VMs must be selected to migrate from overloaded host.
(d) To determine which hosts must be selected to place migrated VMs.

This work mainly focuses on Infrastructure-as-a-service (IaaS) environments in cloud data centres to provide an energy-efficient VM placement and quality of services inminimising the SLAs. It is essential to manage the heterogeneous mixed type of workloads since numerous distinctive users provision VMs in a dynamic fashion and dispose of diversified applications on shared physical resources. While the resource provider is oblivious and uninformed of the types of application that are deployed in the system and hence the system must be application skeptic, that is, must be capable of dealing with unknown mixed workloads effectively and efficiently. Another essential factor that needs to be handled is the quality of service guarantees, which are settled in the SLAs made between cloud providers and cloud consumers. Since numerous applications exist together in the system, therefore, it is essential to use an independent workload quality of service metric to measure the performance delivered to those applications. To establish system-wide quality of service, it is necessary to use such quality of service metric. IaaS only has been recognised as the most promising model, and it uses various virtualization technologies for instance Xen hypervisor [2], which efficiently manages the computing workload by assigning them in a proper manner. The problem of VM placement becomes crucial [3–5] as virtualization is the crux of cloud computing and the VM placement is usually pertaining to server consolidation [6]. Many metal-heuristic algorithms were used by different researchers in cloud computing.

Each of the afore-stated sub-problems must operate in an optimisedway, and this study tries to address the VM placement problem as it is necessary to manage the mapping of VMs to the appropriate physical machines (PMs) in the cloud data centres to avoid too many migrations that may lead to performance degradation. In order to perform the mapping of VMs correctly onto a PM, it is important to know the PM's capacity and whether it can fulfil the VMs resource demand without having resource conflicts, which aligns with the data center's policies. However, it is not only adequate to make good VM placement choices initially but also it is necessary to change the initial VM mapping in a dynamic way that is suitable for the changing conditions in the data center's VM load. To address the issue, this work proposes two meta-heuristic algorithms – (a) the modified firefly algorithm and (b) the Hierarchical Cluster based Modified Firefly algorithm (HCMFF). The performance of the proposed algorithms is evaluated by using CloudSim simulation toolkit and is compared with earlier work in [7]. Firefly algorithm is a meta-heuristic algorithm, which is used for optimisation problems. This gives an assurance of finding near-optimal solutions within a remarkable decline in the amount of time. Henceforth, the use of meta-heuristics is acquiring considerable attention. The sequence of the study is as follows:

(a) A comparison study between firefly and honeybee algorithms: The firefly algorithm gives a better result because it has the following advantages: (i) automatic subdivision of the whole population into subgroups (ii) the natural capability of dealing with multi-modal optimisation (iii) high ergodicity and diversity in the solutions. All these advantages make firefly algorithm unique and very efficient. The details impact of the all participating parameters is also shown.
(b) Comparison of Honeybee Cluster based Technique (HCT) and hierarchical cluster based modified firefly algorithm (HCMFF): The HCMFF gives a better result as the searching time of the most

appropriate PM for placing a particular VM is reduced, and it has been observed that by combining hierarchical clustering with firefly algorithm the total number of VM migrations had been reduced to a great extent. This is because the VMs will be sent to a specific cluster of PMs (which can provide the amount of resource required by the VMs) instead of sending the VMs randomly. Thus the advantages of firefly along with that of the hierarchical clustering show a nearly optimal result.

(c) An overall comparison between all participating algorithms: The overall results of all four algorithms are analysed. The HCMFF gives better than the entire participating algorithm. The results show that both modified firefly algorithm and HCMFF algorithm reduces energy consumption and some SLA violations.

The HCMFF performed better than other algorithms because it is competent in finding the best cluster among the different clusters of PMs that will be most capable and efficient for any VM placement. Firefly algorithm is swarm-intelligence-based, so it has the same type of advantages that other swarm intelligence-based algorithms have. However, firefly algorithm has two prime benefits over other algorithms: automatic subdivision and the ability to deal with multimodality. First, firefly algorithm is based on attraction and attractiveness decreases with distance. This leads to the fact that the whole population can automatically subdivide into subgroups, and each group can swarm around each mode or local optimum. Among all these modes, the best global solution can be found. Second, this subdivision allows the fireflies to be able to find all optima simultaneously if the population size is substantially higher than the number of modes. This automatic subdivision ability makes it particularly suitable for highly nonlinear, multimodal optimisation problems. All these advantages of firefly algorithm make it even more efficient when combined with hierarchical clustering algorithm's merits stated above, thereby outperforming all the algorithms that HCMFF is compared with.

The rest of the paper is organised as follows. Section 2 presents the related work. Section 3 provides an introduction and explanation of the proposed algorithms. Section 4 shows the experimental result. Finally, Section 5 provides the summary of the study and concludes the paper

2. Literature review

The cloud computing provides resources based on SLA created through negotiation between the service provider and users [8]. It is necessary to minimise energy consumption and thus it is very difficult to maintain the trade-off between energy and performance. To overcome this problem, many researchers proposed different methods. Bobroff et al. [9] proposed a new algorithm for preserving performance. Their algorithm remaps the VM to PM for future resource demand. Barbagallo et al. [10] described a bio-inspired algorithm hinged on the scout-worker migration method where some of the scouts are proposed to move from one physical node to another so that they can cooperatively find a suitable destination for the migrated VMs.

Metaheuristic algorithms have been widely studied for VM placement in the literature [11]. The study [12] shows performance of various Swarm Intelligence (SI) approaches including Genetic algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Differential Evolution (DE), Artificial Bee Colony (ABC), Glow-worm Swarm Optimization (GSO), and Cuckoo Search Algorithm (CSA). Such algorithms are widely used for solving the problem of VM placement, along with the Genetic Algorithm (GA), Honeybee algorithm (HB), Ant Colony Optimization algorithm (ACO) as listed in [13].

2.1. Ant colony algorithm

The study [14] proposes an approach based on ant colony algorithm to effectively balance power consumption among nodes. However, they have only focused on overload, under load and idle host detection but didn't consider VM-migration in their work. Several studies hardly consider historical data and system fluctuations which lead to load inequality of the system. In [15] a multi-objective ant colony system algorithm was proposed for the VM placement with the aim of obtaining a group of non-dominated solutions that manages the trade-off between resource wastage and power consumption. The authors compared the proposed algorithm with multi-objective GA, two single-objective algorithms namely bin packing, and MMAS; the outcome of the experiment proved that the proposed algorithm is much efficient than the algorithm it was compared to. The authors [16,17] explain the placement problem based on the proxy method. Ant colony optimisation can be used to solve multi-objective optimisation problems to optimise total processing resource wastage and memory resource wastage [18]. This work focused only on the performance.

2.2. Genetic algorithm

Hu et al. [19] proposed a scheduling strategy of resources based on a genetic algorithm which considers historical data and the current state of the system and therefore estimates in advance the influence it will have on the system. Hence this strategy solves the problem of load imbalance and huge migration cost. Falkenauer [20] proposed an enhanced approach of a genetic algorithm to handle the server consolidation problem using the group-based encoding scheme. In [21] Savant proposed genetic algorithm as a scheduling strategy for load balancing of VM resources. The VM resource scheduling strategy focuses on system load balancing. The study of [22] uses the GA approach to find the effect of the deployment of new VM resources in the system. The author proved that the traditional algorithm, when used for resource scheduling, ends up in an imbalance of load and the number of VM migration also increases. In [23] another GA-based approach (GABA) was proposed which could self-reconfigure the VMs in cloud data centres consisting of heterogeneous PMs. While in [24] the VM placement problem is designed as a multi-objective optimisation problem to minimise various issues such as power consumption, resource wastage and the cost of thermal dissipation. To tackle all these issues, the authors proposed an optimal GA with fuzzy multi-objective evaluation.

2.3. Firefly algorithm

In the recent years, the firefly algorithm research work has multiplied considerably. Faster et al. presented an extensive and abridged review [25] on firefly algorithm. Some variants of firefly algorithm were proposed and implemented in various fields, for example, the authors in studies [26–30] designed discrete form of firefly algorithm for tackling the combinatorial optimisation problems and discrete problems. This discrete pattern of firefly algorithm can be used in a variety of applications, for instance, graph colouring, travelling-salesman problems (TSP), etc. In [30] a discrete form of firefly algorithm was proposed for solving the scheduling problems. In addition to that the authors [29,31,32] demonstrated that the problem of scheduling and travelling-salesman could be solved in a much progressive manner. In [33,34] firefly algorithm was applied in solving the problems of clustering and classification and firefly algorithm gave an excellent result. In [35] firefly algorithm has also been applied in the training of neural network. Eventually from [36–38] it was demonstrated that for any kind of optimisation problems that are dynamic in nature, firefly algorithm has always proved to be quite efficient. A multi-swarm based firefly algorithm is used in dynamic environments.

2.4. Honeybee and ant colony algorithm

In [39] the authors proposed eco-friendly algorithm by combining both honeybee and ant colony algorithm for cloud computing which reduced the operational cost by minimising power consumption which also diminished global warming to a great extent. The proposed Bee-Ants colony system was used for proper energy efficient resource management where initially the jobs are divided into two parts; the first part which looks after the proper management of overloaded. The underloaded CPUs with service rescheduling was carried out by honeybee algorithm. The second part, which helps to manage the idle CPUs (power consumption management) is achieved by ant colony algorithm.

2.5. Particle swarm algorithms

Particle Swarm algorithms are used for efficient VM allocation to physical servers to reduce the total resource wastage and a number of servers used [40]. An improved particle swarm optimisation approach for virtual machine placement is proposed by Wang et al. [41]. The immune algorithm is also used for energy optimisation in cloud computing [42,43]. The Glowworm swarm optimisation algorithm uses features with some better-known swarm intelligence based optimisation algorithms [44]. A comparative study on firefly algorithm, particle swarm optimization, is shown in several studies [25,34,45].

2.6. Existing virtual machine placement techniques

VM placement is crucial for better resource utilisation and energy efficiency in cloud computing infrastructures. Various research work has pontificated the significance of the VM placement problem relevantly, for instance, Cardoso et al. [46] described the importance of placing VM into PM appropriately. In [47] the authors proposed a Power Aware Best Fit Decreasing (PABFD) algorithm for VM placement that is a modification of Best Fit Decreasing algorithm (BFD). The authors [9,48,49] have also formulated numerous heuristics for VM placement problem. In the study [50] the authors dealt with the trade-off between cost and power dependent on tight performance constraint by packing as many VMs in a small number of physical machines and this reduced the cost of VM migration. While the author [51] designed a single-objective algorithm based on max-min ant system (MMAS) metaheuristic to reduce the total amount of PMs needed to handle the currently available load.

The authors [52] proposed an efficient algorithm established in linear and quadratic programming for making the placement of VMs on PMs optimum and the main aim of this work is to minimise the usage of the total number of nodes. The server consolidation problems were solved with the formulations of linear programming [53,54] where the authors created extended restrictions for the problem of VM allocation. The restriction was that the VMs allocated to a PM should be based on some unique attribute so that the total number of VM migrations can be minimised and also a heuristic based on linear programming (LP)-relaxation was built to optimise the linear programming solving cost. The authors [55] addressed the problem of VM provisioning and placement as two constraint satisfaction problem and they proposed a framework for resource management by combining dynamic VM provisioning manager and VM placement manager which are utility based. On the other hand, the authors [56] solved the constraint programming based dynamic consolidation problem by designing an Entropy resource manager for similar clusters that considers both the issues of VM allocation and VM migration to the available nodes.

Virtualization technology also tried to minimise the consumed energy [57]. These efforts started in the study [58] in which it was mentioned that the scintillating features of virtualization technology such as

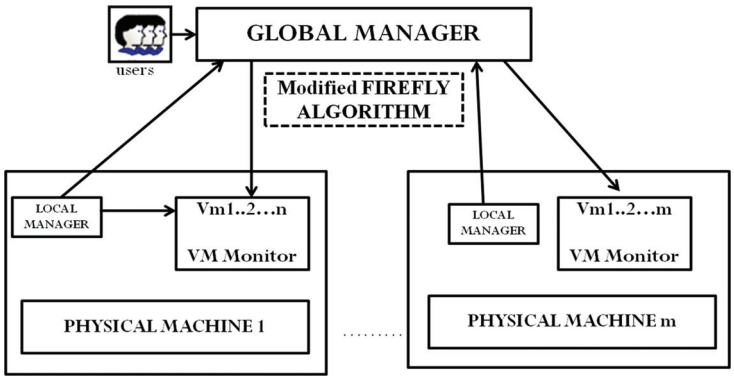

Fig. 1. System model with the proposed firefly algorithm.

migration could be used to cause systems to be power-aware. The nature-inspired honeybee algorithm is used in solving the dynamic VM placement problem [7]. They have tackled the problem of power efficient resource management in virtualized data centers to maximise the cloud provider's profit by minimising both power consumption and SLA violation. Bouras et al. [59] defined a framework showing the effort to capture all the technical parameters entailed in provisioning a service with qualitative guarantees. Addis et al. [60] proposed a unifying framework that provides very efficient and robust solutions at multiple time-scales. Sharifi et al. [61] consider energy efficiency along with performance. They showed that automatic consolidation of VMs does not minimise the power consumption of data centers but it can also cause energy wastage. They then proposed a scheduling algorithm that was energy-aware using a group of objective functions regarding fitness consolidation metric and was much better when compared to other scheduling algorithms.

However, these algorithms do not use exploitation mechanism efficiently. This study uses hierarchical clustering method for the VM placement. This helps in finding the best cluster among the different clusters by minimising the energy usage level.

3. Proposed method

This study mainly considers Infrastructure-as-a-Service (IaaS) as it has been recognised as the most promising model. IaaS is represented by a large-scale data centre comprising of a large number of the heterogeneous physical node where each node is characterised by CPU performance, disk storage, the amount of RAM and network bandwidth [8]. The system model with the proposed firefly algorithm is presented in Fig. 1.

The task from the users is accepted by the Global Manager. The software layer of the system is tiered comprising of local and global managers. All the local managers maintain the list or indexes of PMs in a particular cluster for other clusters. When a new VM instance request is sent to the global manager, it takes the updates of the available resources from all the local managers of each cluster. Thus it maps the VM to the most appropriate cluster of PM. The VM monitor (VMM) maintains seclusion at all times between VMs by managing and multiplexing the physical resources access. Each of the VM is self-supporting with its operating system because of the virtualization of the physical resources and hence numerous VMs can be executed on the single physical machine (PM). The separation between physical and virtual resources provided by the VMM allows elasticity of resource provisioning for VMs. As a PM,

a VM too has resources such as CPU, memory, and input/output (I/O) devices associated with it and these resources needs to be provisioned to each of the VMs while doing their instantiation. The responsibility of the VMM is to multiplex the resources across VMs as these resources can be overcommitted. To determine the initial levels for resource provisioning of a VM "sizing process" is used which depends on applications resource usage profiles or assessment to fulfil the load demand and other processes. This architecture is supported by the firefly algorithm.

Every VM have various kinds of loads, and as these loads keep on increasing with time, the upper threshold value of a PM will be reached or crossed resulting in the imbalance of load in the system. To avoid such a situation, proper VM allocation must be done to enhance the resource utilisation and consequently improve the overall performance of the cloud data centres. VM placement or allocation problem is also known as VM instance scheduling. Any algorithm is considered profitable if it efficiently allocates a large number of VMs to very few PMs and also avoids the overutilization of PMs which often increases the number of VM migrations. In VM placement problem it may not be possible to get the best placement results within polynomial time. However, the meta-heuristic algorithms can get near optimal solutions, if not the best. Due to this reason, the study chooses firefly algorithm for VM placement that can use exploitation mechanism efficiently. VM migration takes place when a PM is overloaded, and by shifting few VMs, the resource utilisation of that particular PM can be minimised. Also, if a PM is not fully utilised, then the resource will be wasted. Thus by migrating VMs from under loaded PMs the resource wastage, as well as energy consumption, can be reduced. However, if the total number of VM migration increases then it will also increase the SLA violation. The increase in SLA violation will result in performance degradation.

Therefore the cloud providers will benefit a lot if they group the PMs based on their ability to manage different kinds of VM instances. For example, if a VM instance is too large then it would be better to allocate this particular VM to a PM which will be capable of handling such large instance instead of allocating it randomly. Again for this purpose, the study used the concept of hierarchical clustering algorithm so that it can minimise the time required to search the best PM while performing VM migration. It can easily find the best cluster among the different clusters of PMs that will be most capable and efficient for any VM placement. The dynamic VM consolidation problem is divided into four sub-problems: (a) Checking whether the host is under loaded; (b) Checking whether the host is overloaded; (c) Selection policy to migrate VMs from the overloaded host; and (d) VMs placement for placing the VMs in allocation or migration to another host [47]. Among all the mentioned sub-problems, we are focused more on the VM placement.

3.1. Problem formulation of VM placement

Assuming, a set of VMs denoted by VM $= \{vm_1, vm_2, \ldots, vm_n\}$ where each of the vm_i is a trinity represented as $vm_i = (cpu_i, ram_i, bw_i)$, $1 \leqslant i \leqslant n$, the values of the triplets denotes CPU, memory and bandwidth demands of VMs respectively. Let PM $= \{pm_1, pm_2, \ldots, pm_m\}$ denote a set of PMs and each of the pm_j is also a trio represented as $pm_j = (cpu_j, ram_j, bw_j)$, $1 \leqslant j \leqslant m$, the values of the triplets denote the total resource capacity of the j^{th} PM. In addition, x_{ij}, $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$ and y_i, $1 \leqslant i \leqslant m$ are decision variables, $x_{ij} = 1$ if and only if vm_j is mapped onto pm_i, $y_i = 1$ if pm_i is used to host virtual machine. The objective function is to minimise $\sum_{i=1}^{m} y_i$ while discovering all values of x_{ij}. The absolute restrictions constraints in the above stated description is that each of the VM can be allocated on only one physical machine at a time. The details of constraints are also referred from the study of [62]. For each type of resources (CPU, memory and bandwidth, the quantity of resource requests of VMs) placed in the same physical machine must be less or equal to ability/capacity of the PMs hosting them; The total numbers of PMs that allocate VMs [47] are not more than m, $\sum_{i=1}^{m} y_i \leqslant m$.

3.2. Assumptions of the firefly algorithm

For VM placementthis study proposes energy efficient modified firefly algorithm. This algorithm is based on the demeanour of different species of fireflies that generate terse and cadent flashes. Most of the time the pattern observed for the flashes is exclusive and distinct for every particular species of fireflies, for instance, the cadent of the flashes, the rate of flashing and the total time for which the flashes are noticed. Each and every of these patterns collectively composes a kind of pattern that attracts both male and female fireflies to each other and thus the female of a species reunite to a distinctive pattern of the male of the same species. At a certain distance 'r', the intensity of light from the light source conforms to the inverse square law [45]. That is, as 'r' increases the intensity of light 'I' will decrease and is given in terms of $I \propto 1/r^2$. Furthermore, the air or medium keeps on enthralling the light and as a result of which the light becomes feeble as the distance increases. Thus when these two factors, namely the intensity of the light and air absorption or enthralling are combined, it makes most fireflies seeable at a narrow distance, usually to a few hundred meters at night which are pretty sufficient for fireflies to confer and communicate with each other.

Yang developed firefly algorithm in late 2007 and 2008 [63,64]. The firefly algorithm was inspired by the flashing motif and action of fireflies. It uses the following three rules (or rather assumptions).

(a) It is considered that one firefly is captivated to other fireflies regardless of their sex. That means all fireflies are unisex.
(b) It states that for any two fireflies that are flashing, the brighter one will be attracted to less bright one and less bright to the brighter one. The brightness and attractiveness are proportional to each other, and both will decrease when their distance increases. However, a firefly will move randomly if there is no one brighter than that particular firefly.
(c) The objective function is used to determine the brightness of firefly [45]. The brightness is directly proportional to the objective function's value for all maximisation problems. Other forms of the brightness have a function as used in genetic algorithms [65].

To refit the firefly algorithm concept to VM placement problem the proper translation of terminology used in the firefly algorithm must be done efficiently and this terminology is the crucial factor in a combinatorial space from a continuous one. VM placement is one of the combinatorial optimisation problems and as such the key concepts related to firefly algorithm (which is the above three assumptions) must be described by VM placement problem before solving this problem. The basic firefly algorithm assumes that all fireflies are unisex, and the main terms described are brightness and attractiveness of fireflies. These assumptions are modified in order to relate it to VM placement problem and as such the following three assumptions are made due to the fact that they are VMs which need to be properly allocated on PMs depending on the availability of resources. To align firefly algorithm to VM placement, the three crucial assumptions need to be redefined and discussed. From the three assumptions made in the basic firefly algorithm, the study uses that the firefly flashing behaviour by modifying the concept in VM placement methods, which are given as follows.

Assumption 1. It is assumed that all the fireflies are not unisex, which implies that VM's are female fireflies and PM's are male fireflies. The female fireflies will be attracted to male fireflies depending on the brightness of the male firefly and their brightness. For PMs, the brightness is more if PM is not overloaded or slightly loaded and brightness is less if PM is overloaded or going to be overloaded very soon. For VMs, the brightness is more if the VM is not overloaded or slightly loaded. Brightness is less if VM is overloaded. That is, less bright VM will be placed on those PMs which are brighter and bright VM will be placed on less bright PMs.

Assumption 2. Attractiveness and brightness are proportional to each other. For any two male flashing fireflies (the PMs), the less bright female firefly (VMs) will move towards the one which is brighter PM than the less bright PM. Attractiveness and brightness both decrease as their distance increases (that is, distance increases when the resource utilisation of both PM and VM increases).

Assumption 3. The brightness of male and female fireflies are determined by the view of the objective function which is, in our case, the resource utilisation of the PMs and VMs. The more the resource utilisation, the less will be the brightness. The less the resource utilisation the more will be the brightness. However, in the case of PMs the threshold values are set, and for values below and above the brightness decreases. That is if a PM is underutilised and is below the lower threshold then also the PM becomes less bright. If a PM is over-utilized and is above the upper threshold, then the PM becomes less bright. Therefore a PM is brighter only when its resource utilisation is in between the lower and upper threshold.

From these three assumptions, the concept of firefly algorithm was clearly depicted and presented in accordance to VM placement problem. The formal definition of "brightness" in this context is explained below. For PM the brightness is defined by the amount of resource available at the time of VM placement. That is, if a PM is said to be brighter than another PM then it means the resource provided by the first PM is more compared to the second PM. Hence the first PM is not overloaded or less loaded. For VM the brightness is defined as the amount of resource needed by a VM while placing that VM in a PM. If more resources are requested by a particular VM, then it shows less brightness.

3.3. Proposed modified firefly algorithm (MFF) for VM placement

As discussed in the theory this proposed modified firefly algorithm (MFF) considers that fireflies are not unisex. The fireflies are males and females belonging to a different variety of species. The variation in the light intensity and formulation of the attractiveness are the two important factors in the firefly algorithm [45,63–65]. For simplicity, it is assumed that the attractiveness of a firefly is determined by its brightness which in turn is connected with the encoded objective function.

$$x_j = PEnum_j \times PEmips_j + VMbw_j \tag{1}$$

where $PEnum_j$ is the number of processor in VM_j; $PEmips_j$ is a million instructions per second of all processors in VM_j; $VMbw_j$ is the bandwidth and communication ability of VM_j. The brightness 'I' of any VM could be chosen as $I(x_j)$ proportional to $f(x_j)$ where $f(x_j)$ is the current resource utilisation by that particular VM_j. In the case of female firefly the brightness increases if the resource utilisation increases. The location x of a PM is the capacity of any PM 'i' which is given by:

$$x_i = PEnum_i \times PEmips_i + PMbw_i \tag{2}$$

where $PEnum_i$ is the number processor in PM_i; $PEmips_i$ is a million instructions per second of all processors in PM_i; PM bw_i is the bandwidth communication ability of PM_i. The brightness 'I' of any PM could be chosen as $I(x_i)$ proportional to $f(x_i)$ where $f(x_i)$ is the current resource utilisation by that particular PM_i. In the case of male firefly the brightness increases if the resource utilisation increases. Although the attractiveness (β denotes attractiveness) is relative, it should be determined by the other fireflies, specifically with the brightness of each male and female fireflies. Thus it will vary with the distance r_{ij} between male firefly i and female firefly j. The distance r_{ij} is determined by the difference in resource utilisation of male firefly (i.e. PM) and the female firefly (i.e. VM). The distance between them will be different if the resource utilisation of male firefly is morethefemale firefly. Inversely more

the resource utilisation of male firefly and more the resource utilisation of female the distance will be more. Also, light intensity decreases with the distance from its source. If the difference between the resource utilisation of male and female fireflies is more, then there is less possibility of placing a VM in a PM. The air media also absorb light. In this study the underutilization of resource usage in PMs is denoted as the absorption co-efficient. It should allow the attractiveness to differ with the varying degree of absorption. However, the light intensity or attractiveness value β depends on the distance r between the fireflies and the media light absorption coefficient γ. The attractiveness of each firefly is determined using the equation below:

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

where β_0 represents the attractiveness of the firefly at $r = 0$. The movement of the less bright female firefly j is attracted to another more attractive (brighter) male firefly i is determined by

$$x_i = x_i + \beta_0 e^{-\gamma r}{}_{i,j}^{2}(x_j - x_i) + \alpha\varepsilon_i \tag{4}$$

where the second term is rise due to the attraction and third term is randomization with α being the randomization parameter, and ε_i is a vector of random numbers taken from a Gaussian or uniform distribution. The parameter γ now represents the variation of the attractiveness, and its value is critically necessary for deciding the speed of the convergence and how the firefly algorithm behaves. In theory, $\gamma \in (0, \infty)$, but in areal application, $\gamma = O(1)$ is determined by the characteristic distance $r(= \Gamma = 1/\gamma)$ of the system to be optimised. Thus for most applications, it conventionally varies from 0.1 to 10.

The pseudo code of proposed modified firefly algorithm (MFF) is provided as Algorithm 1.

Algorithm 1: Modified firefly algorithm

MFF Meta-heuristic ( )

1. Begin;
2. Initialize algorithm parameters:
   MaxGen: The maximal number of generations
   γ: The coefficient of light absorption
   r: The specific distance from the light source
   d: The realm space
3. Characterize the objective function of $f(x)$, where $x = (x_1, \ldots, x_d)$
4. Produce the introductory population of fireflies or x_i (for $i = 1$ to n)
5. Evaluate the intensity of light I_i at x_i via $f(x_i)$
6. While ($t <$ *MaxGen*)
7. For $i = 1$ to n (all n male fireflies);
8. For $j = 1$ to m (m female fireflies);
9. If ($I_j > I_i$)
10. Move firefly i towards j by using Eq. (4);
11. End if
12. Attractiveness varies with distance r via $\mathrm{Exp}[-\gamma r^2]$;
13. Evaluate new solutions and update light intensity;
14. End for j;
15. End for i;
16. Rank the fireflies and find the current best;
17. End while;
18. Post process results and visualisation;
19. End procedure

3.3.1. Hierarchical clustering method

In combination with the proposed firefly algorithm, we want to use the clustering method to reduce the time while migrating the VMs to find the best cluster for virtual machine placement. The process of grouping or partitioning data based on some similitude is known as clustering.

Clustering algorithms are of two types, namely, hard clustering and soft clustering. Hierarchical clustering is where a nested series of the division is created, and the partitioned clustering is used with a segregation of given data. They fall under the type of hard clustering. Whereas rough sets, fuzzy sets, evolutional algorithms or artificial neural networks (ANNs), and particularly genetic algorithms (GAs) are soft clustering algorithms. In this proposed work, the hierarchical clustering algorithm is used. Hierarchical algorithms produce a nested series of divisions of the data that can be interpreted by using a tree structure that is commonly called as a dendrogram. Hierarchical algorithms are of two types, namely, divisive and agglomerative. The divisive clustering starts with one cluster with all the patterns and at each consecutive step a cluster is divided; this method goes on till it finishes up with each pattern in a cluster or a group of clusters with exactly one pattern. A top-down approach is used by the divisive algorithm for creating divisions of the data. In divisive algorithms when two patterns are put into two distinct clusters at any step, then at all the consecutive steps they remain in distinct clusters. To the contrary, agglomerative algorithms use a bottom-up approach where starting with n single clusters when the size of the input dataset is n and each pattern of the input data set is in a distinct cluster. At each subsequent steps, the most matching pair of clusters is joined to decrease the size of the division by one.

The proposed system model of Hierarchical Cluster based Modified Firefly algorithm (HCMFF) is as follows. The algorithm is shown in Fig. 2. The execution process of the firefly algorithm is designed using the concept of the hierarchical cluster. The related literature and ideas are collected from earlier studies [47].

A significant characteristic of the agglomerative algorithms is that once the two patterns are put in the same cluster at a step, then they remain in the same cluster at all the consecutive steps. Agglomerative clustering follows a bottom-up approach [66,67]. This work follows the agglomerative clustering algorithm for making clusters of PMs based on the type of resources provided by the PMs. In this study the total number of input data sets is equal to the total number of PMs which is 'm', the similarity between the patterns is equal to the type or characteristic of resources provided by the PMs. For instance, to form the clusters it considered few parameters like CPU utilisation, bandwidth speed etc. There are total 800 PMs with the characteristics similar to the servers considered in [47]. The purpose of forming clusters of PMs is to minimise the time taken while migrating a VM by reducing the searching procedure of most capable PMs.

The hierarchical clustering algorithm with bottom-up approach is described as follows: $D = [d(i, j)]$ is the $N * N$ adjacency matrix. All the clusters are assigned series numbers $0, 1, \ldots, (n - 1)$ and $L(k)$ is the level of the k^{th} cluster. The adjacency between clusters (r) and (s) is denoted $d[(r), (s)]$ and a cluster with series number m is denoted by (m).

Step 1. Begin with the disjoint clustering having level $L(0) = 0$ and sequence number $m = 0$.

Step 2. Find the least dissimilar pair of clusters in the current clustering, say pair (r), (s), according to $d[(r), (s)] = \min d[(i), (j)]$, where the minimum is over all pairs of clusters in the current clustering.

Step 3. Increment the sequence number: $m = m + 1$. Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to $L(m) = d[(r), (s)]$.

Step 4. Update the proximity matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The

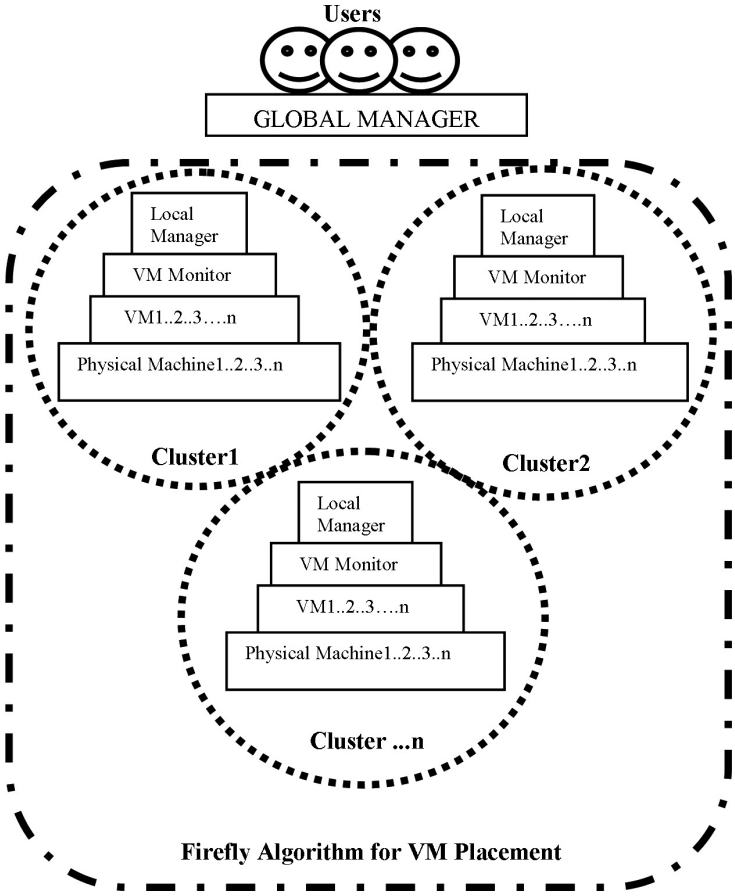

Fig. 2. System model with the proposed HCMFF algorithm.

proximity between the new cluster, denoted (r, s) and old cluster (k) is defined in this way: $d[(k), (r, s)] = \min d[(k), (r)], d[(k), (s)]$.

Step 5. If all objects are in one cluster, stop or else, go to Step 2.

The pseudo code of the hierarchical cluster based modified firefly (HCMFF) is provided below as Algorithm 2.

Algorithm 2. Hierarchical cluster based modified firefly (*HCMFF*)

Step 1. Resources are clustered as a combination of RESOURCE, BANDWIDTH and MEMORY by using hierarchical clustering.

Step 2. Each cluster is deliberated as a single resource.

Step 3. VMs are classified by different types of requirements such as small instances and large instances.

Step 4. Initialize firefly parameters.

MaxGen: Maximal number of generations (total number of 'n' VMs and 'm' PMs).

γ: The light absorption coefficient, which means that the brightness decreases if the distance between resource utilization of PM and VM is more and also it decreases if the resource utilization of PM is under the lower threshold.

r: The particular distance from the light source.
d: The domain space i.e. the total number of clusters formed.

Step 5. Define the objective function of $f(x)$, where $x = PEnum \times PEmips + VMbw/PMbw_j$
where, *PEnum* is the number processor in VM/PM,
PEmips is a million instructions per second of all processors in VM/PM
$VMbw_j/PMbw_j$ is the bandwidth communication ability of VM_j

Step 6. Generate the initial population of fireflies i.e. number of PMs $= 1$ to n and Number of VMs $= 1$ to M
Let $PM = \{PM_1, PM_2, \ldots, PM_n\}$ and $VM = \{VM_1, VM_2, \ldots, VM_m\}$

Step 7. Determine the light intensity of I_i or I_j at x_i or x_j via $f(x_i)$ or $f(x_j)$
The brightness 'I_i' of any PM could be chosen as $I(x_i)$ proportional to $f(x_i)$ where $f(x_i)$ is the current resource utilisation by that particular PM_i. In the case of male firefly, the brightness increases if the resource utilisation decreases.
The brightness 'I_j' of any VM could be chosen as $I(x_j)$ proportional to $f(x_j)$ where $f(x_j)$ is the current resource utilisation by that particular VM_j. In the case of female firefly, the brightness increases if the resource utilisation decreases.

Step 8. While ($t <$ *MaxGen*)
For $i = 1$ to m (all m male fireflies);
For $j = 1$ to n (n female fireflies)
if ($I_j > I_i$)
move firefly i towards j by using Eq. (4);
end if
Attractiveness varies with distance r via $\mathrm{Exp}[-\gamma r2]$;
Evaluate new solutions and update light intensity;
End for j;
End for i;
Rank the fireflies and find the current best;
End while;

Step 9. Post process results and visualisation;

Step 10. End procedure

A flowchart for general firefly algorithm for VM placement is given in Fig. 3.

3.3.2. Advantage of using hierarchical clustering algorithm for VM placement

By using an agglomerative clustering algorithm, the clusters of PMs are setup based on the type of resources provided by the PMs. Each PM is identified by CPU performance, disk storage, the amount of RAM and network bandwidth. The software layer of the system is tiered comprising of local and global managers. The benefits of such clustering (groups of clusters) are provided below: The searching time of the most appropriate PM for placing a particular VM is reduced. All the local managers maintain the list or indexes of PMs in a particular cluster for other clusters. When a new VM instance request is sent to the global manager, it takes the updates of the available resources from all the local managers of each cluster. Thus it maps the VM to the most appropriate cluster of PM. The migration time is thus reduced. Normally VMs with large instance took a longer time to serve the instances. The cluster indexes provide the VMs along with its threshold value that can accept such large instances. Therefore, as the choosing time of VM placement is reduced, the migration time is also reduced atomically. Most importantly, it has been observed that by combining hierarchical clustering with firefly algorithm the total number of VM migrations had been reduced largely. This is because the VMs will be sent to a specific cluster of PMs (which can provide the amount of resource required by the VMs) instead of sending the VMs randomly.

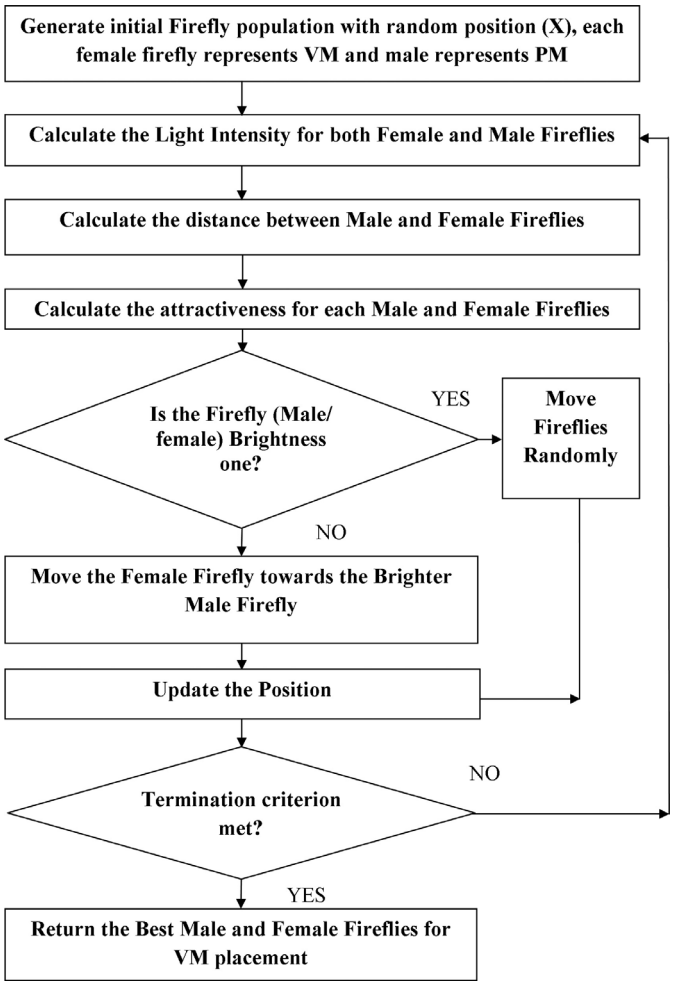

Fig. 3. Flowchart of firefly algorithm for VM placement.

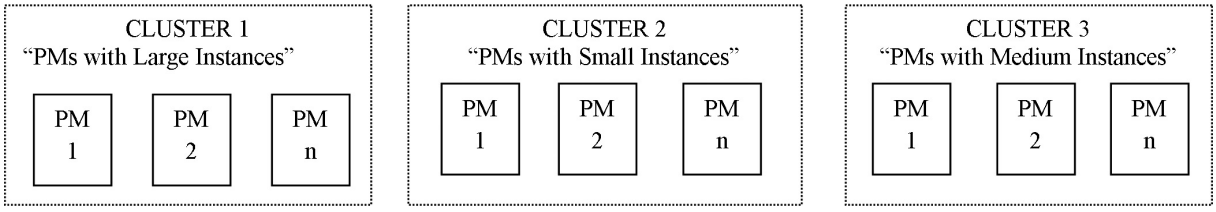

Fig. 4. Different clusters of PMs that serve different type of VM instances.

3.3.3. Different clusters of PMs and how they help in VM placement

In Fig. 4 the PM represents the Physical Machine. All PMs that can serve large VM instances are grouped together in cluster 1; all PMs that can serve small VM instances are put in cluster 2; all PMs that can serve medium instances are in cluster 3.

In Fig. 5 if the large VM instance is sent to cluster 1 then it will be served very efficiently as this cluster contains PMs that can serve VMs with large instances properly. If the large VM instance is sent

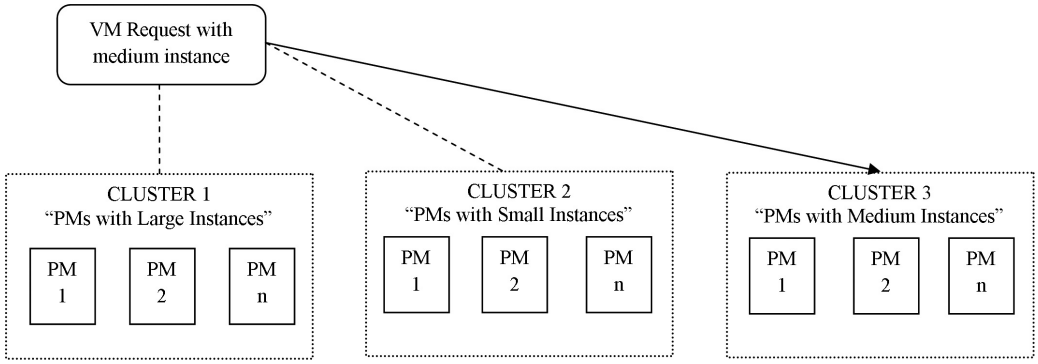

Fig. 5. A large VM instance request arrives which is sent to cluster 1.

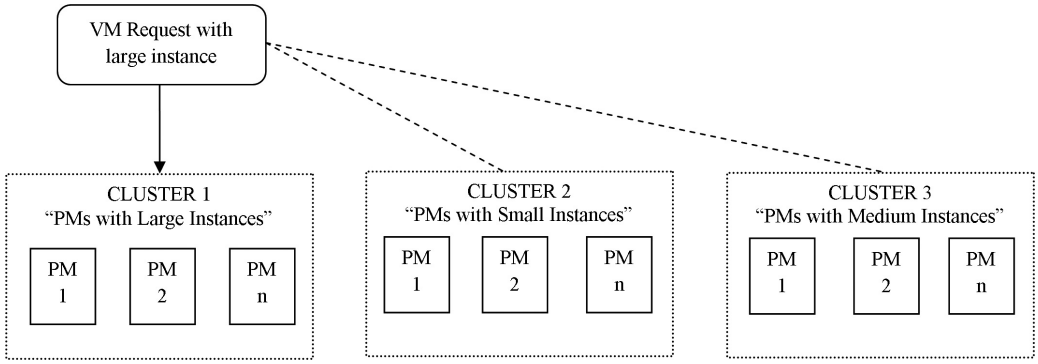

Fig. 6. A medium VM instance request arrives which is sent to cluster 2.

to cluster 2 then it will get overloaded immediately as this cluster contains PMs that can only serve small instances and as a result of this the number of migrations will be more. If the large VM instance is sent to cluster 3 then it will become overloaded very soon because the cluster cannot support the large VM instance when the resource utilisation increases with time, leading to a large number of VM migrations.

In Fig. 6 if the medium VM instance is sent to cluster 1 then it will be served, but some resources will be left underutilised as the VM instance is medium. So it will not use all the resources provided by the PM as this cluster contains PMs that can serve VMs with large instances properly. If the medium VM instance is sent to cluster 2 then it will get overloaded very fast as this cluster contains PMs that can only serve small instances and as a result of this the number of migrations will be increased. If the medium VM instance is sent to cluster 3 then it will be served very efficiently. So wastage of resources can be avoided if the VM instance is sent to cluster 1 and no overutilization will take place if sent to cluster 2, thus avoiding VM migrations.

In Fig. 7 if the small VM instance is sent to cluster 1 then it will be served, but the major portion of the resources will be left underutilised as the VM instance is small. So it will not use all the resources provided by the PM as this cluster contains PMs that can serve VMs with large instances properly. Thus resource wastage will take place leading to increased energy consumption. If the small VM instance is sent to cluster 2 then it will be served very efficiently. Wastage of resources will not occur like it could happen if the VM instances are sent to cluster 1 and cluster 3. If the small VM instance is sent to cluster 3 then some amount of resources will be left underutilised as small instance VM will not use all

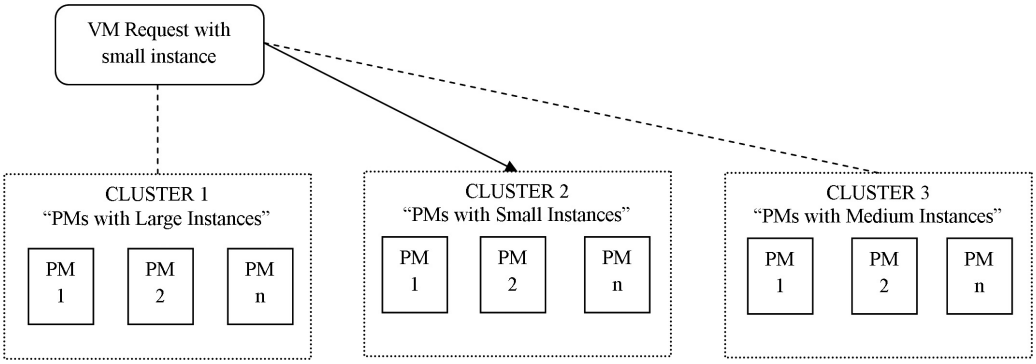

Fig. 7. A small VM instance request arrives which is sent to cluster 3.

the resources provided by a PM that can serve medium instances. Therefore, again underutilization will occur and will lead to higher energy consumption.

From Figs 4–7, it can be seen that all the VM instances could be sent to the most appropriate cluster that can serve it very efficiently by placing the VM to available PMs in that particular cluster. Once a VM instance requests arrive it can be easily sent to a particular cluster with the help of clustering algorithm used. After it is sent to a particular cluster, then it will be placed in a PM that is most capable of serving that particular VM instance. Thus searching of the PMs for VM placement will become easy and also the time for search will reduce as it is already sent to a cluster which can serve the VM instance. The only difference is that the search will be performed within that cluster, and the most suitable PM will be searched for VM allocation. All the PMs in a cluster may not be free at a given period and so by using firefly algorithm, the most suitable PM can be found and hence VM placement can be done. Thus using the clustering technique, the time for searching the most appropriate PM will be reduced and also underutilization, or overutilization will be avoided to a great extent which will also help in reducing the total number of VM migrations.

4. Performance evaluation and results

The metrics used for measuring the energy consumption and violation of SLA are given below. The performance of the proposed work has been evaluated using existing metrics [47]. This algorithm is used to optimise two main parameters-energy consumption and SLA violation related to the performance degradation. To portray the energy-performance trade-off, both the definition of energy consumption and performance degradation must be defined distinctly. In this study, the Energy Consumption (EC) by a server is defined as a linear function of CPU utilisation, and performance is defined as a function of evaluating the SLA delivered to any VM deployed in an IaaS. The SLA violation is defined with the help of two metrics-SLA Violation Time per Active Physical machine (SLATAH) that rise with overload period of the PM, and Performance Degradation due to Migrations (PDM) that rise due to live migration. Hence these metrics were defined with the assumption that the SLAs are delivered when 100% performance requested by any applications inside a VM is provided at any time.

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{D_{d_j}}{D_{r_j}} \tag{5}$$

Table 1
Characteristics of workload data

Data	Number of VMs	Mean	St. dev	Quartile 1	Median	Quartile 3
Workload 1	1052	12.31%	17.09%	2%	6%	15%
Workload 2	1516	9.26%	12.78%	2%	5%	12%

where M is the number of VMs; D_{d_j} is an estimation of the performance degradation of the VM_j caused by migration; D_{d_j} is total CPU capacity requested by VM_j during its lifetime.

A metric for describing SLA violation (*SLAV*) can be defined as follows:

$$SLAV = SLATAH \times PDM \tag{6}$$

In consideration of formulation above SLA Time (*SLAT*) for each physical machine can be defined as:

$$SLAT_i = (T_{si}/T_{ai}) \;\; 1 \leqslant i \leqslant N \tag{7}$$

where T_{si} is total time during which physical machine i has experience maximum CPU utilisation; T_{ai} is total time during which physical machine i being in the serving VMs; N is the number of active physical machines.

The CloudSim toolkit [68] has been chosen to carry out the experiments in a simulation platform and also real life workload from PlanetLab's monitoring infrastructure [69] has been collected and utilised for the VM workload traces. To compare the proposed algorithms for VM placement with the Honeybee cluster based technique (HCT) from [7] along with the existing algorithms [47] for VM selection, host overload detection and host for load detection. For overload detection existing algorithms from [7] are used, which are as follows: Static Threshold (THR), Median Absolute Deviation (MAD), Inter-Quartile Range (IQR), Local Regression (LR) and Robust Local Regression (LRR). Once the overloads are detected, it uses the different policies of VMs selection such as Maximum Correlation (MC), Minimum Migration Time (MMT), Minimum Utilization (MU) and Random Selection (RS). The simulation was done with 800 heterogeneous PMs in a data centre. Two types of servers were taken, the first type is HP ProLiant ML110 G4 and the second type is HP ProLiant ML110 G5, wherein the 800 PMs were divided into two parts and half of the PMs belong to the first type and the remaining half belong to the second type respectively. The PMs are rigged with multi-core CPUs where each core has 'p' MIPS, and therefore if there are 'n' numbers of cores then the overall capacity of the 'n' multi-core CPUs is 'np'. In this work, it is assumed that each of the VMs can have a single core and not more than that because if a VM needs more capacity than a single core, then the VM should be run parallel on other cores, which is another critical research issue [47]. The data for power consumption is taken from SPEC power benchmark [70] where the power utilisation varies for the elected PMs at each and every load level. Each of the PM is designed to have 1 GB/s network bandwidth and the instances of the VM are of four types, such as: (i) High-CPU Medium Instance; (ii) Extra Large Instance; (iii) Small Instance and (iv) Micro Instance. Instantiation of VM is made conforming to the requirements of resources denoted by the VM types. Nevertheless, throughout the lifetime of VMs there is a variation in the resource utilisation by the VMs by the workload data and hence gives a chance for performing dynamic consolidation. Two different workload data were used that was taken in two different days. At the time of simulation, each VM is assigned workload traces at random from one of the VMs from the corresponding day. The workload data's characteristics are shown in Table 1.

4.1. Selection of algorithms for overload detections

To detect the system overload, several policies were proposed in the study [47]. The policies are Static

Threshold (THR), Median Absolute Deviation (MAD), Inter-Quartile Range (IQR), Local Regression (LR) and Robust Local Regression (LRR). Once the overloads are detected, it uses the different policies of VMs selection such as Maximum Correlation (MC), Minimum Migration Time (MMT), Minimum Utilization (MU) and Random Selection (RS). This study also uses the above policies but was using different heuristics and these policies showed significant improvement in minimising energy consumption. The study also analyses the impact of the use of different algorithms for overload detections such as Static Threshold (THR), Median Absolute Deviation (MAD), Inter-Quartile Range (IQR), Local Regression (LR) and Robust Local Regression (LRR). Each host occasionally executes an overload detection algorithm to avoid performance degradation and SLA violation. Some concept of the algorithms is givenbelow but the details are provided in the earlier study [47].

(a) A Static Threshold (THR) algorithms workin a situation where CPU utilisation threshold value detects a host overload.
(b) The Median Absolute Deviation (MAD) is a measure of statistical dispersion, and it is considered as a robust estimator.
(c) Inter Quartile Range (IQR) sets adaptive CPU utilisation threshold based on another robust statistic, like the difference between the upper and lower quartiles.
(d) Local Regression (LR) works for fitting models to localised subsets of data to build up a curve that approximates the original data.
(e) Robust Local Regression (LRR) works similar to LR but with extra robustness weight.

4.2. Selection of effective policies of VM selection

Once a host overload is detected, the VMs selection process is started. Some concept of the VM selection policies are discussed below, however, the details are provided in the study of Anton and Rajkumar [14]. The different policies of VMs selection used in this study are Minimum Migration Time (MMT), Random Selection (RS) and Maximum Correlation (MC).

(a) Minimum Migration Time (MMT) chooses the VM that requires the minimum time to complete a migration relatively. The migration time is estimated as the amount of RAM utilised by the VM separated by the spare network bandwidth available for the host.
(b) Random Selection (RS) selects a VM to be migrated from the host according to a uniformly distributed discrete random variable.
(c) Maximum Correlation (MC) selects VMs that have the highest correlation of the CPU utilisation with the other VMs.

Some information is collected from work presented in [7] where improved result was achieved by applying them to the honeybee algorithm. Because of this reason of all the VM mentioned above selection and overload detection policies are used in this paper. The results of the simulation are illustrated in the following sections.

4.3. Simulation results of modified firefly algorithm with honeybee

The proposed modified firefly algorithm for VM placement has been implemented, and the results of this study showed are duction in the VM migration, SLA violation and Energy consumption. The experimental result with Workloads 1 and 2 are given in Tables 2 and 3 respectively. Each of the experiments is run 20 times and the common numbers measures obtained after 20 numbers of independent runs are illustrated in the Tables 2–4.

Table 2
Firefly and honeybee for VM placement using overload detection and VM selection for Workload 1

Overload detection-VM selection	Energy (KWh)		SLA		VM migration	
VM placement →	Firefly	Honeybee	Firefly	Honeybee	Firefly	Honeybee
IQR-MC	32.17	41.90	0.00008	0.00012	869	889
IQR-MMT	32.21	41.47	0.00007	0.00013	880	931
IQR-MU	32.35	42.41	0.00009	0.00012	919	907
IQR-RS	32.91	44.44	0.00008	0.00009	867	869
LR-MC	31.81	44.56	0.00008	0.00010	907	900
LR-MMT	32.09	41.45	0.00007	0.00012	874	857
LR-MU	32.50	42.19	0.00008	0.00013	908	896
LR-RS	31.79	44.17	0.00007	0.00011	833	885
LRR-MC	31.66	46.62	0.00009	0.00010	923	841
LRR-MMT	30.87	44.82	0.00009	0.00011	971	918
LRR-MU	32.06	45.38	0.00008	0.00010	860	879
LRR-RS	32.32	41.90	0.00009	0.00012	871	948
MAD-MC	31.73	43.32	0.00008	0.00011	855	884
MAD-MMT	32.91	43.01	0.00007	0.00012	824	893
MAD-MU	32.00	43.31	0.00009	0.00013	900	875
MAD-RS	31.79	44.82	0.00008	0.00011	908	906
THR-MC	33.99	43.46	0.00007	0.00012	881	894
THR-MMT	31.96	43.45	0.00009	0.00011	853	921
THR-MU	32.41	43.46	0.00008	0.00012	891	905
THR-RS	30.82	44.51	0.00009	0.00010	917	911

Table 3
Firefly and honeybee for VM placement using overload detection and VM selection for Workload 2

Overload Detection-VM Selection	Energy (KWh)		SLA		VM migration	
VM placement →	Firefly	Honeybee	Firefly	Honeybee	Firefly	Honeybee
IQR-MC	34.77	47.99	0.00009	0.00013	888	899
IQR-MMT	33.29	46.77	0.00008	0.00014	898	934
IQR-MU	35.34	45.83	0.00010	0.00013	956	966
IQR-RS	34.99	48.54	0.00009	0.00010	877	888
LR-MC	36.33	47.67	0.00009	0.00011	915	915
LR-MMT	35.33	44.88	0.00010	0.00013	867	870
LR-MU	38.55	49.88	0.00009	0.00014	977	920
LR-RS	39.66	48.76	0.00011	0.00012	856	900
LRR-MC	38.61	45.66	0.00010	0.00011	978	876
LRR-MMT	36.81	47.88	0.00010	0.00012	988	953
LRR-MU	35.77	48.39	0.00009	0.00012	888	920
LRR-RS	37.82	49.64	0.00010	0.00013	898	978
MAD-MC	33.75	49.38	0.00009	0.00012	891	921
MAD-MMT	33.93	48.55	0.00011	0.00013	855	929
MAD-MU	37.33	49.89	0.00010	0.00014	919	905
MAD-RS	36.99	49.78	0.00009	0.00012	925	945
THR-MC	39.19	46.66	0.00009	0.00013	898	916
THR-MMT	35.56	44.56	0.00010	0.00012	888	967
THR-MU	37.87	46.66	0.00009	0.00013	898	977
THR-RS	32.88	47.11	0.000010	0.00011	926	944

Figure 8 shows the key comparison use of energy (kWh) of proposed firefly (FF) algorithm with honeybee (HB) algorithm as per the chosen Workloads 1 and 2. The performance declined when the number of VMs was increased as presented in Tables 2 and 3. This result indicated that even when the number of VMs is increased the proposed firefly algorithm perform well in minimising the total energy

Table 4
A comparison of cluster based honeybee technique (HCT) and HCMFF

Overload detection-VM selection	Energy		SLA		VM migration	
VM placement →	HCT	HCMFF	HCT	HCMFF	HCT	HCMFF
IQR-MC	34.71	34.17	0.00009	0.00008	854	889
IQR-MMT	36.52	33.21	0.00009	0.00007	865	880
IQR-MU	34.35	34.35	0.00010	0.00009	887	919
IQR-RS	34.29	33.91	0.00010	0.00008	852	867
LR-MC	34.02	26.17	0.00010	0.00007	856	907
LR-MMT	33.99	35.09	0.00010	0.00006	882	815
LR-MU	36.85	33.50	0.00008	0.00007	869	908
LR-RS	34.75	32.79	0.00009	0.00018	867	866
LRR-MC	36.52	33.66	0.00010	0.00009	896	1218
LRR-MMT	35.17	28.16	0.00010	0.00007	874	830
LRR-MU	33.47	30.04	0.00011	0.00008	908	1183
LRR-RS	35.71	33.32	0.00009	0.00009	861	898
MAD-MC	34.27	32.73	0.00011	0.00008	916	876
MAD-MMT	34.84	32.91	0.00010	0.00016	882	873
MAD-MU	34.70	32.87	0.00009	0.00009	873	945
MAD-RS	34.84	31.98	0.00009	0.00008	866	981
THR-MC	35.88	33.46	0.00009	0.00011	895	889
THR-MMT	34.30	32.77	0.00010	0.00009	868	877
THR-MU	34.88	34.88	0.00010	0.00008	854	874
THR-RS	34.27	30.19	0.00011	0.00009	908	997

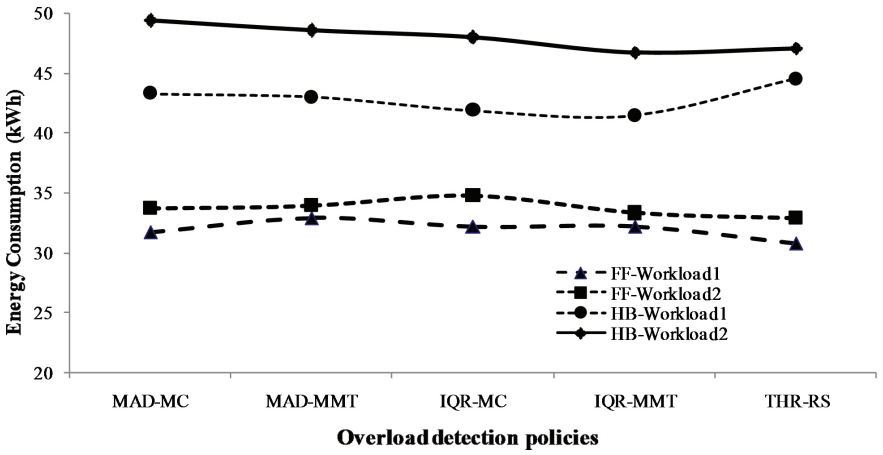

Fig. 8. Energy consumption comparison with Workloads 1 and 2: The proposed firefly algorithm for Workload 1 outperformed the honeybee algorithm and gave better results by giving minimising the total energy consumption.

consumption as in Fig. 8. It is also observed that the percentage of SLA violation was less for firefly algorithm with both Workloads 1 and 2 as in Fig. 9. Hence, the performance of honeybee algorithm was outperformed by firefly algorithm even with workload changes. In selected best pairs of overload detection vs. VM selection policies, it has been observed that for all the workloads the proposed firefly algorithm outperformed the honeybee algorithm. Firefly algorithm gave better results by giving less number of VMs as in Fig. 10.

Figure 11 gives the overall performance of all participating parameters (Overload Detection vs. VM Selection policies) used in this study for Workload 2. It shows that the firefly algorithm consumes less energy than the honeybee. The reason for this is the novel idea of attraction via light intensity as an

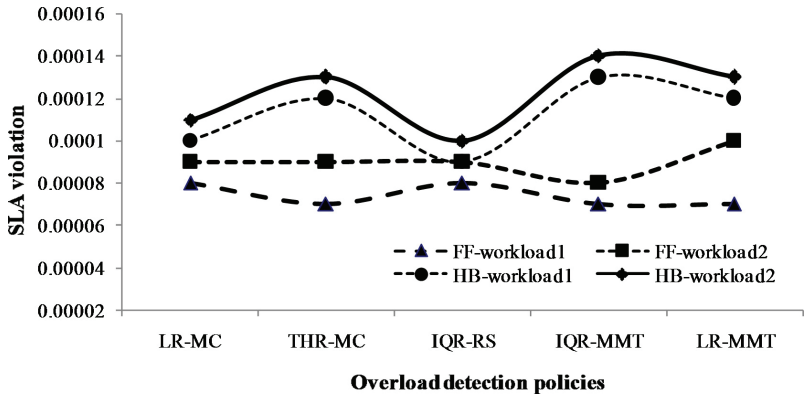

Fig. 9. SLA violation comparison with Workloads 1 and 2: The study shows SLA violation was less for firefly algorithm with both Workloads 1 and 2.

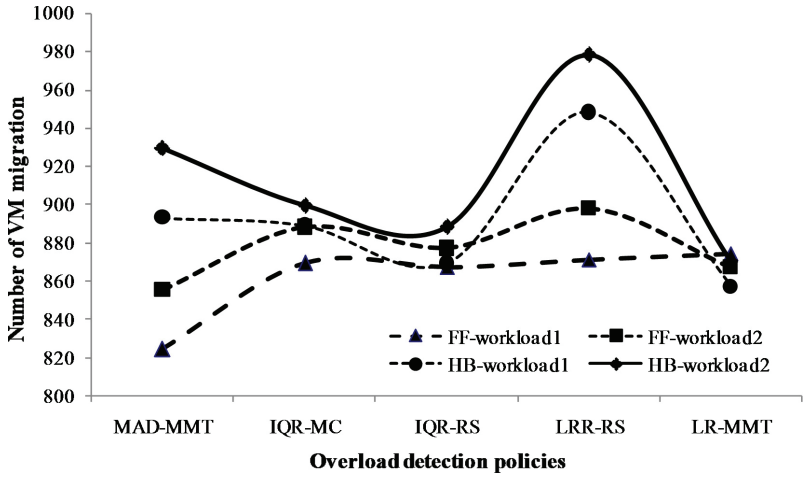

Fig. 10. VM migration comparison with Workloads 1 and 2: Firefly algorithm gave better results by giving less number of VMs.

exploitation mechanism was used in firefly algorithm and the main function of such attraction is to enable an algorithm to converge quickly because these multi-agent systems evolve, interact and attract, leading to some self-organized behaviour and attractors. As the swarming agents evolve, it is possible that their attractor states will move towards to the true global optimality.

The impact of the important parameters on the host overload detection and VM selection policies are shown in Figs 12–14 that relates to energy consumption, SLA Violation and VM migration respectively. From the study results, it is known that the dynamic VM consolidation with firefly algorithm significantly reduces energy consumption by adjusting the number of active servers. The energy consumption is low under overload detection policy (IQR, LR, LRR. MAD and THR). The VM selection policy used are Maximum Correlation (MC), Minimum Migration Time (MMT), Minimum Utilization (MU) and Random Selection (RS). The best result is provided by the pair of Overload Detection (IQR) and VM Selection (MU). In the same manner, the some of the select notable results of SLA violation are provided in Fig. 13.

Figures 15–17 show the impact of most contributing parameters for energy consumption, SLA Violation and VM migration respectively. During the VM placement of firefly algorithms, the parameter

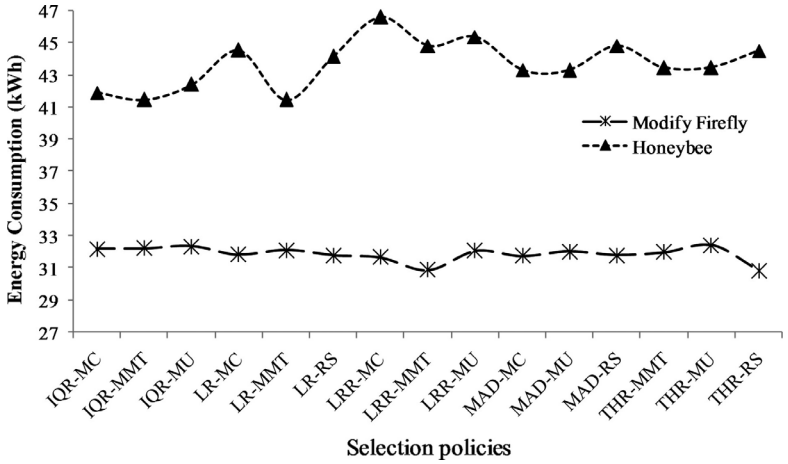

Fig. 11. Comparison of energy consumption: Firefly algorithm and honeybee algorithm: The firefly algorithm consumes less energy than the honeybee as it uses an exploitation mechanism.

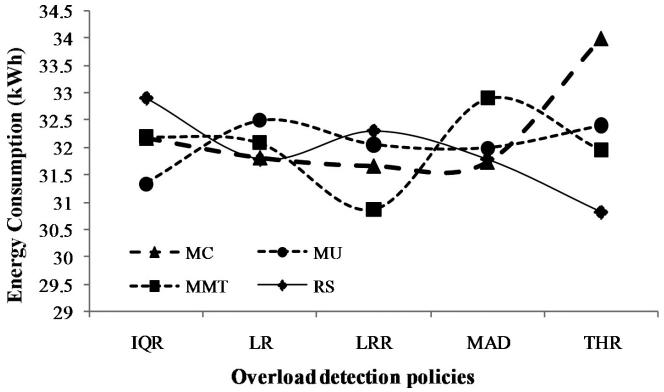

Fig. 12. Energy consumption with firefly algorithm (parameters): VM consolidation significantly reduces energy consumption by adjusting the number of active servers and the best result is provided by overload detection policy (IQR, LR, LRR. MAD and THR).

THR-RS gives the lowest value of energy consumption as 30.82 kWh. The parameter LR-MMT of the honeybee gives the lowest value of energy consumption as 41.45 kWh. In honeybee, the minimum percentage of SLA Violation is contributed by IQR-RS as 0.00009%, whereas in firefly the minimum SLA is 0.00007%. In honeybee, the number of best VM migrations 857 is given by LR-MMT and in firefly it was 824 that is contributed by MAD-MMT. The proposed approach is distributed, scalable, and efficient in managing the energy-performance trade-off.

The firefly algorithm with LR-MMT and THR-RS policies gives a better result for energy compared to other policies. The firefly algorithm together with IQR-RS and LR-MMT show less number of SLA violation compared to other overload detection and VM selection policies and firefly also gave better results than honeybee algorithm. Firefly algorithm combined with LRR-MMT and MAD-MMT has less number of VM migrations than with other overload detection and VM selection policies. The proposed firefly algorithm significantly reduced energy consumption, SLA violation and VM migration in comparison to the Honeybee (HB) algorithm proposed in [7].

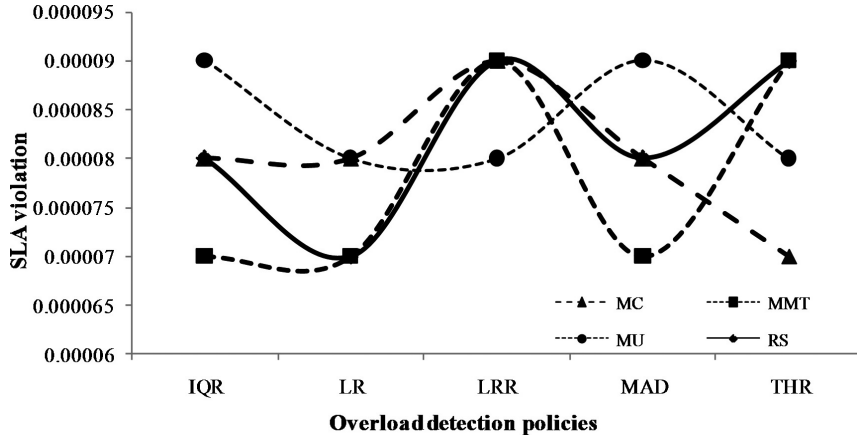

Fig. 13. SLA violation with firefly algorithm (parameters): SLA violation is reduced by parameter IQR and MM.

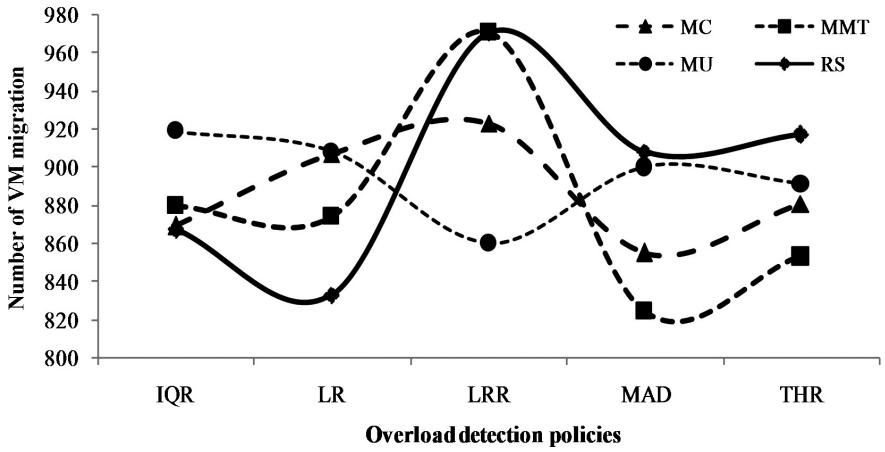

Fig. 14. Number of VM migration with firefly algorithm (parameters).

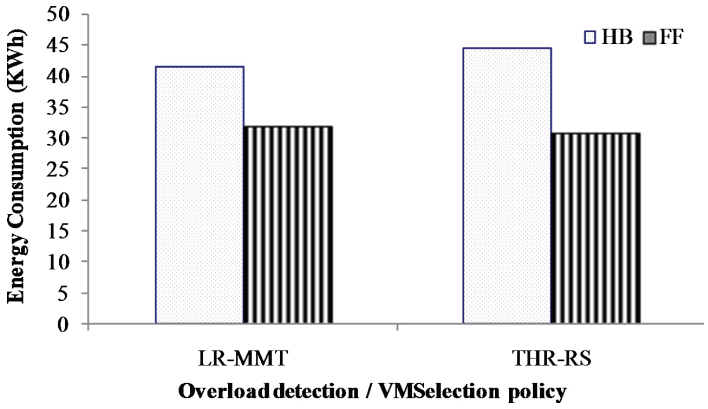

Fig. 15. Energy consumption comparison (firefly and honeybee): LR-MMT and THR-RS policies uses the least energy.

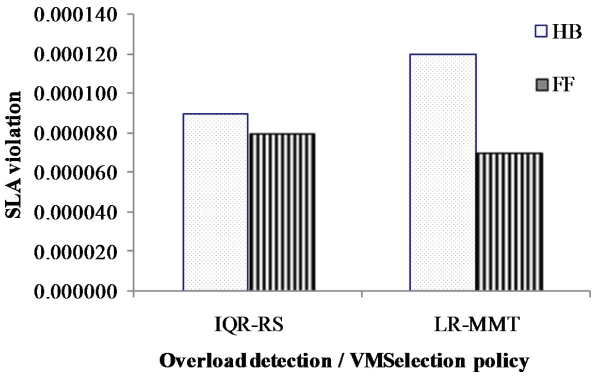

Fig. 16. Comparison of SLA violation (firefly and honeybee): The parameter LR-MMT of the honeybee gives the lowest value of energy.

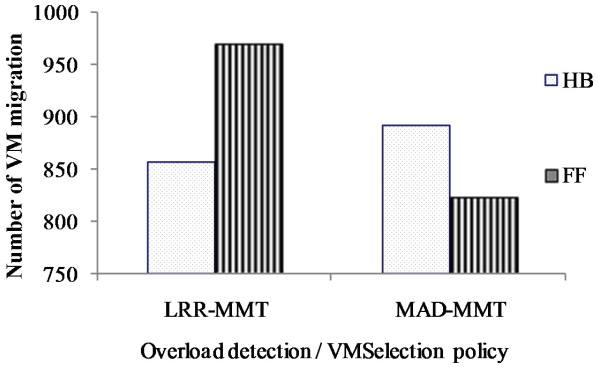

Fig. 17. Comparison of VM migration (firefly and honeybee): The lowest SLA is contributed by MAD-MMT using firefly algorithm.

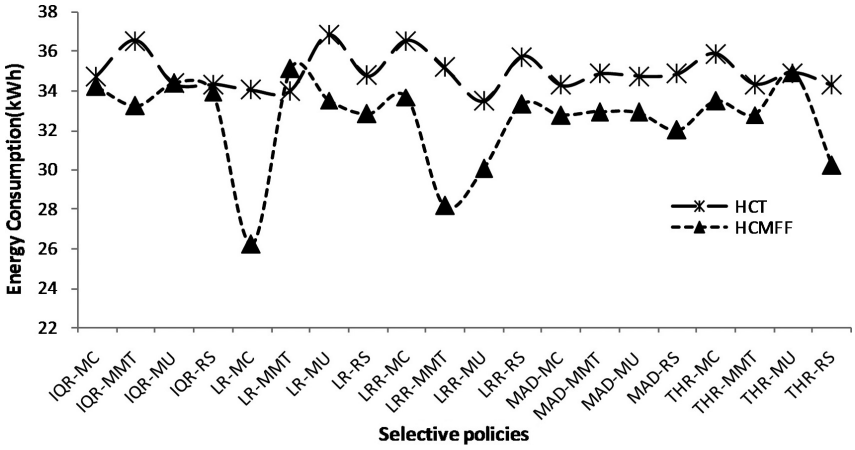

Fig. 18. Energy consumption by HCMFF and HCT (cluster based honeybee technique).

4.4. Simulation results of hierarchical cluster-based modified firefly algorithm (HCMFF)

We wanted to improve the result further. The system model of HCMFF was shown in Fig. 2. The result of HCMFF is compared to HCT. The related literature and ideas are collected from earlier studies [47].

Figure 18 shows the overall performance of most of the parameters used in the study. It shows that the HCMFF consumes less energy than HCT. The prime reason is that the searching time for VM placement is considerably reduced due to the clustering technique. The impact of the critical parameters is shown in the Figs 19–21. The proposed approach showed better results when compared to HCT giving minimised the energy consumption.

Further, Figs 22–24 show the most important contributing parameters for energy consumption, SLA Violation and VM migration respectively. From this simulation study, it can be derived that the dynamic VM placement with HCMFF substantially minimised energy consumption by adjusting the number of active servers. In HCMFF the VM placement with the lowest value of energy consumption is given by LR-MC as 26.17 kWh, wherein the HCT VM placement with the lowest value of energy consumption is given by LRR-MU as 33.47 kWh. In HCT, the minimum percentage of SLA violation was 0.00008%

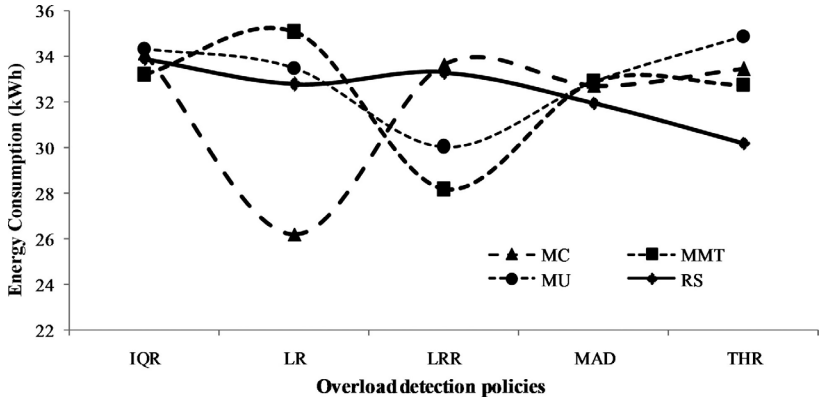

Fig. 19. Energy consumption of HCMFF algorithm: Consumes less energy than HCT due to the clustering technique.

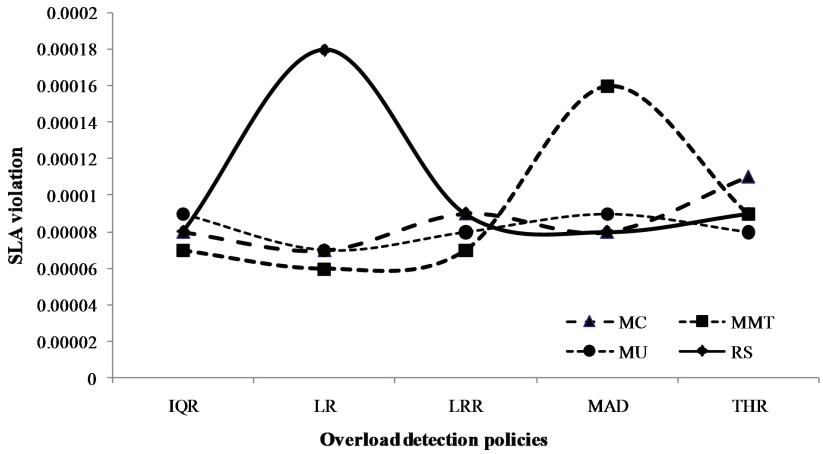

Fig. 20. SLA violation of HCMFF algorithm: It gives less SLA violation due to uses of the clustering technique.

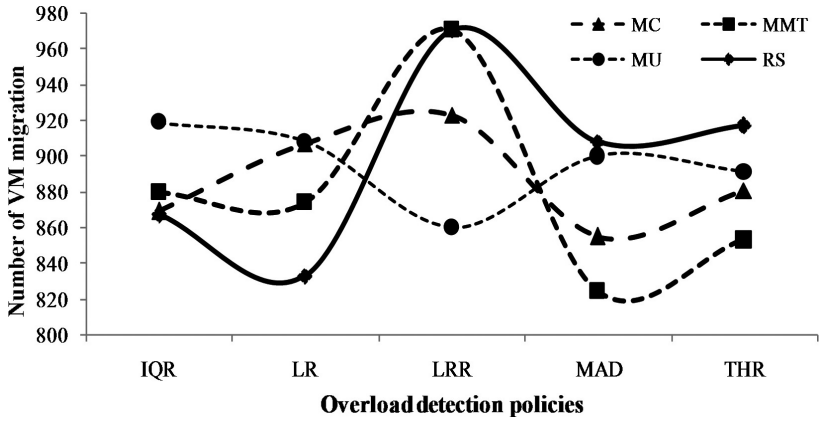

Fig. 21. Number of VM migration of HCMFF algorithm.

Table 5
Best results of all four algorithms

	Firefly	Honeybee	HCT	HCMFF
IQR-MMT	32.21	41.47	36.52	33.21
LR-MC	31.81	44.56	34.02	26.17
LR-RS	31.79	44.17	34.75	32.79
LRR-MC	31.66	46.62	36.52	33.66
LRR-MMT	30.87	44.82	35.17	28.16
LRR-MU	32.06	45.38	33.47	30.04
MAD-RS	31.79	44.82	34.84	31.98
THR-MMT	31.96	43.45	34.3	32.77
THR-RS	30.82	44.51	34.27	30.19

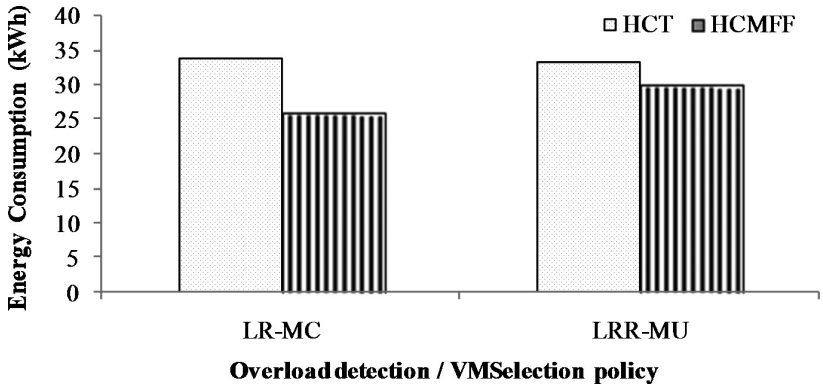

Fig. 22. Energy consumption with HCMFF and HCT: The lowest value of energy consumption is given by LR-MC.

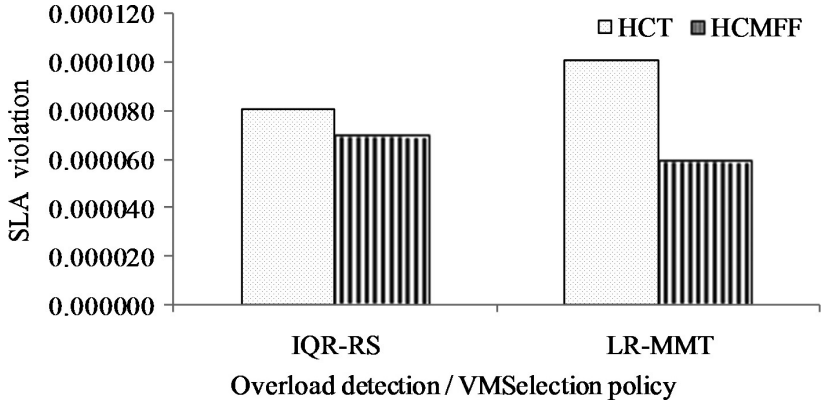

Fig. 23. SLA Violation with HCMFF and HCT: VM placement with the lowest value of energy consumption is given by LR-MMT.

by LR-MU, whereas in HCMFF it was 0.00006% given by LR-MMT. In HCT, the minimum number of VM migrations was 852 by IQR-RS and in HCMFF it was 815 given by LR-MMT.

The results indicate that HCMFF algorithm with LR-MC gave a better result for energy compared to other policies. HCMFF algorithm together with LR-MMT show less number of SLA violation compared to other overload detection and VM selection policies and HCMFF also gave better results than HCT algorithm that was used in [71]. HCMFF algorithm combined with LR-MMT has less number of VM

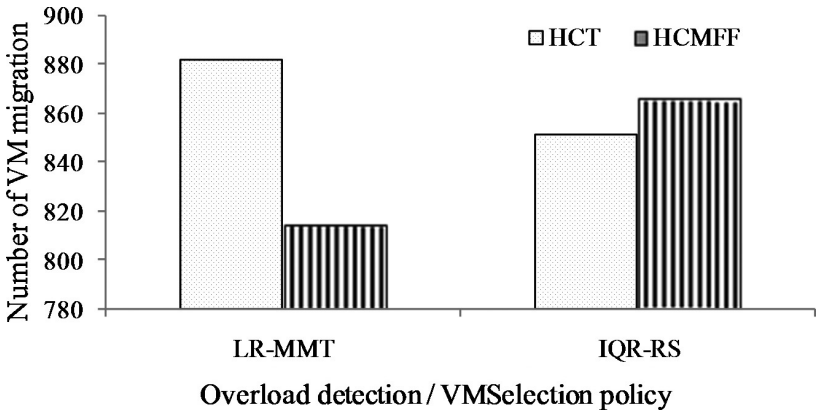

Fig. 24. Number of VM migration with HCMFF and HCT.

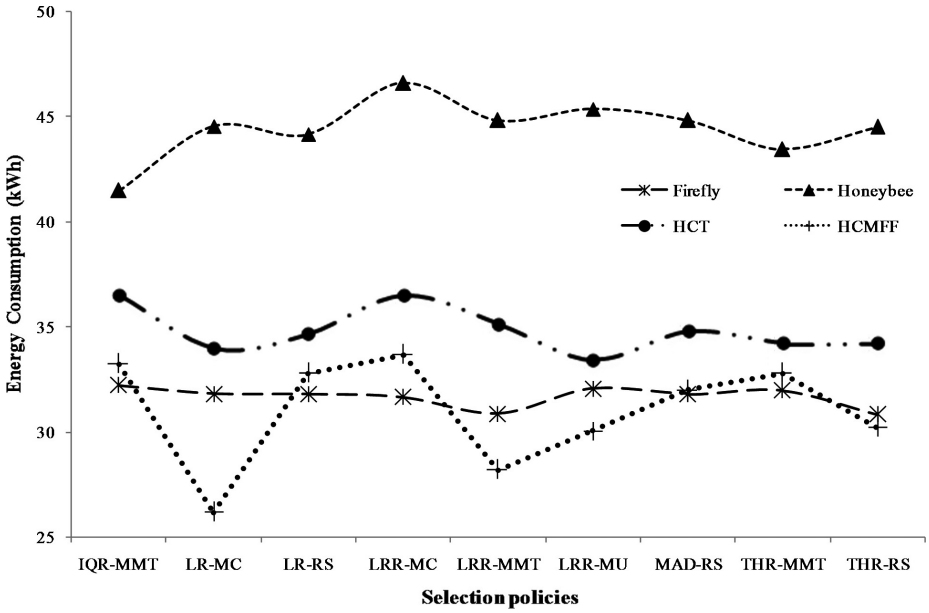

Fig. 25. The overall comparison of four different techniques for VM placement (energy consumption): The HCMFF gives the best result as it uses the combined advantages of firefly and hierarchical clustering concepts.

migrations than with other overload detection and VM selection policies. The results indicated that the HCMFF algorithm performed fewer VM migrations in comparison to HCT algorithm.

4.5. The overall remarks of the study

Some of the best results of all four algorithms on the energy consumption are provided in Table 5. The overall results of all four algorithms on the energy consumption are analysed and shown in Fig. 25. Only the most important contributing parameters are considered for analysis. The HCMFF gives better result than those of HCT, honeybee and firefly. The simulation study of HCMFF outperformed both HCT and firefly. Thus, HCMFF proved to be most efficient for all three metrics (energy consumption, SLA violation and VM migration). The reason HCMFF outweighs other algorithms is because it combines the

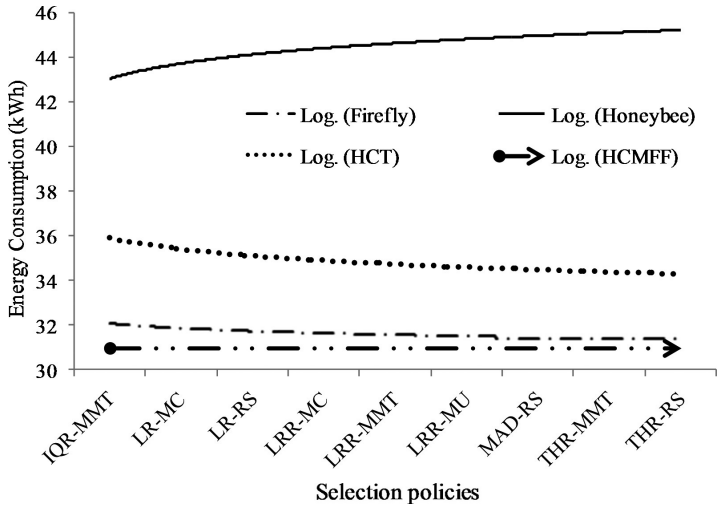

Fig. 26. The overall comparison of four different techniques (energy consumption): The HCMFF consumes less energy than honeybee, HCT and original firefly.

advantages of both firefly and hierarchical clustering algorithms that enhanced the overall performance. A close view of the overall comparison of four different techniques is provided in Fig. 26. The HCMFF consumes 12% less energy than Honeybee algorithm, 6% less than HCT algorithm and 2% less than original firefly.

5. Conclusion

VM placement has become an important research problem to provide energy efficient cloud computing environment. The cloud providers must implement energy efficient resource management techniques to maximise their return on investment (ROI). Hence dynamic consolidation of VMs has become an essential solution for this problem that is achieved by switching idle servers to power-saving modes. The proposed modified firefly algorithm and hierarchical cluster based modified firefly algorithm (HCMFF) reduce the energy consumption in the datacentre. The efficiency of the proposed algorithms is evaluated through simulations in CloudSim3.0.3 using workload traces from PlanetLab. This study contributes a new VM placement algorithm using a meta-heuristics concept. Both these algorithms show better results with different combinations of overload detection policy and VM selection policy. Modified firefly algorithm and HCMFF show significant improvement as compared with honeybee algorithm and existing Honeybee cluster based technique (HCT) respectively. The uses of the different algorithms for overload detections and effective policies of VM selection gives better merit in this study.

The strength of the modified firefly algorithm (MFF) is that it uses two important features, namely automatic subdivision and the ability to deal with multimodality for optimisation. Firefly algorithm gives an assurance of finding near-optimal solutions within a remarkable decline in the amount of time. It also has the following advantages: such as dynamic or automatic subdivision of the whole population into subgroups, and high ergodicity and diversity in the solutions. Such advantages make firefly algorithm unique and very efficient. The study shows that the modified firefly algorithm uses 10% less energy than Honeybee algorithm.

The strength of HCMFF is that the searching time for VM placement is substantially reduced by the use of hierarchical clustering which helps in finding the best cluster among the different cluster of physical machines that will be most capable and efficient for any VM placement. This algorithm uses maximum exploitation mechanism to efficiently use energy and other resources. The study shows that HCMFF consumes 12% less energy than Honeybee algorithm, 6% less than HCT algorithm and 2% less than original firefly.

The use of the appropriate algorithm can help in efficient usages of energy in cloud computing. However, this work considers only a single meta-heuristic algorithm and requires further comparison with the various meta-heuristic algorithms for virtual machine placement, e.g., ACO, PSO, etc. to verify the performance of the algorithms in an extensive manner. Also more metrics can be considered to improve the evaluation and validation process of the algorithms.

References

[1] S. Kumar, V. Talwar, V. Kumar, P. Ranganathan and K. Schwan, vManage: Loosely coupled platform and virtualization management in data centres, in: *Proc 6th Int Conf on Autonomic computing*, ACM, (2009), 127–136.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield, Xen and the art of virtualization, in: *Proceedings of ACM Symposium on Operating Systems Principles*, ACM, (2003), 164–177.

[3] L. Zeng, B. Veeravalli and Q. Wei, Space4time: Optimization latency-sensitive content service in cloud, *Journal of Network and Computer Applications*, Elsevier **41** (2014), 358–368.

[4] Y. Wang, Z. Zhou, L. Liu and W. Wu, Replica-aided load balancing in overlay networks, *Journal of Network and Computer Applications*, Elsevier **36**(1) (2013), 388–401.

[5] S.K. Garg, A.N. Toosi, S.K. Gopalaiyengar and R. Buyya, SLA-based virtual machine management for heterogeneous workloads in a cloud data center, *Journal of Network and Computer Applications*, Elsevier **45** (2014), 108–120.

[6] W. Vogels, Beyond server consolidation, *ACM Queue* **6** (2008), 20–26.

[7] A.N. Singh and M. Hemalatha, Cluster based bee algorithm for virtual machine placement in cloud data centre, *Journal of Theoretical and Applied Information Technology* **57** (2013), 1–10.

[8] R. Buyya, C.Y. Shin and S. Venugopal, Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities, in: *Proc 10th IEEE Int Conf on High-Performance Computing &* Communications, IEEE, (2008), 5–13.

[9] N. Bobroff, A. Kochut and K. Beaty, Dynamic placement of virtual machines for managing SLA violations, in: *Proc International Symposium on Integrated Network Management*, IEEE, (2007), 199–128.

[10] D. Barbagallo, E.D. Nitto, D.J. Dubois and R. Mirandola, A bio-inspired algorithm for energy optimization in a self-organizing data center, *Self-Organizing Architectures*, Springer-Verlag, (2010), 127–151.

[11] S.H.H. Madni, M.S.A. Latiff, Y. Coulibaly and S.M. Abdulhamid, An appraisal of meta-heuristic resource allocation techniques for IaaS cloud, *Indian Journal of Science and Technology*, 9.4, 2016.

[12] M.N.A. Wahab, S. Nefti-Meziani and A. Atyabi, A comprehensive review of swarm optimization algorithms, PloS one 10.5, 2015.

[13] M. Dorigo, M. Birattari and T. Stutzle, Ant colony optimization, in: *IEEE Computational Intelligence Magazine*, IEEE **1**(4) (2006), 28–39.

[14] X. Liu and D. He, Ant colony optimization with greedy migration mechanism for node deployment in wireless sensor networks, in: *Journal of Network and Computer Applications*, Elsevier **39** (2014), 310–318.

[15] 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* **79**(8) (2013), 1230–1242.

[16] K. Li and H. Shen, Optimal proxy placement for coordinated en-route transcoding proxy caching, *IEICE Trans Inform Syst* **87**(12) (2004), 2689–2696.

[17] K. Li and H. Shen, Optimal placement of web proxies for tree networks, in: *Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service*, IEEE, (2004), 479–486.

[18] 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*, Springer (2014), 153–164.

[19] J. Hu, J. Gu, G. Sun and T. Zhao, A strategy on load balancing of virtual machine resources in cloud computing environment, *3rd Int Symposium on Parallel Architectures, Algorithms and Programming*, IEEE, (2010), 89–96.

[20] E. Falkenauer, A hybrid grouping genetic algorithm for bin packing, *Journal of Heuristics*, Springer **2**(1) (1996), 5–30.

[21] S. Sawant, A genetic algorithm scheduling approach for virtual machine resources in a cloud computing environment, Master's Projects, Paper 198. http://scholarworks.sjsu.edu/etd_projects/198, (2011), (accessed on 10 May, 2016).
[22] J. Hu, J. Gu, G. Sun and T. Zhao, A scheduling strategy on load balancing of virtual machine resources in cloud computing environment, in: *IEEE's Third International Symposium on Parallel Architectures, Algorithms and Programming*, (2010), 89–96.
[23] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi and L. Yuan, Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers, in: *Proc of the IEEE International Conf on Services Computing*, (2010), 514–521.
[24] J. Xu and J. Fortes, Multi-objective virtual machine placement in virtualized data center environments, in: *Proceedings of the IEEE/ACM International Conference on Green Computing and Communications &* 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing, (2010), 179–188.
[25] I. Fister, Jr, I. Fister, X.S. Yang and J. Brest, A comprehensive review of firefly algorithms, *Swarm and Evolutionary Computation*, Elsevier **13** (2013), 34–46.
[26] K. Durkota, Implementation of a discrete firefly algorithm for the QAP problem within the sage framework, B Sc Thesis, Czech Technical University, 2011.
[27] I. Fister, Jr, I. Fister, J. Brest and X.S. Yang, Memetic firefly algorithm for combinatorial optimization, *Bio Inspired Optimisation Methods and Their Applications* **2** (2012), 75–86.
[28] T. Hassanzadeh, H. Vojodi and A.M.E. Moghadam, An image segmentation approach based on maximum variance intra-cluster method and firefly algorithm, in: *Proceedings of 7th International Conference on Natural Computation*, (2011), 1817–1821.
[29] G.K. Jati and S. Suyanto, Evolutionary discrete firefly algorithm for travellingsalesman problem, *International Journal for Research in Science &* Advanced Technologies, Lecture Notes in Artificial Intelligence LNAI, 6943, Springer, (2011), 393–403.
[30] M.K. Sayadi, R. Ramezanian and N. Ghaffari-Nasab, A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problem, *Int J of Industrial Engineering Computations* **1** (2010), 1–10.
[31] S. Palit, S. Sinha, M. Molla, A. Khanra and M. Kule, A cryptanalytic attack on the knapsack cryptosystem using binary firefly algorithm, in: *Proc 2nd Int Conf on Computer and Communication Technology*, (2011), 428–432.
[32] A. Yousif, A.H. Abdullah, S.M. Nor and A.A. Abdelaziz, Scheduling jobs on grid computing using firefly algorithm, *Journal of Theoretical and Applied Information Technology* **33** (2011), 155–164.
[33] M.A. Rajini, A hybrid meta-heuristic algorithm for classification using micro array data, *International Journal of Scientific &* Engineering Research **3** (2012), 1–9.
[34] J. Senthilnath, S.N. Omkar and V. Mani, Clustering using firefly algorithm: Performance study, *Swarm and Evolutionary Computation* Elsevier **1** (2011), 164–171.
[35] S. Nandy, P.P. Sarkar and A. Das, Analysis of nature-inspired firefly algorithm based back propagation neural network training, *International Journal of Compute Application* **43** (2012), 8–16.
[36] S.M. Farahani, A.A. Abshouri, B. Nasiri and M.R. Meybodi, A Gaussian firefly algorithm, *International Journal of Machine Learning and Computing* **1** (2011), 448–453.
[37] S.M. Farahani, B. Nasiri and M.R. Meybodi, A multiswarm based firefly algorithm in dynamic environments, in: *Proc Third international conference on signal processing systems*, (Yantai, China, 2011), 68–72.
[38] A.A. Abshouri, M.R. Meybodi and A. Bakhtiary, New firefly algorithm based on multi swarm and learning automata in dynamic environments, in: *Proc 3rd Int Conference on Signal Processing Systems*, (Yantai, China, 2011), 73–77.
[39] K. Mukherjee and G. Sahoo, Green cloud: An algorithmic approach, *International Journal of Computer Applications* **9**(9) (2010).
[40] D. Kumar and Z. Raza, A PSO based VM resource scheduling model for cloud computing, in: *Proc IEEE's International Conference on Computational Intelligence and Communication Technology (CICT)*, (2015), 213–219.
[41] 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: *Proc 19*$^{\text{th}}$ IEEE International Conference on Parallel and Distributed Systems (ICPADS'13), IEEE, (2013), 102–109.
[42] L. Dai and J.H. Li, An optimal resource allocation algorithm in cloud computing environment, *Applied Mechanics and Materials* **733**(1) (2015), 779–783.
[43] W. Shu, W. Wang and Y. Wang, A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing, *EURASIP Journal on Wireless Communications and Networking*, Springer, (2014), 1–9.
[44] K.N. Krishnan and D. Ghose, Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions, *Swarm Intelligence* Springer **3**(2) (2009), 87–124.
[45] S.K. Pal, C.S. Rai and A.P. Singh, Comparative study of firefly algorithm and particle swarm optimization for noisy non-linear optimization problems, *I J Intelligent Systems and Applications*, l, Mecs press, (2012), 50–57.

[46] M. Cardosa, M. Korupolu and A. Singh, Shares and utilities based power consolidation in virtualized server environments, in: *Proceedings of IFIP/IEEE Integrated Network Management (IM'09)*, (2009), 327–334.
[47] A. Beloglazov and R. Buyya, Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers, *Concurr Comput Pract Experience*, Wiley **24**(13) (2011), 1–24.
[48] S. Srikantaiah, A. Kansal and F. Zhao, Energy aware consolidation for cloud computing, in: *Proc of HotPower'08 Workshop on Power Aware Computing and Systems*, (2008), 1–5.
[49] 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* (2009), 17–24.
[50] A. Verma, P. Ahuja and A. Neogi, pMapper: Power and migration cost aware application placement in virtualized systems, in: *Proc of the 9th ACM/IFIP/USENIX International Conference on Middleware*, (2008), 243–264.
[51] E. Feller, L. Rilling and C. Morin, Energy-aware ant colony based workload placement in clouds, in: *Proceedings of the IEEE/ACM International Conference on Grid Computing (GRID)*, (2011), 26–33.
[52] S. Chaisiri, B. Lee and D. Niyato, Optimal virtual machine placement across multiple cloud providers, in: *Proceedings of the IEEE Asia-Pacific Services Computing Conference*, (2009), 103–110.
[53] M. Bichler, T. Setzer and B. Speitkamp, Capacity planning for virtualized servers, *Workshop on Information Technologies and Systems (WITS)*, Milwaukee, USA, (2006).
[54] B. Speitkamp and M. Bichler, A mathematical programming approach for server consolidation problems in virtualized data centers, *IEEE Trans Services Comput* IEEE, (2010), 266–278.
[55] H. Van, F. Tran and J. Menaud, Performance and power management for cloud infrastructures, in: *Proc of the IEEE 3rd International Conference on Cloud Computing*, IEEE, (2010), 329–336.
[56] F. Hermenier, X. Lorca, J. Menaud, G. Muller and J. Lawall, Entropy: A consolidation manager for clusters, in: *Proc of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, ACM, (2009), 41–50.
[57] A. Beloglazovy, R. Buyya, Y.C. Lee and A. Zomaya, A taxonomy and survey of energy-efficient data centers and cloud computing systems, Technical Report, CLOUDS TR-2010-3, Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, (2010), 1–51.
[58] R. Nathuji and K. Schwan, Virtual power: Coordinated power management in virtualized enterprise systems, *ACM SIGOPS Operating Systems Review* ACM **41**(6) (2007), 265–278.
[59] C. Bouras and A. Sevasti, Service level agreements for DiffServ-based services' provisioning, *Journal of Network and Computer Applications*, Elsevier **28**(4) (2005), 285–302.
[60] B. Addis, D. Ardagna, B. Panicucci, M.S. Squillante and L. Zhang, A hierarchical approach for the resource management of very large cloud platforms, *IEEE Transactions on Dependable and Secure Computing*, IEEE **10**(5) (2013), 253–272.
[61] M. Sharifi, H. Salimi and M. Najafzadeh, Power-efficient distributed scheduling of virtual machines using workload-aware consolidation techniques, *The Journal of Supercomputing*, Springer **6** (2011), 46–66.
[62] B. Perumal and A. Murugaiyan, A firefly colony and its fuzzy approach for server consolidation and virtual machine placement in cloud data centers, *Advances in Fuzzy Systems*, (2016), Article ID 6734161, 15, doi: 10.1155/2016/6734161.
[63] X.S. Yang, Engineering optimisation: An introduction with meta-heuristic applications, John Wiley & Sons, 2010.
[64] X.S. Yang, Firefly algorithms for multimodal optimization, in: *Stochastic Algorithms: Foundations and Applications, SAGA, Lecture Notes in Computer Sciences*, Springer **5792** (2009), 169–178.
[65] A.O. Bajeh and K.O. Abolarinwa, A comparative study of genetic and Tabu search algorithms, *International Journal of Computer Applications* **31** (2011), 43–48.
[66] S.C. Johnson, Hierarchical clustering schemes, *Psychometrika*, Springer **32**(3) (1967), 241–254.
[67] R. D'andrade, U-statistic hierarchical clustering, *Psychometrika*, Springer **43**(1) (1978), 58–67.
[68] R.N. Calheiros, R. Ranjan et al., CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, *Software: Practice and Experience*, Wiley **41** (2011), 23–50.
[69] K.S. Park and V.S. Pai, CoMon: A mostly-scalable monitoring system for Planet-Lab, *ACM SIGOPS Operating Systems Review*, ACM **40** (2006), 65–74.
[70] The SPECpower benchmark website, [Online]. (accessed on 18/10/2014). Available: http://www.spec.org/power_ssj2008/.
[71] L. Zhao, Y. Ren, M. Li and K. Sakurai, Flexible service selection with user-specific QoS service support in service-oriented architecture, *Journal of Network and Computer Applications*, Elsevier **35**(3) (2012), 962–973.

Authors' Bios

Esha Barlaskar completed her masters degree in Computer Science with specialization in Artificial Intelligence from Assam Don Bosco University, India. Currently Esha is a PhD student in the High Performance and Distributed Computing (HPDC) cluster at Queen's University of Belfast, United Kingdom. Esha's research interests include: cloud data center monitoring, dynamic resource management in cloud and dynamic consolidation of virtual machines.

Yumnam Jayanta Singh is a professor and head of the Department of Computer Science, Engineering and Information Technology, Assam Don Bosco University, India. He earned his PhD in Computer Science and Information Technology. His areas of research interest are Real Time Distributed Database, Cloud Computing, Digital Signal processing, Data warehousing and mining, etc.

Biju Issac is a senior lecturer at the School of Computing, Teesside University, United Kingdom. He earned PhD in Networking and Mobile Communications, along with MCA (Master of Computer Applications) and BE (Electronics and Communications Engineering). His research interests are in computer networks, application of artificial intelligence and optimization problems.