Soft Comput
DOI 10.1007/s00500-017-2905-z

METHODOLOGIES AND APPLICATION

Energy-aware virtual machine allocation and selection in cloud data centers

V. Dinesh Reddy[1,2] · **G. R. Gangadharan**[1] · **G. Subrahmanya V. R. K. Rao**[3]

Abstract Data centers evolve constantly in size, complexity, and power consumption. Energy management in cloud data centers is a critical and challenging research issue. It becomes necessary to minimize the operational costs as well as environmental impact and to guarantee the service-level agreements for the services provided by the data centers. We propose a modified discrete particle swarm optimization based on the characteristic particle swarm optimization for the initial placement of virtual machines and a novel virtual machine selection algorithm for optimizing the current allocation based on memory utilization, bandwidth utilization, and size of the virtual machine. By means of simulations, we observe that the proposed method not only saves the energy significantly than the other approaches, but also minimizes the violations of service-level agreements.

Keywords Data center · Virtual machine placement · Energy efficiency · VM selection · Discrete PSO

Communicated by V. Loia.

✉ G. R. Gangadharan
grgangadharan@idrbt.ac.in

V. Dinesh Reddy
dineshvemula@gmail.com

G. Subrahmanya V. R. K. Rao
subrahmanyavrk.rao@cognizant.com

[1] Institute for Development and Research in Banking Technology (IDRBT), Hyderabad, India

[2] School of Computer and Information Sciences, University of Hyderabad, Hyderabad, India

[3] Cognizant Technology Solutions, Chennai, India

1 Introduction

The energy intensity of data centers and the explosive growth of cloud usage calls for reducing energy use through introduction of new technologies and techniques. Increasing deployment of virtual machines on data centers requires more physical machines to fulfill the computing demands of users. Therefore, the size of a data center is increasing continually, resulting in high energy consumption and high operating costs (Gandhi and Harchol-Balter 2011). One of the primary reasons for high energy consumption is that the dynamic power range of servers, indeed without serving, still devours about 70% of their peak power (Hu and Phung-Duc 2015). From the energy consumption perspective, holding servers idle or under-utilized with slipshod management is highly inefficient. Recent growth in energy costs and environmental footprint requires advances in techniques that abate the overall energy consumption of a data center.

Energy demand is expected to grow by more than 70% in next 20 years with data centers being the key contributors for this expansion (Pernici et al. 2012). According to the Environmental Protection Agency report, data centers consume 100 times more power than traditional office systems (Belady 2007). The current carbon emission of worldwide data centers is 0.6% of the total global carbon emissions, and it is estimated to go up to 2.6% by 2020 (Curry et al. 2012). Reducing total energy consumption of the data center requires energy-aware policies and designs of software and hardware (Negru et al. 2016). In a cloud data center, the optimal way of virtual machine allocation improves energy efficiency and resource utilization. So, the virtual machines in a heterogeneous environment should be placed, balancing the load among physical machines without violating any service-level agreements (SLA) and taking care of energy efficiency (Garg et al. 2014).

Published online: 06 November 2017

To accommodate the growing demands of users and other background processes using the same physical resources, data centers are required to make optimal use of all the resources by increasing utilization and visibility. Virtual machine (VM) allocation is an important approach for improving the resource utilization and energy efficiency of the data centers. With rapid development of virtualization technology, migration of virtual machines became important to reduce the number of active servers. Proper selection of virtual machines for migration minimizes the number of power-on nodes. Research and implementation of fast energy-efficient virtual machine allocation and selection algorithms considering multiple resources can build energy-efficient data centers. Our main aim is to minimize energy consumption in a data center by reducing the server idle conditions and improving the server utilization.

In this work, we propose a modified discrete particle swarm optimization (MDPSO) approach for optimal energy-aware virtual machine allocation that minimizes the power consumption of the physical machine by estimating the increase in the power consumption before a VM is placed onto the physical machine. Further, we present a novel virtual machine selection method, namely MBS-VM selection, which optimally selects the virtual machines from a over-utilized or under-utilized server and performs migration to further improve the utilization of servers that in turn improves the energy efficiency in virtualized data centers. Experimental results show that our proposed approach reduces the energy consumption of the data center up to 32%, idle machines up to 40% and migrations up to 30%.

The salient contributions of this article are as follows:

1. A resource allocation model based on energy consumption of servers and a power consumption model based on SPECpower_ssj®2008
2. A system for energy-aware virtual machine allocation and selection (EViMAS) in cloud data centers.
3. A novel optimization approach namely MDPSO for initial placement of virtual machines that aims lower energy consumption of the data center.
4. A novel virtual machine selection algorithm based on memory utilization, bandwidth utilization, and size of the VM (MBS-VM).

The remainder of this paper is organized as follows. Section 2 describes related work in reducing energy consumption of data centers using virtual machine allocation and selection. Section 3 presents a resource allocation model and a server power consumption model. Section 4 presents the proposed EViMAS system architecture. An algorithm for energy-aware virtual machine allocation using MDPSO is presented in Sect. 5. Section 6 presents the memory, bandwidth, and size-based VM selection (MBS-VM) algorithm. Experimental results are described in Sect. 7. Solving six-hump Camelback function using MDPSO is given in Sect. 8, followed by conclusions and future directions in Sect. 9.

2 Related work

Energy-efficient resource allocation and selection is a challenging issue in data centers. Generally, the VM allocation issue is considered as a NP-hard problem (Michael and David 1979; Vomlelov and Vomlel 2003), inferring that an optimal arrangement cannot be found in deterministic polynomial time. First fit, best fit, modified best fit decreasing, and linear programming techniques are deterministic in nature and used in resource allocation and selection in data centers (Shi and Hong 2011; Maguluri et al. 2012; Mohamed and Swarnammal 2016). These techniques do not always guarantee the optimal solution. Bio-inspired approaches like PSO and genetic algorithm (GA) are iterative in nature and provide global optimal solutions in case of energy-efficient virtual machine allocation in data centers (Wu et al. 2012; Valle et al. 2008; Blackwell 2005). Younge et al. (2010) proposed a power-aware scheduling approach in which virtual machines are scheduled in a way that total power consumption of the servers is minimized. The authors proposed a greedy-based algorithm to load each node with as many virtual machines as possible which in turn decreases the number of active hosts and energy consumption. However, this approach may lead to SLA violations in peak time. Jin et al. (2012) presented a resource allocation approach considering both deterministic and stochastic resources. They proposed max–min multi-dimensional stochastic bin packing method to maximize the minimum utilization ratio, while satisfying the demands of virtual machines for both the resources. However, they have not considered the energy consumption and the heterogeneity of modern data centers. Also, the servers are assumed to have identical capacities. Verma et al. (2016) proposed a dynamic resource demand prediction and allocation framework by classifying tenants according to their changing resource requirements in multi-tenant service clouds. This model is useful for preparing the correct type of VMs in advance but does not minimize the energy consumption. Agrawal et al. (2015) modeled the energy-aware placement using linear programming technique to find the optimal solution. However, the computation time increases exponentially with increase in the number of virtual machines in Agrawal et al. (2015).

Quang-Hung et al. (2014) considered the problem of reducing energy consumption for high performance computing clouds and presented an energy-aware resource allocation considering multi-faceted resources, start time, and finish time of virtual machines. However, only the completion time of servers was optimized, without considering the system

resources and their utilization. Wang et al. (2016) used an integer bi-level genetic algorithm and defined a local search operator, reassigned map and reduce tasks to make energy-aware multi-job scheduling in cloud environment. However, authors have not considered the SLA violations and the computation complexity of the proposed approach is high as there are multiple levels of optimizations involved. Beloglazov et al. (2012) used a modified best fit decreasing (MBFD) algorithm to place virtual machines in power-efficient manner and proposed three heuristics for optimization of current allocation. However, MBFD does not always guarantee the optimal allocation and the proposed heuristics does not consider the memory and storage utilization. Jeyarani et al. (2011) proposed a self-adaptive PSO (SAPSO) for VM allocation in data centers. Though SAPSO performs well in terms of energy consumption, the analysis of SLA violations shows that this method does not maintain the quality of service when the VM requests increase. Dashti and Rahmani (2015) presented a modified PSO approach for allocation of migrated virtual machines from over-loaded hosts to improve energy efficiency in a cloud computing environment. But placing migrated VMs again in other hosts may lead to aggressive migration which causes high energy consumption and leads to SLA violations in a cloud computing environment. Kumar and Raza (2015) focused on minimizing the total resource wastage with efficient VM allocation using PSO in cloud operations. Wang et al. (2013b) proposed a PSO-based energy-aware VM placement optimization with local fitness first scheme. These two approaches have not considered over-utilization of the hosts which may lead to SLA violations. Ricciardi et al. (2011) developed a methodology that exploits load fluctuations and effectively control the system using service-demand matching algorithm and determines the subset of servers that may be powered off in data centers. However, their proposed method did not always guarantee the optimal solution and the authors have not considered the SLA violations.

Virtual machine migration is a major way for reducing unnecessary consumptions in a data center. Palmieri and Castagna (2007) proposed a swarm-based metaheuristic to the resource scheduling and balancing problem, minimizing runtime and fairly balancing load in next generation grids. However, they have not considered the energy consumption of computing resources of local grid nodes. Wang et al. (2013a) presented a decentralized double threshold VM selection policy considering the utilization of the physical nodes. However, they have not considered the energy consumption and this policy may not give the optimal solution always. Zhang et al. (2015) presented an approximate approach based on bin packing algorithm to migrate virtual machines. They considered the resource utilization and the migration cost to get the optimal solution. Bose et al. (2011) proposed an architecture for integrating replication and scheduling strategies to optimize the live migration costs across wide area networks. They replicated VM images across the cloud sites and the replica placement is done using CloudSpider. This method requires additional storage requirements. Beloglazov and Buyya (2012) presented a continuous online optimization of VM allocation and dynamic consolidation. Consolidation of virtual machines is performed after analyzing resource usage patterns by virtual machines for reducing energy consumption. However, the cost incurred by this optimal online algorithm was higher and aggressive consolidation may lead to performance degradation. Zhou et al. (2016) proposed an adaptive three-threshold energy-aware algorithm (ATEA) considering resource usage patterns of virtual machines. ATEA migrates virtual machines on over-loaded machines to lightly loaded machines which may not guarantee the optimal energy consumption of data center. A summary of the said literature is illustrated in Table 1.

In our work, an energy-aware virtual machine allocation algorithm together with a virtual machine selection scheme for migration is proposed. Discrete PSO (DPSO) is modified with the objective to minimize the increase in the energy consumption of the host while placing the virtual machine. Once we find the VM-physical machine map using the MDPSO algorithm, we perform the optimization of the current allocation using MBS-VM algorithm. The MBS-VM algorithm selects the virtual machines in either over-loaded or under-utilized hosts. To the best of our knowledge, this paper is a first proposal presenting energy-aware virtual machine allocation and selection in virtualized data centers, considering the relationship between performance and energy consumption of servers, while maintaining service-level agreements.

3 Resource allocation model in cloud data centers

Let M be the number of virtual machines and N be the number of physical hosts/physical machines (PM), respectively. V is defined as a set consisting of virtual machines, where v_i is an instance of a virtual machine and $|V| = \mathrm{M}$. P is a collection of physical machines, where p_j is an instance of a physical machine and $|P| = \mathrm{N}$.

$$V = \{v_1, v_2, \ldots, v_k\} \tag{1}$$

$$P = \{p_1, p_2, \ldots, p_t\} \tag{2}$$

In our model, each physical machine (p_j) is characterized as a 5 tuple:

$$p_j = (id_j, cpu_j, storage_j, bw_j, cores_j)$$

where id_j is the unique identity of the physical machine, cpu_j is the computing power of the physical machine generally

Table 1 Summary of the related work

Reference	VM allocation	VM selection	Methodology	Considered	
				Energy	Multiple resources
Wu et al. (2012)	Yes	No	Genetic algorithm	Yes	No
Younge et al. (2010)	Yes	No	Greedy algorithm	Yes	No
Jin et al. (2012)	Yes	No	Max–min multi-dimensional stochastic bin packing	No	Yes
Verma et al. (2016)	Yes	No	Exponential moving average, Polynomial regression, ARX, and ARMAX	No	No
Agrawal et al. (2015)	Yes	No	Linear programming	Yes	No
Quang-Hung et al. (2014)	Yes	No	MinDFT-ST and MinDFT-FT	No	Yes
Wang et al. (2016)	Yes	No	Integer bi-level genetic algorithm	Yes	Yes
Beloglazov et al. (2012)	Yes	Yes	Modified best fit decreasing	Yes	No
Jeyarani et al. (2011)	Yes	No	Self-adaptive PSO	Yes	No
Dashti and Rahmani (2015)	Yes	No	Modified PSO	Yes	No
Kumar and Raza (2015)	Yes	No	Particle swarm optimization	Yes	No
Wang et al. (2013b)	Yes	No	PSO-based local fitness first	Yes	No
Palmieri and Castagna (2007)	Yes	No	Ant-colony based metaheuristic	No	No
Wang et al. (2013a)	No	Yes	Two-threshold decentralized migration	No	No
Zhang et al. (2015)	No	Yes	Bin packing	No	Yes
Bose et al. (2011)	Yes	Yes	CloudSpider	No	No
Beloglazov and Buyya (2012)	Yes	Yes	Optimal online deterministic algorithm	Yes	No
Zhou et al. (2016)	Yes	Yes	Adaptive three-threshold algorithm	Yes	No
Kolodziej et al. (2015)	Yes	No	Genetic algorithm	Yes	No

given in million instructions per second (MIPS), $storage_j$ is the capacity of the random access memory of the physical machine, $cores_j$ is the number of cores in a physical machine and bw_j is the bandwidth that is allocated to a physical machine. Each virtual machine is characterized by a 4 tuple, given by:

$$v_i = (id_i, cpu_i, storgae_i, bw_i)$$

where id_i gives the unique ID of the VM. $cpu_i, storage_i$, and bw_i are the amount of processing power, memory, and bandwidth requested by the VM, respectively. In our allocation model, a physical machine can be assigned one or more virtual machines but each virtual machine must be assigned to only one physical machine. We use a variable y_{ij} to indicate whether a virtual machine is allocated to a physical machine or not. y_{ij} is 1, if ith virtual machine is allocated to the jth physical machine and 0 otherwise.

3.1 Problem definition

The problem of virtual machine allocation in a cloud data center is formulated as follows: Derive a mapping from the set of virtual machines (V), to the set of physical machines (P) that should maximize the resource utilization and minimize the energy consumption. Resource utilization of a physical machine can be decomposed into cpu utilization, memory utilization and bandwidth utilization by all the virtual machines in that host. v_i^{cpu}/p_j^{cpu} gives the cpu utilization of jth host (p_j) by ith virtual machine (v_i). v_i^{mem}/p_j^{mem} gives the memory utilization of jth host (p_j) by ith virtual machine (v_i). v_i^{bw}/p_j^{bw} gives the bandwidth utilization of jth host (p_j) by ith virtual machine (v_i). The objective of maximizing physical resource utilization is defined as follows:

$$Maximize \left\{ \frac{v_i^{cpu}}{p_j^{cpu}}, \frac{v_i^{mem}}{p_j^{mem}}, \frac{v_i^{bw}}{p_j^{bw}} \right\}$$

The allocation must satisfy the following constraints :

$$\forall i \sum_{j=1}^{m} y_{ij} = 1 \tag{3}$$

$$\forall j \sum_{i=1}^{n} y_{ij} \,.\, v_i^{cpu} \leq p_j^{cpu} \tag{4}$$

$$\forall j \sum_{i=1}^{n} y_{ij} \cdot v_i^{mem} \leq p_j^{mem} \tag{5}$$

$$\forall j \sum_{i=1}^{n} y_{ij} \cdot v_i^{bw} \leq p_j^{bw} \tag{6}$$

Equation (3) ensures that one virtual machine is allocated to only one physical machine though one physical machine may have more than one virtual machines. Equations (4), (5), and (6) check the resource requests for each physical machine and guarantee that they will not exceed the capacity of a physical machine. It is possible to define the upper and lower utilization limits for a physical machine to ensure reliability.

Generally, we find that the increasing energy consumption is primarily caused by idle or under-utilized servers. We model the energy consumption of a physical machine based on CPU utilization and it is better that the resource utilization is modeled by its CPU utilization among the other factors like memory, disk and network bandwidth (Lin et al. 2015). Modeling server power consumption based on CPU utilization is traditionally followed and demonstrated accuracy in predicting the power consumption for fixed workloads (Dasgupta et al. 2011). Thus, the utilization of jth physical machine by ith virtual machine is given as follows:

$$p_u^{j(i)} = \left\{ \begin{array}{ll} \left\lfloor \frac{v_i^{cpu}}{p_j^{cpu}} \right\rfloor * 100 & \text{if } y_{ij} = 1 \\ 0 & \text{otherwise.} \end{array} \right\} \tag{7}$$

The physical machine utilization is the sum of the utilization rates of all VMs in that physical machine (p_j), given by

$$p_j^u = \sum_{i=1}^{m} p_u^{j(i)} \tag{8}$$

3.2 Server power consumption model

The power consumption is generally calculated on the basis of the maximum and minimum energy consumption of the host (Beloglazov et al. 2012; Wang et al. 2013b). But it is important to understand how a server uses and consumes power under different loads. Power and performance characteristics of servers measured under different loads are presented in SPECpower_ssj®2008 benchmark (Gray et al. 2008). We found some deviations in energy consumptions of servers with varying utilization. We used SPEC proven methodologies for iteratively measuring the performance and using them to calculate energy consumption. The SPECpower_ssj®2008 provides the details of power consumption for servers from the idle state to the peak in 10% intervals. We presented this as an array

$$SP = (a_0, a_1, \ldots, a_i, a_{i+1}, \ldots, a_{10})$$

where $a_0, a_1, a_2, \ldots, a_{10}$ represents the power consumption when a physical machine is operated at 0%, 10%, 20%,…, 100% utilization, respectively. The estimated power consumption of a physical machine (p_j) with utilization 'u' is defined in Eq. 9.

$$E(p_j(u)) = a_i + (a_{i+1} - a_i) * (10 * u - i) \tag{9}$$

where $i = floor(10 * u)$ and $(a_{i+1} - a_i)$ represents the increase in energy consumption of the server when the utilization is increased from i to $i + 1$. So, the total energy consumption (TEC) of a data center is defined as follows:

$$TEC = \sum_{j=1}^{n} E(p_j(u)) \tag{10}$$

4 System architecture of EViMAS

The high-level architecture of our proposed energy-aware virtual machine allocation and selection (EViMAS) in cloud data centers is shown in Fig. 1. The problem involves placing the VMs to physical machines in energy-efficient manner, and optimizing of the current VM allocation. EViMAS integrates the following two modules :

1. *Placement optimizer* This module performs the VM provisioning for new requests and places the VMs to physical machines using a modified DPSO (See Sect. 5) subject to the resource and SLA constraints.
2. *Migration manager* This module performs real time monitoring considering the utilization of VMs and physical machine and checks whether the physical machines exceeds threshold or not. If any physical machine is over-loaded or under-utilized, then the migration manager selects some of the virtual machines using MBS-VM algorithm (See Sect. 6) and places them in other physical machines using the placement optimizer.

5 Energy-aware VM allocation using MDPSO

The problem of energy-aware VM allocation is a NP-hard problem with a large search space that leads to a combinatorial explosion of possible solutions (Michael and David 1979). From the literature (Kennedy and Eberhart 1997; Fernandez-Martinez and Garcia-Gonzalo 2011) we observe that the DPSO is a good algorithm to circumvent the combinatorial explosion and to solve several optimization problems in discrete-valued search space. Hence, we propose a modified DPSO with a novel encoding of the particles and

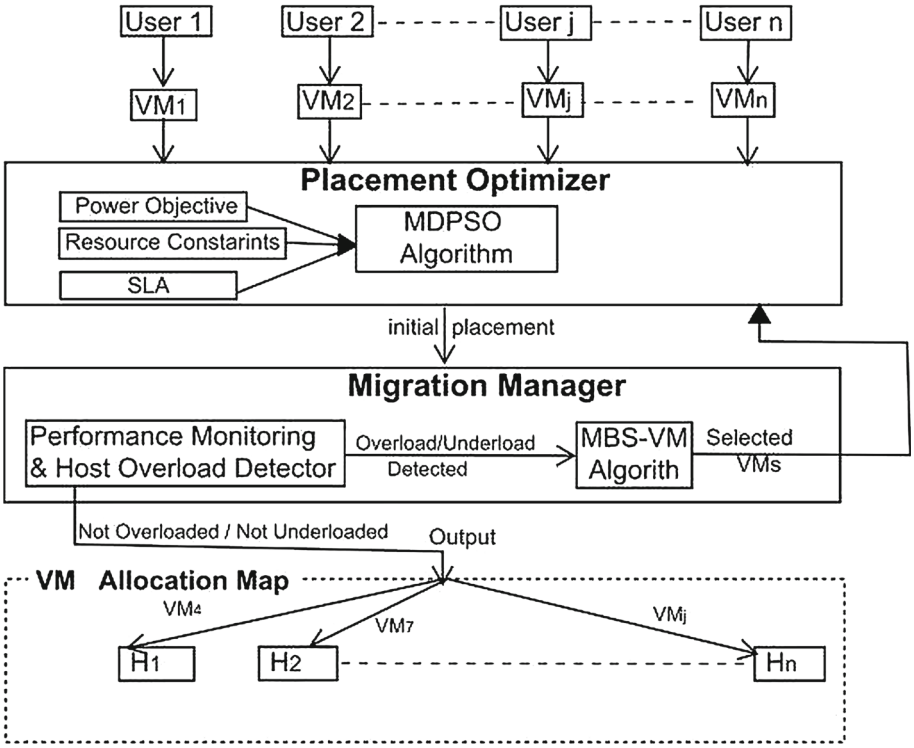


Fig. 1 System Architecture of EViMAS

operators that efficiently solves the large-scale VM allocation problem.

5.1 Fitness evaluation

Let "u" be the current utilization of a physical machine (p_j). A virtual machine v_i will be placed in p_j if the physical machine satisfies the conditions in Eqs. 3, 4, 5, and 6. Then the increase in the energy consumption of the physical machine is calculated using Eq. 11.

$$\delta P_j^i = E(p_j(u1)) - E(p_j(u)) \tag{11}$$

where '$u1$' is the utilization of the physical machine after v_i is placed in p_j and δP_j^i is the increase in the energy consumption of the physical machine p_j if the virtual machine v_i is allocated to the physical machine p_j.

In this work, we minimize the total data center energy consumption by optimizing the increase in power consumption while placing a virtual machine to a physical machine. Our objective is to minimize the δP_j^i when placing a VM into a physical machine, satisfying service-level agreements. In other words, if we allocate a virtual machine to a physical machine, then the workload can execute in the virtual machine with minimum energy consumption and good performance. Our aim is to minimize the fitness function (f) given as follows.

$$f = \sum_{j=1}^{n} \sum_{i=1}^{m} \delta P_j^i \tag{12}$$

5.2 Particle swarm optimization (PSO) and discrete PSO (DPSO)

PSO is a stochastic evolutionary computational technique, inspired by the social conduct in individuals and creatures particularly bird flocking (Parsopoulos and Vrahatis 2010; Kennedy 1997). PSO is a popular method to find optimum of a numerical function defined on a continuous domain. The algorithm is initialized with a population of particles, where the solution to the VM allocation problem is represented as particle ($X_i(t)$). Each particle is randomly positioned in the global search space and represents a candidate solution. Each particle remembers its current position, personal best achieved so far and moving velocity. For energy-aware VM allocation, a particle position yielding a smaller δP value is treated as an optimal solution. For each iteration, it calculates the fitness and velocities of each particle and flies in the search space toward the local and global best solutions in a navigated way. By its stochastic mechanism, PSO explores new areas to ensure not to fall into local optima. Starting with a random seed, it tries to find the optimal solution by memorizing the local (L_i) and the global best position ($\hat{L_j}$). The particle velocity is updated using initial velocity and two random weights that influence the movement of the particle as shown in Eq. 13.

$$V_i(t+1) = wV_i(t) + k_1 r_1(t)(L_i(t) - X_i(t)) + k_2 r_2(t)(\hat{L_j}(t) - X_i(t)) \quad (13)$$

Then the particle position is modified using the updated velocity as shown in Eq. 14.

$$X_i(t+1) = X_i(t) + V_i(t+1); \quad (14)$$

where w is the inertia weight coefficient, k_1 and k_2 are called the learning factor coefficients and r_1 and r_2 are the random numbers between 0 and 1.

DPSO is a version of classical PSO that is modified to work with discrete-valued search spaces consisting of binary/integer values (Kennedy and Eberhart 1997). DPSO shows discrete or subjective discrimination between variables (Chen et al. 2014). DPSO is widely adapted and used to solve several combinatorial optimization problems when a good specialized algorithm is not available. In DPSO, a particle is composed of binary or integer variables and velocity is converted to chance of probability. This algorithm efficiently maintains the divergence of the swarm.

5.3 VM allocation using MDPSO

We propose a modified discrete particle swarm optimization approach (MDPSO) to find optimal energy-aware virtual machine allocation. Particles are modeled to present a map of virtual machines to be placed in available physical machines in a data center. We defined a novel particle encoding scheme in MDPSO with modified operators and velocity updation.

5.3.1 Particle position and velocity

Assume that 'm' workloads are allocated to any of the 'n' heterogeneous physical machines. We define a "position (X)" as a list of 'm' possible values and each value is chosen from the set P given in Eq. 2. Every value in X represents a physical machine in a cloud data center chosen from the set P. Let $(v_1, v_2, v_3, \ldots, v_m)$ is the list of VMs to be placed in the available physical machines in a data center. Then the particle position is a list of size M that gives the corresponding allocation of each VM in an optimization period (t) given as follows.

$$X_i(t) = (p_1, p_5, p_1, \ldots, p_j, \ldots, p_n, \ldots, p_j)$$

The velocity gives the direction and tendency of a particle to move to a better position, shown in Eq. 15. We model the velocity of a particle as a set $V_i(t)$, containing elements such as $(p_i \rightarrow p_j)$ where each $p_i,\ p_j \in P$ and $|V_i(t)| =$M. In MDPSO, the particles use information of the local best and global best to adjust their velocities.

$$V_i(t) = \{(p_i \rightarrow p_j) \mid p_i,\ p_j \in P \text{ and } |V_i(t)| = \text{M}\} \quad (15)$$

5.3.2 Transition ($\odot$) operator

The transition operator is modeled to reflect the particle position ($X_i(t)$) updation based on its velocity. We use $\odot$ to denote the transition operator. A particle $X_i(t)$ uses a new velocity $V_i(t)$ to adjust its current position and builds a new position $X_i(t+1)$. Let p_i be the value from $X_i(t)$ and $(p_j \rightarrow p_k)$ be the corresponding velocity from $V_i(t)$ then $p_i \odot (p_j \rightarrow p_k)$ is defined as follows:

$$p_i \odot (p_j \rightarrow p_k) = \left\{ \begin{array}{ll} p_k & \text{if } p_i = p_j \\ p_i & \text{otherwise} \end{array} \right\} \quad (16)$$

If we apply $(p_i \rightarrow p_j)$ to evaluate a value different from p_i, the position remains the same.

5.3.3 Subtraction ($\ominus$) operator

The subtraction operator is defined to reflect the change between two particles, each representing a placement solution. As far as X_i, X_k is concerned $X_i \ominus X_k$ will give the velocity. In our modified approach, $X_i \ominus X_k$ is defined for each p_i, p_k as follows:

$$p_i \ominus p_k = \{p_k \rightarrow p_i\} \quad (17)$$

where p_i is the value from X_i and p_k is the corresponding value from X_k.

5.3.4 Addition ($\oplus$) operator

We define the addition operator ($\oplus$) to calculate the particle velocity updation, which depends on different factors including the local best of each particle, the global best of the swarm and velocity inertia of a particle. Between two velocities $V_i(t)$ and $V_j(t)$, the addition defines the updated velocity. Let $(p_i \rightarrow p_x)$, $(p_y \rightarrow p_j)$ be the velocity values from $V_i(t)$ and $V_j(t)$, respectively. $V_i(t) \oplus V_j(t)$ is defined as follows:

$$V_i(t) \oplus V_j(t) = \left\{ \begin{array}{ll} p_i \rightarrow p_j & \text{if } p_x = p_y \\ p_i \rightarrow p_x & \text{if } p_x \neq p_y \end{array} \right\} \quad (18)$$

5.3.5 Multiplication operator ($\otimes$)

We modify the multiplication operator to update the particle velocity when it is multiplied by a real positive coefficient. The $\otimes$ operator is stochastic and defined only for a coefficient between 0 and 1. We can perform 'n' addition operations and one multiplication operation if the coefficient is greater than

one. For each value $(p_i \rightarrow p_j)$ of $V_j(t)$, the multiplication operator is defined as follows:

$$c \otimes V_j(t) = \left\{ \begin{array}{ll} p_i \rightarrow p_i & \text{if } \hat{c} \leq c \\ p_i \rightarrow p_j & \text{if } \hat{c} > c \end{array} \right\} \quad (19)$$

where $c \in [0, 1]$ and $\hat{c} = random(0, 1)$.

Algorithm 1: VM Placement Using MDPSO

Input: List of virtual machines and physical machines
Initialize the parameters *k, r, s, population size* and learning factors (k_1, k_2).
Initialize the population using sub-random sequences.
foreach *particle* $\in$ *Swarm* **do**
  $fitness[i] = \sum_{j=1}^{m} E(p_j(u1)) - E(p_j(u))$.
  $L_i = X_i(t)$. // Local Best
$\hat{L_j}(t)$ = particle with minimum $\sum_{j=1}^{m} E(p_j(u1)) - E(p_j(u))$ in the swarm. // Global Best
prevFitness = fitness.
repeat
  foreach *particle* $\in$ *Swarm* **do**
    // Update Velocity and Position of each particle
    $V_i(t+1) = w \otimes V_i(t) \oplus k_1 r_1(t) \otimes (L_i(t) \ominus X_i(t)) \oplus k_2 r_2(t) \otimes (\hat{L_j}(t) \ominus X_i(t))$ $X_i(t+1) = X_i(t) \odot V_i(t+1)$;
    $fitness[i] = \sum_{j=1}^{m} E(p_j(u1)) - E(p_j(u))$.
    if *fitness[i]* < *prevFitness[i]* **then**
      $L_i = X_i(t+1)$.
  $\hat{L_j}(t+1)$ = particle with minimum $\sum_{j=1}^{m} E(p_j(u1)) - E(p_j(u))$ in the swarm.
  if *fitness* $(\hat{L_j}(t+1)) < fitness(\hat{L_j}(t))$ **then**
    $\hat{L_j}(t) = \hat{L_j}(t+1)$.
    Flag = True;
    count ++;
    if *Flag* == *True and count* > *r* **then**
      Return $\hat{L_j}(t)$.
  else
    if *fitness* $(\hat{L_j}(t+1)) == fitness(\hat{L_j}(t))$ **then**
      Flag = false;
      count1 ++;
      if *count1* > *s* **then**
        Return $\hat{L_j}(t)$.
    else
      Flag = false.
      count = *count1*= 0.
until *k-times*;

Using the said operations, we improve the DPSO by reformulating Eqs. 13 and 14 as follows:

$$V_i(t+1) = w \otimes V_i(t) \oplus k_1 r_1(t) \otimes (L_i(t) \ominus X_i(t)) \oplus k_2 r_2(t) \otimes (\hat{L_j}(t) \ominus X_i(t)); \quad (20)$$

$$X_i(t+1) = X_i(t) \odot V_i(t+1); \quad (21)$$

The algorithm for the allocation of virtual machines using our MDPSO is illustrated in Algorithm 1. The particle data in this case are the VM-physical machine map that shows the allocation of a VM to a physical machine. Initially all the VMs are placed randomly in different physical machines. We calculate the fitness of each particle in the swarm using Eq. 12. Then we choose the particle having a minimum fitness as the global best. After each iteration, all particles try to move closer to the coordinates of the optimal solution in the population which provides less energy consumption. This happens when we update the velocities of each particle using their local best and global best as shown in Eq. 20. These velocities are applied to the respective particles to update their position as shown in Eq. 21. Individual particles of the swarm are manipulated until the particle matches the target by updating the velocity and position. Algorithm 1 terminates either after k-iterations or achieves continuous improvement in the global best r-times.

6 VM selection and migration using MBS-VM

Live migration enables to respond to peak load periods by moving virtual machines from one physical machine to another physical machine. This technique is widely used to reduce under-utilization in data centers by dynamically migrating VMs to few host considering different thresholds and SLAs. Then, idle physical machines can be turned to sleep mode to improve the energy efficiency. Migration takes place between two physical machines. Once a new configuration file is created on the target physical machine, the selected VM memory is copied to the target physical machine. Meanwhile if any memory page is changed on the source, those pages will be tagged and copied. Migration helps performing maintenances without disrupting operations, optimizing resource pools and avoids failing.

We perform migration of the virtual machines to increase the efficiency and to ensure the SLA. We choose two extreme states of the servers for VM selection: over-utilization and under-utilization. If a server is under-utilized, then we select all the virtual machines in that physical machine. If a server is over-utilized, then we select the virtual machines for migration using the proposed selection algorithm until the physical machine utilization drops below the upper threshold.

We propose a MBS-VM selection algorithm considering the memory utilization, bandwidth utilization, and size of the virtual machine. The migration task adds load to both the target and host systems, which effects the performance of both the hosts as well as the data center. Hence, we carefully select the migratable virtual machines in the over-utilized hosts. In order to achieve this, we designed a cost function that effectively balances memory, bandwidth, and size fac-

tors. This function is further used to select a VM among the migratable VMs. The definitions for the utilization of various elements that we considered are discussed as follows:

Memory utilization Memory utilization of a VM reflects the usage of the physical machine by a VM. Generally, memory usage changes slowly when the host utilization is below the lower threshold and increases rapidly when approaching the upper threshold. Considering these facts, we define the memory utilization of a VM (v_i) in a host (p_j) as shown in Eq. 22.

$$VM_i\ (mm) = \frac{\text{Current allocated RAM for } v_i}{\text{Total RAM requested by the } v_i}. \tag{22}$$

Bandwidth utilization Virtual machine migration involves transferring large amounts of data between hosts. VMs comprising a multi-tier application make complex load interactions among the underlying physical servers. For such applications, we need to consider bandwidth allocation and usage for a VM as shown in Eq. 23

$$VM_i\ (bw) = \frac{\text{Current allocated bandwidth for the } v_i}{\text{Total bandwidth requested by the } v_i}. \tag{23}$$

VM size The amount of data transfer is directly related to the migration cost. The amount of data transfer is the only factor that has a linear relationship with the power consumption and also with migration time when physical machines have stable loads. In fact, the amount of data is controlled by the memory size of migratable VMs (Svärd et al. 2015). VM size is defined as follows:

$$VM_i\ (size) = \text{Size of the virtual machine } (v_i). \tag{24}$$

Based on the concept of main memory utilization, bandwidth utilization, and VM size, the cost function for a VM is defined as the weighted sum, as follows:

$$Cost\ (VM_i) = a.VM_i\ (mm) + b.VM_i\ (bw) + c.VM_i\ (size). \tag{25}$$

where a, b, and c are the random weights such that $a+b+c = 1$.

Algorithm 2 describes the process of virtual machine selection and migration using memory utilization, bandwidth utilization, and VM size. The MBS-VM algorithm calculates the cost of each migratable VM in a physical machine. It selects a virtual machine with the highest cost among the migratable VMs and then checks whether the selection of this VM leads to an aggressive consolidation which increases the possibility of over-utilization. If not, the algorithm adds this VM to the migration list. This process continues until the CPU utilization drops below the upper utilization threshold when the upper threshold is violated. If the physical machine is under-utilized, i.e., utilization falls below the lower threshold, it is better to put the physical machine to sleep mode. To achieve this, we migrate all VMs to another physical machine to eliminate the idle power consumption. If a VM is running in an over-utilized physical machine, the proposed algorithm automatically migrates the VM to a target host which gives the better performance. Once the process of VM selection using the MBS-VM algorithm is completed, the selected virtual machines are admitted for VM provisioning.

Algorithm 2: VM Selection Using MBS-VM

```
Input: List of physical machines: hostList
Initialize lowerThreshold, upperThreshold.
foreach host ∈ hostList do
    host-utilization= getHostUtilization (host);
    VMlist =host.getVmlist().
    if host-utilization >= upperThreshold then
        repeat
            foreach VM ∈ VMlist do
                Initialize a, b, and c with random value between
                0.0 and 1.0.
                total = a + b + c.
                a = a/total.
                b = b/total.
                c = c/total.
                Cost(Vm) =
                a.VM_i (mm) + b.VM_i (bw) + c.VM_i (size).
            Sort the VMlist in decreasing order of their Cost.
            probedVM = VM with highest cost.
            if probedVM.isAggersive() = FALSE and probedVM
            is not in migration then
                update host-utilization without probedVM.
                migrationList.add (probedVM)
                VMlist = VMlist − probedVM .
            else
                continue;
        until host-utilization < upperThreshold;
    if host-utilization <= lowerThreshold then
        migrationList.addAll (VMlist).
        Remove all the VMs form VMlist of the host.
return migrationList.
```

7 Performance evaluation

We compare our proposed approach with few other approaches concerning energy consumption, SLA violations, number of migrations and active hosts. We performed our experiments simulating a cloud data center using CloudSim (Calheiros et al. 2011). CloudSim follows a layered implementation structure and provides support for simulating cloud data center environments.

Table 2 Physical machines configurations

Host type	MIPS	RAM (GB)	Bandwidth (MBPS)	Storage (GB)
G4 Xeon 3040	3000	20	10	125
Xeon X 5675	3000	20	10	125
G4 Xeon 3075	3000	20	10	125

Table 3 Virtual machines requirements

VM type	MIPS	RAM (MB)	Bandwidth (MBPS)	Storage (GB)
Type 1	30	613	100	0.25
Type 2	40	870	100	0.25
Type 3	50	1740	100	0.25

7.1 Experimental environment

In our experiments, we simulated a cloud data center comprising 100 heterogeneous physical machines, characterized according to their configurations as shown in Table 2. Every VM component in CloudSim has the following characteristics related to a VM: accessible memory (RAM), processor (MIPS), storage size, , bandwidth, and the VMs internal provisioning policy. Further, these resource requests are of heterogeneous nature and VMs vary from micro-instances (30 MIPS, 600 MB RAM) to large instances (50 MIPS, 1740 MB RAM). The characteristics for each type of VM are presented in Table 3. The broker/user submits requests for provisioning of VMs. Each virtual machine processes time-varying workload. This workload is designed in such a way that CPU is loaded according to a uniformly distributed random variable. This load runs for 4,32,000 million instructions which is equal to running a machine with 300 MIPS processing power for 24 min with 100% utilization. We assume the idle conditions of the host upon starting. All the programs are written in Java language, and simulations have been executed in a 64-bit Windows 8 Operating System running on an Intel Core i5-3230M 2.60GHz HP ProBook with four cores and 4 GB of RAM.

Tests were performed on the same simulation platform to set the initial parameters for MDPSO. Initial populations sizes of 10, 20 and 30 fall to a local optimum, regardless of the swarm behavior, whereas the population size of 50 needs more iterations. So, the population size of swarm is set to 40 which has a linear convergence rate with increasing number of VMs and providing optimal solution. From our experiments, we observed that the maximum number of iterations above 150 ensures a global optimum solution without falling to local optimum. So, the maximum number of iterations is set to 150. Learning parameters (k_1, k_2) are set to 3, 2, respectively, to prevent divergence of the particles. These values are chosen after several experiments, and the way these weights coordinate during the searching process after each iteration is illustrated in "Appendix."

7.2 Analysis of VM allocation approaches

We evaluated the performance of our proposed approach by comparing the solutions of Beloglazov et al. 2012, Wu et al. (2012), Kumar and Raza (2015), and Dashti and Rahmani (2015). The ultimate goal of any VM placement algorithm is to reduce the number of active hosts thereby reducing the energy consumption of the data center. We evaluated the performance of our approach with 20 physical machines and 50 virtual machines. We found that the proposed approach places 14 virtual machines into the first physical machine, and the remaining VMs are distributed to the other physical machines thereby reducing the number of active physical machines to 9. Figure 2 illustrates the number of virtual machines placed in each of the physical machines by the said algorithms. The number of active physical machines by Beloglazov et al. (2012), Wu et al. (2012), Kumar and Raza (2015) and Dashti and Rahmani (2015) are 15, 14, 12, and 12, respectively. The proposed approach uses the minimum number of physical machines while satisfying the resource requirements of VMs. Thus, the idle hosts can be switched off to improve the energy efficiency further.

In an environment of 300 virtual machines, the MDPSO leads to the minimum energy consumption of 2.25 kWh where Beloglazov et al. (2012), Wu et al. (2012), Kumar and Raza (2015), and Dashti and Rahmani (2015) consume 3.15, 2.92, 2.47, and 2.50 kWh, respectively. Thus, in our proposed approach, the energy consumption is decreased 32% in comparison with Beloglazov et al. (2012), and 25% in comparison with Wu et al. (2012). We tested with the same setup varying the number of VMs, starting form 50–300. The results are presented in Table 4.

7.3 Analysis of VM selection algorithms

We analyzed our proposed MBS-VM selection algorithm in terms of energy consumption and the number of migrations occurred for the given setup. We compared our proposed approach with the Minimization of Migrations (MM) pol-

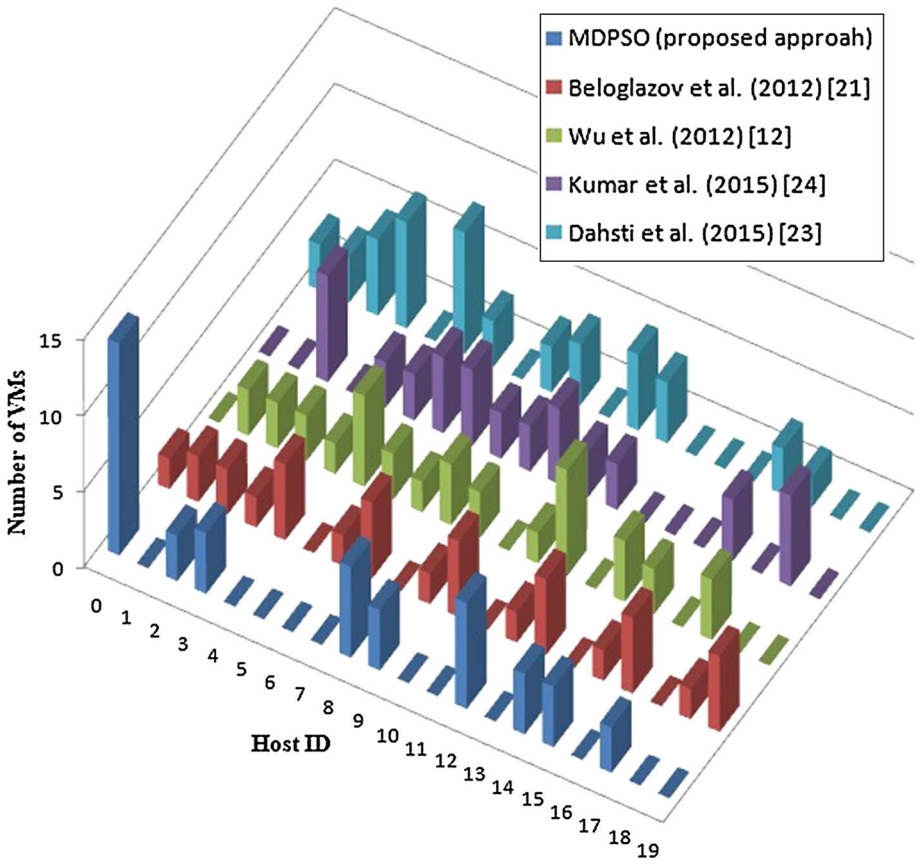


Fig. 2 Comparison of active hosts in different algorithms

Table 4 Performance comparison in terms of energy consumption

VMs	Energy consumption (kWh)				
	MDPSO (proposed)	Beloglazov et al. (2012)	Wu et al. (2012)	Kumar and Raza (2015)	Dashti and Rahmani (2015)
50	0.91	1.10	1.09	0.92	0.92
100	1.01	1.52	1.31	1.03	1.20
150	1.30	2.02	1.82	1.40	1.60
200	1.45	2.35	1.89	1.65	1.75
250	1.70	2.64	2.50	1.92	1.97
300	2.25	3.15	2.93	2.47	2.50

icy (Beloglazov et al. 2012) and the Minimum Memory Size (RAM) policy (Zhou et al. 2016).

The MM policy selects the minimum number of virtual machines to avoid a host being over-utilized using thresholds. The RAM policy selects a VM with the minimum memory size in a over-utilized host which in turn gives less migration time under the same spare network bandwidth. The comparison of these algorithms in a data center with 300 virtual machines and 100 hosts is presented in Fig. 3. The energy consumption by the data center using MBS-VM, RAM, and MM algorithms are 2.25, 3.30, and 3.87 kWh, respectively. In our algorithm, a, b, and c values are chosen randomly. We have simulated this experiment for 20 runs and the mean values of a, b, c are reported as follows: $a = 0.37464$, $b = 0.28705$, and $c = 0.33817$. Figure 3 shows that our proposed approach performs better than the MM policy and the RAM policy with the increase in the number of virtual machines.

7.4 Performance comparison in terms of migrations

Excessive virtual machine migrations in a data center could affect the desired QoS level under peak loads and could increase the energy consumption of a data center. In Fig. 4, we present a comparison of our proposed approach with different VM allocation policies in a data center with 100 physical

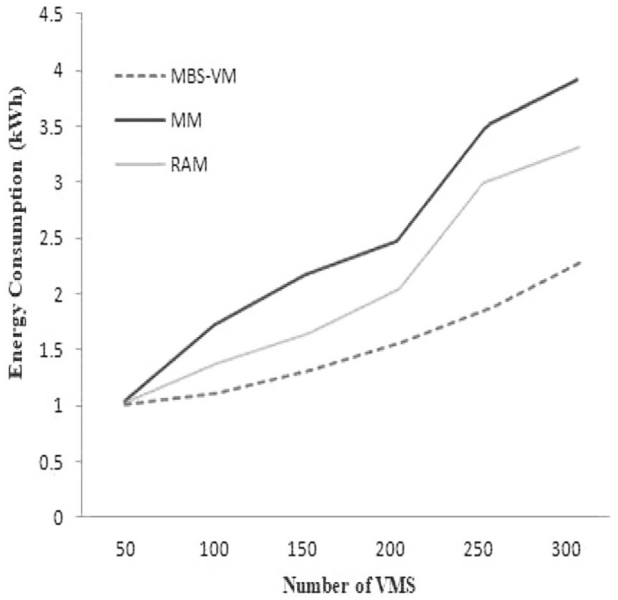


Fig. 3 Comparison of VM selection algorithms

machines. The comparison is in terms of the number of migrations with increasing virtual machines. From the experiments, we observed that the number of migrations for all the algorithms is almost equal in case of 50 virtual machines. But with the increase in number of virtual machines, the proposed approach outperformed the other approaches. Our proposed approach entails the minimum number of migrations with 50, 100, 150, 200, 250, and 300 virtual machines on 100 physical machines.

7.5 SLA violations

Another important aspect to ensure the smooth operations of a data center environment is to guarantee the service-level agreements (SLA) in terms of satisfying QoS requirements such as availability, reliability, and throughput. SLA violations are determined as follows :

$$\text{SLA violation} = \frac{(\text{Requested resource} - \text{Allocated resource})}{\text{Requested resource}} \tag{26}$$

The SLA is violated, if a physical machine fails to allocate the requested processor share to a VM. In our model, we calculate the SLA violation for each VM request using Eq. 26. Average SLA violations as well as resource requests that were not allocated are determined by CloudSim and the results are presented in Table 5. The average SLA violations for 50 and 100 virtual machines remain mostly same for all the approaches. However, our proposed MDPSO demonstrates less average violations with the increase in the number of virtual machines (See Table 5).

7.6 Convergence analysis

The scalability of our approach is specified in terms of the time to converge and the space complexity of MDPSO allocation and MBS-VM selection algorithms. These algorithms are influenced by three factors: number of VMs (M), phys-

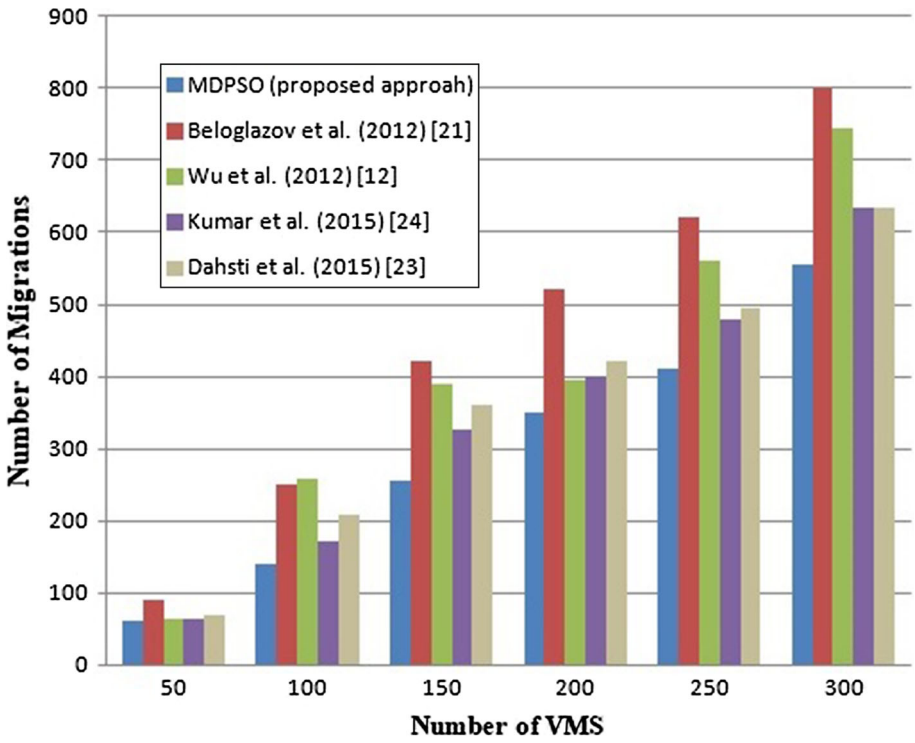
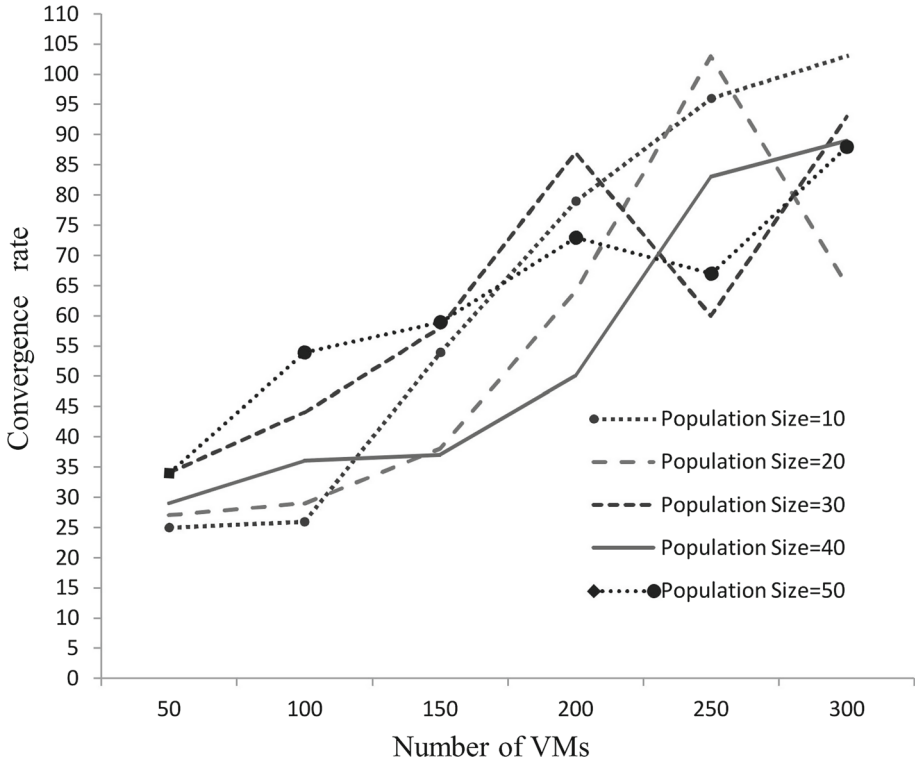


Fig. 4 Number of migrations versus number of virtual machines

Table 5 Average SLA violations

VMs	Average SLA violations (%)				
	MDPSO (proposed)	Beloglazov et al. (2012)	Wu et al. (2012)	Kumar and Raza (2015)	Dashti and Rahmani (2015)
50	9	9	9	9	9
100	9	9.25	9	9.1	9
150	9	10	10	9	91
200	9.13	10.56	10.05	9.22	9.45
250	9.51	11.72	11.47	9.75	9.84
300	11.26	13.25	12.54	11.89	12.41

Fig. 5 MDPSO convergence



ical machines (N) and number of migratable VMs. Starting with a swarm of size S and repeating the procedure for K times, to reach the global optimum solution, the computational complexity of the MDPSO algorithm is in the order of O(SK) (Parsopoulos and Vrahatis 2010).

The time complexity of the MBS-VM algorithm is proportional to the product of the number of over-utilized physical machines and the number of migratable VMs in each of these physical machines. In Fig. 5, we plot the convergence of the MDPSO algorithm (in terms of number of iterations it took to reach the global optimum solution) for varying population size, ranging from 10 to 50. From our experiments, we observe that our approach converges to global optima quickly, maintaining a linear relationship with increasing virtual machines for population sizes of 10 and 40. Hence, we have chosen 40 as population size for our experiments.

In our proposed MDPSO approach, we modeled each particle as a M-dimensional vector, so that the search space is limited to I^M where I is the population size. We plot the maximum number of iterations for converging to the global optimum with virtual machines ranging from 50 to 300 in Fig. 5. We can see that complexity of our approach is less than the linear time complexity (100 in this case).

8 Solving six-hump camelback function by MDPSO

To further illustrate the performance of our proposed MDPSO algorithm, we have solved the six-hump Camelback function, a benchmark test function for optimization problems. This is a simple 2D multimodal function that depends on two variables X_1 and X_2. The search space and schematic plot of

Table 6 Number of evaluations required to find global optimum

	Mean		Standard deviation	
	Pop = 30	Pop = 60	Pop = 30	Pop = 60
MDPSO	2098	2292	2585	1912
ANPSO	2798	4569	857	1316
SPSO	2872	5820	827	1469

function values can be found in Floudas et al. (2013). Six-hump Camelback function is generally considered for testing the performance of evolutionary algorithms. This test function has different aspects of the global optimization problem. However, the continuous variables are considered in discrete space here. The details of the problem are presented as follows.

$$f(X) = (4 - 2.1 * X_1^2 + X_1^4/3) * X_1^2 + X_1 * X_2 + (-4 + 4 * X_2^2) * X_2^2 \quad (27)$$

where $-1.9 \leq X_1 \leq 1.9$, $-1.1 \leq X_2 \leq 1.1$.

The function is symmetric about the origin and has six local optima two of which are global optima that are located at $(-0.08984, 0.71266)$ and $(0.08984, -0.71266)$. Due to many local optima in the search space, the result may depend on the initialization of the population. Thus, we can get an optimal solution that can vary within a given threshold from $fmin(X^*) = -1.0316285$.

For this test problem, using MDPSO algorithm a total of 20 runs are executed with a population sizes of 30 and 60 with $k_1 = 3$ and $k_2 = 2$ (chosen after experiments). In each run, the stopping criterion was that at least one particle would carry the global minima which is -1.0316 or it should be less than -1.0315, i.e., the best particle has a fitness near to the known global best fitness with an acceptance threshold (ϵ) of 0.0001. All the programs are written in Java language and executed in a 64-bit Windows 8 Operating System running on an Intel Core i5-3230M 2.60GHz HP ProBook with four cores and 4 GB of RAM.

We got 100% success rate with the goal of reaching the stopping criteria, although the required number of function evaluations varied between the runs. The mean and standard deviations of number of function evaluations taken over 20 runs are shown in Table 6.

We compare our proposed MDPSO method with speciation-based PSO (SPSO) (Li 2004) and adaptive niching PSO (ANPSO) (Bird and Li 2006). For comparison, the same test function is used and a total of 20 runs was made for each algorithm with an initial population sizes of 30 and 60. The maximum niche size is chosen to be greater than or equal to 3. In each run, the same stopping criterion is used. Table 6 summarizes the comparisons between MDPSO, ANPSO, and SPSO for the test function. The mean value in Table 6 represents the average number of function evaluations required to find the global optimum and the standard deviation represents how much each run differs from the mean value. It is clear that ANPSO and SPSO require more number of evaluations than MDPSO to solve the six-hump Camelback function. Our results demonstrate that the MDPSO can solve difficult test functions more reliably and with fewer evaluations. However, the standard deviation for MDPSO is high because of randomness in selection of initial population.

9 Conclusions

A data center is equipped with several thousands of physical machines, each of them handling several requests for virtual machines. An optimal resource allocation strategy for a data center helps achieving improved utilization of the data center. Our proposed MDPSO approach achieved high utilization of physical machines by decreasing the number of active physical machines. We have proposed a VM selection method for efficiently choosing virtual machines when a physical machine is either over-utilized or under-utilized. The experimental results illustrate that the combination of proposed allocation and selection algorithms leads to significant reduction in energy consumption in a cloud data center up to 32%. In our future work, we plan to continue our research in optimal allocation and selection of virtual machines to improve the energy efficiency of data centers considering I/O- and CPU-intensive workloads.

Acknowledgements This research received funding from the Netherlands Organization for Scientific Research (NWO) in the framework of the Indo Dutch Science Industry Collaboration programme in relation to project NextGenSmart DC (629.002.102). We thank Prof. Marco Aiello, University of Groningen, Netherlands for his useful insights and comments. We thank reviewers for their valuable and useful suggestions for the improvement of this article.

Compliance with ethical standards

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Conflict of interest V. Dinesh Reddy, G. R. Gangadharan and G. Subrahmanya V. R. K. Rao declares that they have no conflict of interest.

10 Appendix: Coordination of the particles

We simulated a data center comprising 100 heterogeneous physical machines and 300 virtual machines in the said experimental environment with the following initial parameters:

Population size = 40,

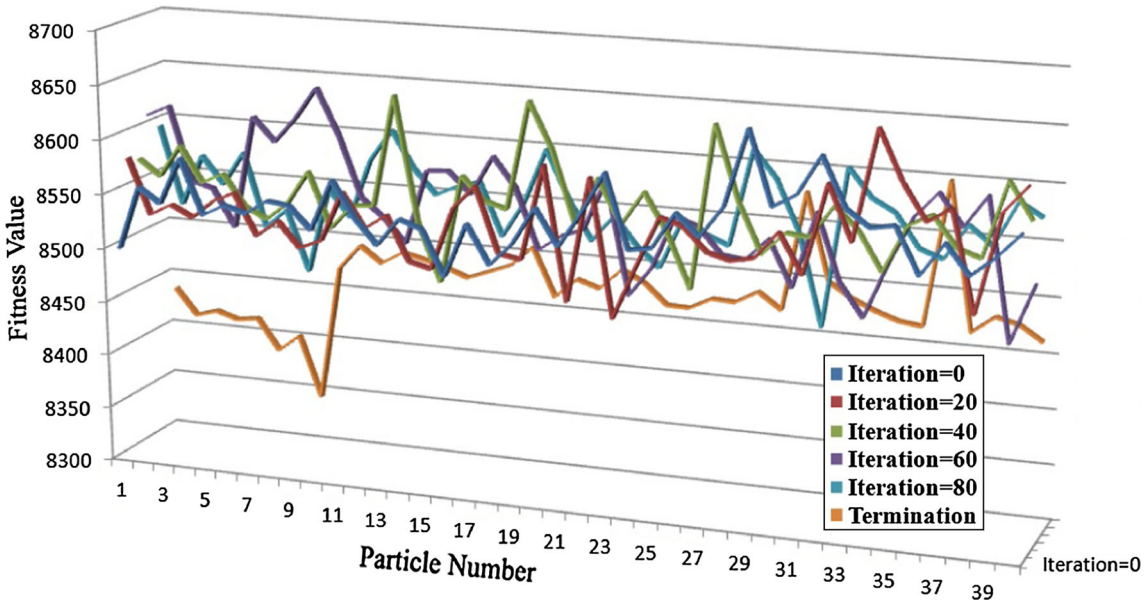


Fig. 6 Particle coordination

Inertia weight coefficients: $k_1 = 3$ and $k_2 = 2$.

These values are chosen after several experiments and the way these weights coordinate the searching process after each iteration is presented here. We presented the change in the fitness value of each particle, starting from the first iteration to termination with an interval of 20 in Fig. 6.

References

Agrawal P, Borgetto D, Comito C, Da Costa G, Pierson JM, Prakash P, Rao S, Talia D, Thiam C, Trunfio P (2015) Scheduling and resource allocation. In: Pierson JM (ed) Large-scale distributed systems and energy efficiency: a holistic view. Wiley, New Jersey, pp 225–262

Belady CL (2007) In the data center, power and cooling costs more than the it equipment it supports. Electron Cool 13(1):24

Beloglazov A, Buyya R (2012) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr Comput Pract Exp 24(13):1397–1420

Beloglazov A, Abawajy J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener Comput Syst 28(5):755–768

Bird S, Li X (2006) Adaptively choosing niching parameters in a PSO. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. ACM, London pp 3–10

Blackwell TM (2005) Particle swarms and population diversity. Soft Comput 9(11):793–802

Bose SK, Brock S, Skeoch R, Rao S (2011) CloudSpider: combining replication with scheduling for optimizing live migration of virtual machines across wide area networks. In: Proceedings of IEEE/ACM international symposium on cluster, cloud and grid computing (CCGrid). IEEE, Washington, pp 13–22

Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41(1):23–50

Chen CL, Huang SY, Tzeng YR, Chen CL (2014) A revised discrete particle swarm optimization algorithm for permutation flow-shop scheduling problem. Soft Comput 18(11):2271–2282

Curry E, Hasan S, White M, Melvin H (2012) An environmental chargeback for data center and cloud computing consumer. In: Proceedings of first international workshop on energy efficient data centers. Springer, Berlin, pp 117-128

Dasgupta G, Sharma A, Verma A, Neogi A, Kothari R (2011) Workload management for power efficiency in virtualized data centers. Commun ACM 54(7):131–141

Dashti SE, Rahmani AM (2015) Dynamic VMs placement for energy efficiency by PSO in cloud computing. J Exp Theor Artif Intell 28:97–112

Fernandez-Martinez JL, Garcia-Gonzalo E (2011) Stochastic stability analysis of the linear continuous and discrete PSO models. IEEE Trans Evol Comput 15(3):405–423

Floudas CA, Pardalos PM, Adjiman C, Esposito WR, Gms ZH, Harding ST, Klepeis JL, Meyer CA, Schweiger CA (2013) Handbook of test problems in local and global optimization, vol 33. Springer, Berlin, pp 111–113

Gandhi A, Harchol-Balter M (2011) How data center size impacts the effectiveness of dynamic power management. In: Proceedings of 49th annual allerton conference on communication, control, and computing. IEEE, Washington, pp 1164–1169

Garg SK, Toosi AN, Gopalaiyengar SK, Buyya R (2014) SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter. J Netw Comput Appl 45:108–120

Gray L, Kumar A, Li, H (2008) Characterization of SPECpower_ssj2008** benchmark. In: Proceedings of SPEC benchmark workshop. www.spec.org

Hu J, Phung-Duc T (2015) Power consumption analysis for data centers with independent setup times and threshold controls. In: Proceedings of the international conference on numerical analysis and applied mathematics (ICNAAM-2014), vol 1648. American Institute of Physics (AIP) Publishing, eid 170005

Jeyarani R, Nagaveni N, Ram RV (2011) Self adaptive particle swarm optimization for efficient virtual machine provisioning in cloud. Int J Intell Inf Technol 7(2):25–44

Jin H, Pan D, Xu J, Pissinou N (2012) Efficient VM placement with multiple deterministic and stochastic resources in data centers. In: Proceedings of global communications conference (GLOBECOM). IEEE, Washington, pp 2505–2510

Kennedy J (1997) The particle swarm: social adaptation of knowledge. In: Proceedings of IEEE international conference on evolutionary computation. IEEE, Washington, pp 303–308

Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: Proceedings of IEEE international conference on systems, man, and cybernetics. Computational cybernetics and simulation, vol 5. IEEE, Washington, pp 4104–4108

Kolodziej J, Khan SU, Wang L, Zomaya AY (2015) Energy efficient genetic-based schedulers in computational grids. Concurr Comput Pract Exp 27(4):809–829

Kumar D, Raza Z (2015) A PSO based VM resource scheduling model for cloud computing. In: Proceedings of IEEE international conference on computational intelligence and communication technology (CICT). IEEE, Washington, pp 213–219

Li X (2004). Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization. In: Proceedings of genetic and evolutionary computation conference. Springer, Berlin, pp 105–116

Lin W, Xu S, Li J, Xu L, Peng Z (2015) Design and theoretical analysis of virtual machine placement algorithm based on peak workload characteristics. Soft Comput 21:1301–1314

Maguluri ST, Srikant R, Ying L (2012) Stochastic models of load balancing and scheduling in cloud computing clusters. In: Proceedings of INFOCOM. IEEE, Washington, pp 702–710

Michael RG, David SJ (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman & Co., New York

Mohamed MSP, Swarnammal SR (2016) An efficient framework to handle integrated VM workloads in heterogeneous cloud infrastructure. Soft Comput 21(12):3367–3376

Negru C, Mocanu M, Cristea V, Sotiriadis S, Bessis N (2016) Analysis of power consumption in heterogeneous virtual machine environments. Soft Comput 21(16):4531–4542

Palmieri F, Castagna D (2007) Swarm-based distributed job scheduling in next-generation grids. Advances and innovations in systems, computing sciences and software engineering. Springer, Berlin, pp 137–143

Parsopoulos KE, Vrahatis MN (2010) Particle swarm optimization and intelligence: advances and applications. IGI Global, Hershey, pp 1–328

Pernici B, Aiello M, Brocke V, vom Brocke J, Donnellan B, Gelenbe E, Kretsis M (2012 What IS can do for environmental sustainability: a report from CAiSE11 panel on green and sustainable IS. Commun Assoc Inf Syst 30, Article 18

Quang-Hung N, Le DK, Thoai N, Son NT (2014) Heuristics for energy-aware VM allocation in HPC clouds. In: Proceedings of international conference on future data and security engineering. Springer, Berlin, pp 248–261

Ricciardi S, Careglio D, Sole-Pareta J, Fiore U, Palmieri F (2011) Saving energy in data center infrastructures. In: Proceedings of international conference on data compression, communications and processing. IEEE, Washington, pp 265–270

Shi W, Hong B (2011) Towards profitable virtual machine placement in the data center. In: Proceedings of fourth IEEE international conference on utility and cloud computing (UCC). IEEE, Washington, pp 138–145

Svärd P, Hudzia B, Walsh S, Tordsson J, Elmroth E (2015) Principles and performance characteristics of algorithms for live VM migration. ACM SIGOPS Oper Syst Rev 49(1):142–155

Valle YD, Venayagamoorthy GK, Mohagheghi S, Hernandez JC, Harley RG (2008) Particle swarm optimization: basic concepts, variants and applications in power systems. IEEE Trans Evol Comput 12(2):171–195

Verma M, Gangadharan GR, Narendra NC, Vadlamani R, Inamdar V, Ramachandran L, Calheiros RN, Buyya R (2016) Dynamic resource demand prediction and allocation in multitenant service clouds. Concurr Comput Pract Exp 28(17):4429–4442

Vomlelov M, Vomlel J (2003) Troubleshooting: NP-hardness and solution methods. Soft Comput 7(5):357–368

Wang X, Liu X, Fan L, Jia X (2013a) A decentralized virtual machine migration approach of data centers for cloud computing. In: Mathematical problems in engineering, Hindawi Publishing Corporation, Cairo

Wang S, Liu Z, Zheng Z, Sun Q, Yang F (2013b) Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers. In: Proceedings of international conference on parallel and distributed systems (ICPADS). IEEE, Washington, pp 102-109

Wang X, Wang Y, Cui Y (2016) An energy-aware bi-level optimization model for multi-job scheduling problems under cloud computing. Soft Comput 20(1):303–317

Wu G, Tang M, Tian Y, Li W (2012) Energy-efficient virtual machine placement in data centers by genetic algorithm. In: proceedings of international conference on neural information processing. Springer, Berlin, pp 315–323

Younge AJ, Laszewski GV, Wang L, Lopez-Alarcon S, Carithers W (2010) Efficient resource management for cloud computing environments. In: Proceedings of international green computing conference. IEEE, Washington, pp 357–364

Zhang X, Li K, Zhang Y (2015) Minimum-cost virtual machine migration strategy in data center. Concurr Comput Pract Exp 27(17):5177–5187

Zhou Z, Hu Z, Li K (2016) Virtual machine placement algorithm for both energy-awareness and SLA violation reduction in cloud data centers. Scientific Programming, Hindawi Publishing Corporation, Cairo