Journal of Network and Systems Management (2020) 28:1316–1342
https://doi.org/10.1007/s10922-020-09538-9

Modified Dragonfly Algorithm for Optimal Virtual Machine Placement in Cloud Computing

Atul Tripathi[1] · Isha Pathak[2] · Deo Prakash Vidyarthi[3]

Received: 12 July 2019 / Revised: 8 May 2020 / Accepted: 14 May 2020 / Published online: 26 May 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

The ease and affordability offered by the cloud computing has attracted large number of customers towards it. Cloud service providers offer its services, to the cloud customers, usually in form of Virtual Machines (VMs). With the growth in the number of customers, cloud data centers encounter overwhelming number of VM requests. These requests need to be mapped on the real cloud hardware and therefore, VM placement has been an important research area in the cloud research community. Virtual machine placement, being an NP hard problem, is modelled as an optimization problem with the objective to optimize resource wastage. Dragonfly Algorithm (DA), a nature inspired technique, originates from static and dynamic swarming behavior of dragonfly and is well suited to solve VM placement problem. Therefore, in the proposed work, a modified dragonfly algorithm is applied for VM placement for better resource utilization at cloud data centers. The performance of the proposed model is analyzed through simulation and comparative study. Observations, obtained from the experiments, exhibit the superiority of the proposed model in solving VM placement problem.

Keywords Binary dragonfly algorithm (BDA) · Virtual machine (VM) · Cloud data center (CDC) · Cloud service provider (CSP) · NP-hard problem · Resource utilization · Information technology (IT)

✉ Deo Prakash Vidyarthi
dpv@mail.jnu.ac.in

Atul Tripathi
atultripathi22@yahoo.com

Isha Pathak
ishaapathak@yahoo.in

1 Mahatma Gandhi Central University, Bihar, India

2 Indian Institute of Information Technology, Kota, Rajasthan, India

3 Jawaharlal Nehru University, New Delhi, India

1 Introduction

Newer technologies are being introduced in the modern technological word with rapid pace. Cloud computing is one such technologies, which is able to dominate the IT world during the last decade. Cloud has changed the manner of resource usage i.e. hardware, software and other general purpose tools used by an individual as well as by industry. Basically, cloud computing provides the IT services to its variety of clients through its large data centers and the internet [1, 2]. The cloud technology comprises of two parts: front-end and back-end. Front end deals with the users' requirement to access the cloud services such as interface, browser, connection etc., whereas back end considers the requirement of the system such as servers, hosts, networks etc. Of the offered cloud services, Infrastructure as a service (IaaS) is facilitated mainly with the help of Virtual Machines (VMs) which gives the ease of its configuration. A suitable VM placement algorithm, eventually maps these VMs to the Physical Machines (PMs) of a cloud data center. Technically, these data centers are supposed to provide infinite computational resources such as CPU, memory, storage, network etc. However, in reality, the cloud resources are finite. A good VM placement algorithm optimally places the VMs on to PMs to satisfy both; cloud service provider (CSP) as well as cloud customer. The CSP aims to maximize its revenue and minimize its operational cost [3] such as resource wastage, power consumption, thermal emission, physical infrastructures etc. Whereas, a cloud customer often aims to minimize its payment and avail best services. To minimize the resource wastage is an important research issues as it has direct impact on the operational cost of cloud data centers. As VM placement procedure is complex and time consuming, an automation is warranted. A bit of mismanagement may lead to application interruption.

The VM placement problem can be modelled as an optimization problem with multiple objectives. Out of these, higher resource utilization with minimization in the operational cost is most essential one. In the proposed work, VM placement problem is modelled as an optimization problem with resource wastage as an objective. This objective, in fact, recurs to many other objectives of the cloud resources. If the resource wastage is too high, it will lead to a greater number of active PMs in the data center resulting not only in high energy consumption but more physical space and cooling appliances too. In recent, energy has been given due consideration for holistic development. More active PMs and more cooling appliances are key contributors to higher operational cost of cloud data centers. Therefore, in the proposed work resource wastage is chosen as an objective and the problem is formulated as a minimization problem. Resource wastage,in cloud data center, can mathematically be formulated with the capping on the upper threshold of the respective resource. As VM placement problem is NP hard [4], therefore, it is difficult to find a feasible deterministic algorithm for this problem. A meta-heuristic is more suited for this and so in the proposed model, a bio-inspired algorithm called Dragonfly Algorithm (DA) has been applied to solve the VM placement problem. DA is based on the swarming behavior of the dragonfly.

The major contributions, of the proposed model, are as follows:

- VM placement problem is modelled as an optimization problem with the most important objectives of resource utilization.
- CPU and Memory are the resources under consideration for the problem formulation of resource utilization.
- A V shaped time varying transfer function is incorporated in the algorithm for a good balance between exploration and exploitation.
- Extensive experimentation is done for the analysis and comparative study.
- The performance of the proposed algorithm is evaluated for small as well as large set of VMs.
- Simulation is done on real and synthesized data.

The outline of the paper is as follows. Section 2 briefs some recent relevant literature on VM placement problem. In Sect. 3, description of VM placement problem and basic Dragonfly algorithm are presented. It also includes the mathematical formulation of VM placement problem as an optimization problem with resource wastage as an objective. The proposed model is presented in Sect. 4 whereas experiment details with comparative performance analysis are provided in Sect. 5. Finally, the conclusion of the proposed work with its future scope is laid down in Sect. 6.

2 Related Work

VM placement has been proven to be one of the emerging research area attracting a large research community [4–9]. This section deals with the recent related methodologies, applied successfully to solve the VM placement problem. These related works have been categorized as follows: heuristic bin packing [4–12], linear programming [6, 13], simulated annealing optimization [14], constraint programming [15], and bio-inspired optimization [16–21].

2.1 Heuristic Bin Packing

VM placement problem is modelled as vector bin packing problem which is a well-known NP-hard problem [4]. Various heuristic methods such as greedy approaches are applied to find the near optimal solution for this NP-hard problem. Grit et al. [5] have applied worst fit and best fit algorithms for solving similar types of problems for network resources. Later, Speitkamp et al. [6] and Bichler et al. [8] extended this concept by applying heuristics such as first fit decreasing (FFD) and best fit decreasing (BFD) to improve the results. Verma et al. [10] have further extended the FFD heuristic and proposed a placement mechanism that caters the power cost as well as the migration cost. Cardosa et al. [7] have also modified first fit and best fit algorithms to incorporate node utility in VM placement problem. Srikantaiah et al. [9] have proposed profile data based consolidation for cloud data center that incorporates the worst fit strategy. Few other works [11, 22, 23] are also based on heuristic strategies.

2.2 Linear Programming

In linear programming, VM placement problem is modelled as simple bin packing problem with linear relationship in objectives and decision variables. Speitkam et al. [6] have formulated VM placement and server consolidation problem as a bin packing problem. A heuristic of LP-relaxation based technique is adopted to minimize the cost. Lin et al. [13] have formulated VM consolidation problem as an integer linear programming and proposed a novel polynomial time heuristic algorithm. In order to reduce the complexity of integer linear programming problem and improvising the efficiency of algorithm, bipartite graph is also embedded.

2.3 Simulated Annealing

Simulated annealing has been proven to be an effective tool to solve an optimization problem. Liao et al. [14] have proposed a framework, called GreenMap, for runtime virtual machine placement. This framework consist of a simulated annealing optimization based algorithm that tries to dynamically map VMs to less number of PMs.

2.4 Constraint Programming

Van et al. [15] have proposed a different framework consisting of two components: dynamic utility based VM provisioning manager and dynamic VM placement manager. Both of these utilizes the concept of constraint programming. Herminier F at el. [24] have proposed a resource manager called Entropy, based on constraint programming. This work handles the problem of VM placement and migration both. Duong-Ba et al. [25] have modelled the VM placement problem as convex optimization problem and proposed a multi-level join VM placement and migration (MJPM) algorithm to solve the resource fragmentation problem.

2.5 Bio-Inspired Techniques

Bio-inspired techniques have always attracted the researchers across to solve NP hard problems. Feller et al. [16] have proposed an Ant Colony Optimization (ACO) based algorithm that binds VMs to PMs based on the current workload. Jeyrani et al. [17, 21] have used a self-adaptive Particle Swarm Optimization (PSO) for VM placement with better power management. Tripathi et al. [26] have modelled VM placement problem as multi-objective optimization problem and applied modified binary particle swarm optimization (BPSO) to solve this. Inspired from our ecosystem, Zheng et al. [27] have proposed a novel VM placement algorithm based on bio-geography based optimization technique to minimize the power consumption and resource wastage in cloud data centers. Abdel-Basset et al. [28] have proposed a new improved Lévy based Whale optimization algorithm for bandwidth efficient

VM placement. Satpathy et al. [29] have proposed an attractive method using Crow search optimization for VM placement problem, modelled as multi-objective optimization problem with resource wastage and power consumption as the objectives.

Few most important works have been summarized in Table 1 for better understanding.

Most of the above discussed models consider power consumption or network bandwidth as the objectives to be optimized. Very few of them have emphasized on resource utilization metric effectively. The proposed work solely concentrates on optimal resource utilization in solving the VM placement problem.

3 The Problem

In this section, VM placement problem is discussed and illustrated with an example. Cloud data center, the backbone of cloud computing, contains large number of physical servers over which large number of VMs are mapped with the help of virtualization technology. The prime objective of virtual machine placement algorithm is to utilize the resources effectively and efficiently. Consider a scenario in which seven VMs need to be placed on physical machines (PMs) of a cloud data center. Each PM has four cores therefore, can host up to four VMs. Assume resource requirement of these VMs are 60%, 35%, 25%, 30%, 35%, 40% and 15% of the total capacity of PMs. Further, assume that PMs are homogeneous and resource utilization threshold is 95%. If each VM is placed on a single PM then 7 PMs are needed. In this scenario, resource utilization will be extremely poor as most of the resources will be underutilized. Since it is possible to place four VMs, keeping the availability of resources in mind, an alternative and optimal way of VM placement is possible as shown in Fig. 1. In this, only three PMs are required to be in active mode with better resource utilization.

For better understanding, the notation used throughout the work is given in Table 2.

For the VM placement problem, in the proposed model, a modified dragonfly algorithm is applied. Therefore, before proceeding to the proposed model, some basics of dragonfly algorithm is discussed in the following section.

3.1 Dragonfly Algorithm

Dragonflies are insects, also known as Odonata. The lifespan of dragonflies are categorized into two milestones: nymph and adult. The time span of Nymph dominates that of adult. The swarming behaviors of dragonflies are very rare but unique. Dragonflies form swarm only for two purposes; to hunt the small insects & fish and to migrate from one place to another. When dragonflies form the swarm in order to hunt, it is called static swarm whereas forming the swarm to migrate is called dynamic or migratory swarm. In static swarm, dragonflies move in small groups to hunt their prey. Also, motion of a dragonfly is back and forth over a small hunting area. Sudden change in their trajectory and local movement makes the static

Table 1 Comparison of some relevant methods of VM placement

Technique	Resources under consideration	Novelty	Strength	Weakness
VM Placement using ACO [20]	CPU, Memory	Optimization of resource wastage and power consumption	Multi-objective modelling	High computationtime
VM Placement using Vector Dot [30]	Bandwidth, Memory, CPU, Network	Integrating server & storage migration	Handling overloaded nodes & dynamic load balancing	Statistical and predictive models followed
VM Placement using Exact Allocation and Migration [31]	CPU	Minimization of number of migrations & energy	Reduction in migration cost	Slow execution
VM Placement using Group Packing Algorithm [32]	Bandwidth, CPU	Future bandwidth Prediction using random variables	Reducing number of PMs	Statistical and predictive models followed
VM Placement using Genetic algorithm [18]	CPU	Forecasting requests and reconfiguration	Increased CPU utilization and reduction in active PMs	Large search space overhead
VM Placement using Enhanced First Fit Decreasing algorithm [33]	CPU	Reuse of VMs	Improved energy efficiency & high system throughput	SLA Violation
Exact algorithms for energy-efficient VM Placement using in data centers [34]	CPU and memory	Address multi objective under multiple-capacity constraints	Better than an on-the-shelf mixed-integer linear program solver	High computation time
Optimistic VM Placement in cloud data center using queuing approach [35]	None	Optimization of completion time, production cost and throughput	High performance gain for cloud applications	Security could be compromised

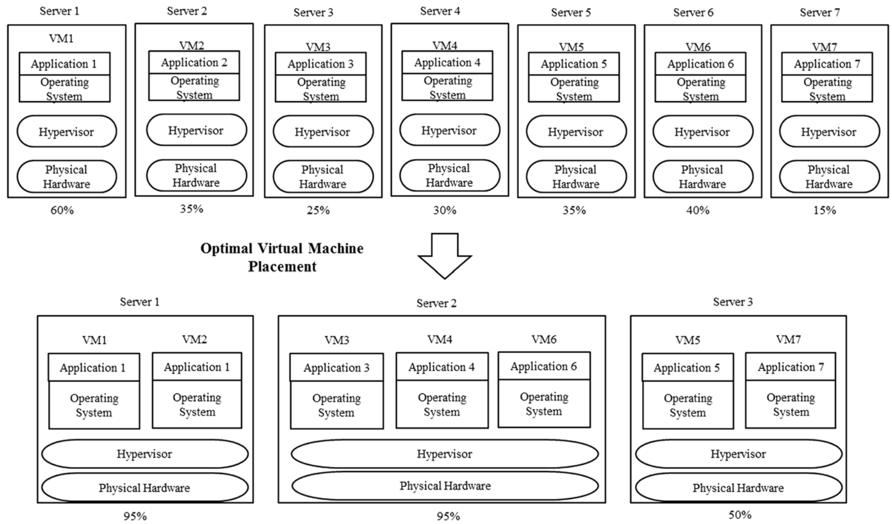

Fig. 1 An example of VM Placement problem [26]

Table 2 Notation

Symbol	Description
S_i	Separation of *i*th dragonfly
X	Position of current individual
N	Total count of neighbors
A_i	Alignment of *i*th dragonfly
V_j	Velocity of the *j*th neighboring individual
C_i	Cohesion of *i*th dragonfly
F_i	Attraction of *i*th dragonfly towards the food source
E_i	Distraction of *i*th dragonfly from the predator
s, a, c, f, e	Weights corresponding to separation, alignment, cohesion, attraction towards the food source and distraction from predators
n	Number of VMs
m	Number of PMs
T_j^c and T_j^m	Normalized remaining CPU and memory resources at *j*th PM
U_j^c and U_j^m	Normalized resource wastage corresponding to CPU and memory resources at *j*th PM
W_j	Total resource wastage at *j*th PM
I	Set of VMs
J	Set of PMs
ρ_{c_i}, ρ_{m_i}	CPU and memory requirement of *i*th VM
R_{c_j}, R_{m_j}	Threshold CPU and memory capacity of *j*th PM

swarming quiet interesting and useful. In dynamic swarm, dragonflies move in a large group over a long distance.

Seyedali Mirjalili [36] proposed an optimization algorithm, based on the swarming behavior of dragonflies and named it Dragonfly Algorithm. The formation of sub-swarms and its back and forth flying, over a different area, makes static swarm very useful for exploitation phase. Similarly in dynamic swarm, flying of large number of dragonflies in one direction favors the exploration phase. Dragonfly algorithm uses three basic properties of swarm movement which are as follows.

- **Separation** All dragonflies, in a swarm, should maintain some distance from the neighborhood dragonfly to avoid the collision. Reynolds [37] has proposed a method to compute the separation of an individual from its neighboring individuals. The Separation S_i of an individual X from its neighbors X_j can be calculated as given in eq. 1.

$$S_i = \sum_{j=1}^{j=N} X - X_j \tag{1}$$

  Where, N is the total count of neighbors.
- **Alignment** All dragonflies should maintain nearly the same velocity as their neighbor. This is called Alignment A_i which can be calculated as given in eq. 2.

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N} \tag{2}$$

  Where, V_j represents the velocity of the j^{th} neighboring individual.
- **Cohesion** The trajectory, of all dragonflies in a swarm, should be towards the center of mass of neighborhood. The Cohesion C_i of an individual X with their N neighbors can be calculated as given in eq. 3.

$$C_i = \frac{\sum_{j=1}^{j=N} X_j}{N} - X \tag{3}$$

In order to survive, all individuals should move towards the food source but simultaneously should stay away from the predators. Attraction towards the food source$X^+$and distraction from the predatorX^-can be measured as given in eqs. 4 and 5 respectively.

$$F_i = X^+ - X \tag{4}$$

$$E_i = X^- + X \tag{5}$$

Dragonfly algorithm utilizes the concept of velocity of PSO algorithm and proposes a new vector called stepΔX. The step vector represents the direction of swarm movement and can be calculated as given in eq. 6.

$$\Delta X_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\Delta X_t \tag{6}$$

Here, *s, a, c, f,* and *e* are the weights corresponding to separation, alignment, cohesion, attraction, and distraction respectively. *w* represents the inertia weight. Position vector, after *t* iteration, will be updated as given in eq. 7.

$$X_{t+1} = X_t + \Delta X_{t+1} \tag{7}$$

For balancing the exploration and exploitation in the optimization process, fine tuning of *s, a, c, f* and *e* is required. In dynamic swarm, dragonflies align themselves and maintain proper separation and cohesion whereas in static swarm the alignment is very low and cohesion is very high in order to attack on a prey (food source). Therefore, for exploiting the search space high alignment and low cohesion weights are assigned. On the other hand, for exploring the search space low alignment and high cohesion weights are assigned.

For stochastic behavior in artificial dragonflies, it is assumed that dragonflies are flying in the search space using random walk also called as Levy flight in case of zero neighborhood. For this, the position of a dragonfly will be updated according to eq. 8.

$$X_{t+1} = X_t + Levy(d) \times X_t \tag{8}$$

Where *d* represents the dimension of the position vector. Levy flight [38] can be calculated as in eq. 9.

$$Levy(x) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{1/\beta}} \tag{9}$$

Where r_1 and r_2 are two randomly generated real numbers in the range [0, 1] andβ is a constant tuned during the experiment.σ is calculated according to eq. 10.

$$\sigma = \left(\frac{\Gamma(1+\beta) \times Sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{1/\beta} \tag{10}$$

Where$\Gamma(x) = (x-1)!$.

Basic dragonfly algorithm, for continuous search space, is presented below.

Algorithm 1: Dragonfly Algorithm

Generate the random population of dragonflies $X_i\ \forall i$
Initialize the velocity or step vector $\Delta X_i\ \forall i$
While *(termination condition is not satisfied)* ***Do***
    Find the fitness values for each dragonflies $X_i\ \forall i$
    Update X^+ ***and*** X^-
    Update the values for w, s, a, c, f and e
    Compute $S_i, A_i, C_i, F_i,$ and E_i *using equations (1), (2), (3), (4) and (5) respectively*
    Update the radius of neighborhood
    If (*neighborhood count is not zero)*
        Update ΔX_{t+1} *using equation (6)*
        Update X_{t+1} *using equation (7)*
    else
        Update X_{t+1} *using equation (8)*
    End If
    Check the validity of new position of dragonflies
End While

4 The Problem Formulation

In cloud data center, VMs' requirement are served by physical machines using an appropriate placement scheme. In this section, VM placement problem is mathematically formulated as a multidimensional bin packing problem.

Let there are n VMs to be mapped over m PMs. If we consider the possibility of a case where a VM may not be assigned to any of PMs, there will total be $(m+1)^n$ ways for assigning n VMs onto m PMs.

4.1 Resource Wastage Modelling

A VM contains various resources such as CPU, memory, storage, network bandwidth etc. In the proposed work, only CPU and memory have been considered as these two are prime VM resources. The total resource wastage at jth PM can be formulated as in eq. 11.

$$W_j = \frac{\left|T_j^c - T_j^m\right| + \varepsilon}{U_j^c + U_j^m} \tag{11}$$

$T_j^c$and$T_j^m$are normalized remaining CPU and remaining memory at j^{th} PM,U_j^c and$U_j^m$are normalized wastage corresponding to CPU and memory.ε, a small real number, is fixed to 0.0001.

4.2 Resource Wastage as an Optimization Problem

VM placement problem can mathematically be modelled as an optimization problem with the objective to minimize resource wastage. Let I represents the set of VMs and J represents the set of PMs, ρ_{c_i} represents the CPU requirement and ρ_{m_i} represents the memory requirement of i^{th} VM, R_{c_j}represents the threshold CPU capacity and R_{m_j} represents the threshold memory capacity. Two Boolean variables x_{ij} and y_j can be defined as follows.

$$x_{ij} = \begin{cases} 1 \text{ } \textit{if the VM } i \in I \textit{ is assigned to physical server } j \in J \\ 0 \text{ } \textit{otherwise} \end{cases}$$

$$y_j = \begin{cases} 1 \text{ } \textit{if the server } j \in J \textit{ is in use} \\ 0 \text{ } \textit{otherwise} \end{cases}$$

The VM placement problem can be viewed as given in eq. 12

$$f = Minimize \sum_{j=1}^{m} W_j = \sum_{j=1}^{m} \left[y_j \times \frac{\left| \left(R_{c_j} - \sum_{i=1}^{n} (x_{ij} \cdot \rho_{c_i}) \right) - \left(R_{m_j} - \sum_{i=1}^{n} (x_{ij} \cdot \rho_{m_i}) \right) \right| + \varepsilon}{\sum_{i=1}^{n} (x_{ij} \cdot \rho_{c_i}) + \sum_{i=1}^{n} (x_{ij} \cdot \rho_{m_i})} \right] \tag{12}$$

Subject to:

$$\sum_{j=1}^{m} x_{ij} = 1 \; \forall i \in I \tag{13}$$

$$\sum_{i=1}^{n} \rho_{c_i} \cdot x_{ij} \le R_{c_j} \cdot y_j \; \forall j \in J \tag{14}$$

$$\sum_{i=1}^{n} \rho_{m_i} \cdot x_{ij} \le R_{m_j} \cdot y_j \; \forall j \in J \tag{15}$$

The objective function in eq. 12 deals with the minimization of resource wastage of all deployed physical machines. eq. 13 represents the condition that each VM can be assigned to only one physical machine. eqs. 14 and 15 represent that requested resource requirement of all VMs mapped to a particular PM should not exceed the threshold value of the corresponding resources. The search space, associated with the formulated optimization problem, is binary in nature. Therefore, the dragonfly algorithm for the binary search space with some modifications is used to solve this problem.

5 The Proposed Model

This section discusses the detail of VM placement using modified Dragonfly Algorithm (VMPDA) and modifications done in original dragonfly algorithm to tune it for VM placement in cloud data center.

5.1 Particle Representation and Generator Function

If n virtual machines are to be mapped over m physical servers, then each particle X is represented by a matrix of order $n \times m$ with each entry x_{ij} as 0 or 1.

$$X = \begin{bmatrix} 0 & 1 & \dots 0 \cdots & 0 \\ 1 & 0 & \dots 0 \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \dots 0 \dots & 0 \end{bmatrix}. \tag{15}$$

The randomly generated initial population may or may not satisfy the constraints indicated in eq. (13). Therefore, following generator algorithm is used.

Algorithm 2: *Generator(n, m)*

```
1. x = zeros(n, m);          // Generates a matrix of n × m
2. j = randi([1 m],1,n);     // Generates a vector of n×1 that has all values randomly between [1,m]
3. For i= 1 to n
        x (i, j (i))=1; // Assigning a particular VM to a physical machine
4. End For
```

Seyedali Mirjalili [36] has also modified the dragonfly algorithm to work in binary search space using the concept of transfer function [39, 40]. Two types of transfer functions; *S-shaped* and *V-shaped* restrict the output value to 0 or 1. The transfer function, used in binary dragonfly algorithm (BDA), is given in eq. 16.

$$T(\Delta x) = \left| \frac{\Delta x}{\sqrt{\Delta x^2 + 1}} \right| \tag{16}$$

Based on the value obtained in eq. 16, new position of dragonflies is calculated as given in eq. 17.

$$X_{t+1} = \begin{cases} \bar{X}_t \text{ } if \text{ } rand < T(\Delta x_{t+1}) \\ X_t \text{ } if \text{ } rand \geq T(\Delta x_{t+1}) \end{cases} \tag{17}$$

Here, *rand* is a random real number in the range [0, 1].

However, eq. 16 does not provide a balanced approach between exploitation and exploration as some better area remains unexplored. Therefore, in the proposed model, a time varying transfer function [41] is used as shown in eq. 18.

$$T(\Delta x, \tau) = \begin{cases} 1 - \dfrac{2}{1 + e^{-2x/\tau}} \; if \; x \leq 0 \\ \dfrac{2}{1 + e^{-2x/\tau}} - 1 \; if \; x > 0 \end{cases} \tag{18}$$

Here, time variable τ starts with some value and is reduced over the time as shown in eq. 19.

$$\tau = \left(1 - \frac{t}{T}\right)\tau_{\max} + \frac{t}{T}\tau_{\min} \tag{19}$$

Here, t is current iteration, T is maximum number of iteration and τ_{max} and τ_{min} are maximum and minimum values of controlling parameter. Based on the above transfer function, new position of a dragonfly can be calculated as in eq. 20.

$$X_{t+1} = \begin{cases} 0 \quad if \; rand \leq T(\Delta x_{t+1}) \; and \; \Delta x_{t+1} \leq 0 \\ 1 \quad if \; rand \leq T(\Delta x_{t+1}) \; and \; \Delta x_{t+1} > 0 \\ X_t \quad if \; rand > T(\Delta x_{t+1}) \end{cases}, \tag{20}$$

In binary search space, the distance between two dragonflies cannot be calculated, therefore, all dragonflies are assumed to be in a single swarm. All the weights corresponding to separation, alignment, cohesion, attraction, distraction and inertia i.e. *s, a, c, f, e* and *w* are adaptively tuned.

Proper problem formulation, with the above-mentioned modification in the dragonfly algorithm, will lead towards an optimal solution of the problem in a binary search space.

5.2 Proposed Dragonfly Algorithm for VM Placement Problem

The virtual machine placement algorithm using dragonfly algorithm in a binary search space is given as Algorithm 3.

Algorithm 3: Modified Dragonfly Algorithm for Virtual Machine Placement (VMPDA)

1. *Generate the random population of dragonflies X_i $\forall i$ using Generator()*
2. *Initialize velocity or step vector ΔX_i $\forall i$, $\boldsymbol{\tau_{max}}$ and $\boldsymbol{\tau_{min}}$*
3. *Initialize $\boldsymbol{\tau}$ using equation (19)*
4. ***While** (termination condition is not reached) **do***
5. *Find the fitness values for each dragonfly X_i $\forall i$ using equation 12*
6. *Update X^+ **and** X^-*
7. *Update the values for w, s, a, c, f and e*
8. *Compute S_i, A_i, C_i, F_i, and E_i using equations (1), (2), (3), (4), and (5) respectively*
9. *Update ΔX_{t+1} using equation (6)*
10. *Compute $T(\Delta x)$ using equation (18)*
11. *if $\left(rand \le T(\Delta x_{t+1}) \text{ and } \Delta x_{t+1} \le 0\right)$*
12. $X_{t+1} = 0$
13. *else if $\left(rand \le T(\Delta x_{t+1}) \text{ and } \Delta x_{t+1} > 0\right)$*
14. $X_{t+1} = 1$
15. *else if $\left(rand > T(\Delta x_{t+1})\right)$*
16. $X_{t+1} = X_t$
17. ***End if***
18. End While

The flowchart of VMPDA algorithm is given in Fig. 2.

6 Simulation Experiments and Analysis

The performance and effectiveness of the proposed VMPDA algorithm is done by comparing it with state of art such as VM placement using ant colony search optimization (VMPACS) [20], VM placement using genetic algorithm (VMPGA) [18], VM placement using artificial bee colony (VMPABC) and VM placement using binary particle swarm optimization (VMPBPSO) [26]. These implementation is done using CloudSim simulation toolkit [42]. The experiments are carried out using synthesized data from similar work [20] and also real data from Amazon EC2 [43].

For the realistic verification and analysis of the proposed model, experiments are conducted for two types of VM sets: small set of VMs which consist of 50 VMs with certain CPU and memory requirements and large set of VMs which consist of 2000 VMs with certain CPU and memory requirement. The physical servers are simulated with the same dimension of resources as the VMs. To cater the worst-case possibility, the number of PMs are kept same as number of VMs. For simplicity, PMs' configuration are considered homogeneous, though heterogeneous environment can also be simulated with the proposed model. For each algorithm, ten Monte Carlo simulations are performed and their average is reported. Keeping in mind the complexity of the simulated algorithms the termination condition is kept in form of

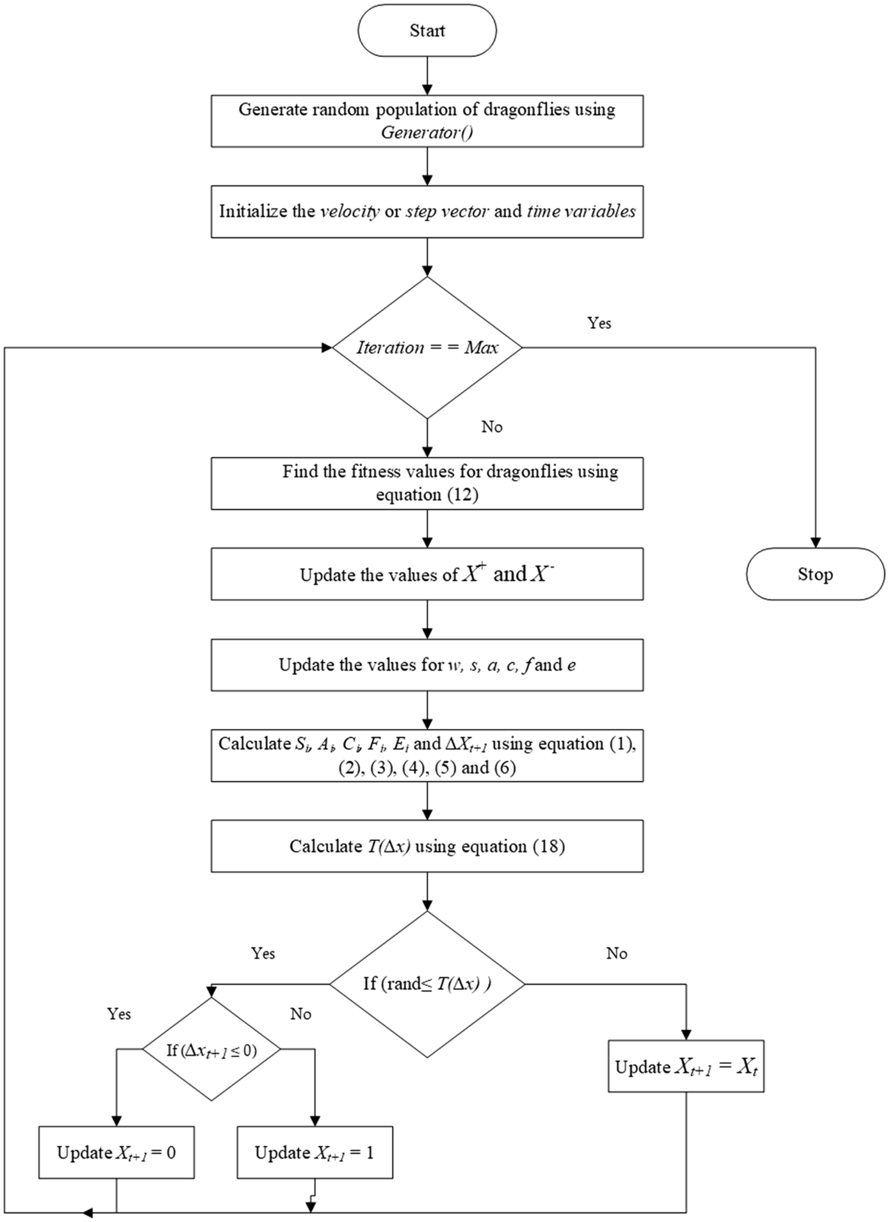

Fig. 2 The VMPDA flowchart

number of function evaluations which is set to 70,000 for both types of data sets so that the algorithm could settle.

6.1 Comparison Based on Resource Wastage Cost

Entire experiment is divided into two test cases based on their reference CPU requirement $\bar{\rho}_c$ and memory requirement $\bar{\rho}_m$ using the *VM_configuration* Algorithm 4.

Algorithm 4: ***VM_configuration***

For *i = 1 to n do*
    $\boldsymbol{\rho}_{\boldsymbol{c}_i}$ = *rand* $(2\bar{\boldsymbol{\rho}}_c)$;
    $\boldsymbol{\rho}_{\boldsymbol{m}_i}$ = *rand* $(\bar{\boldsymbol{\rho}}_m)$;
    $\boldsymbol{r} = \boldsymbol{rand}\ (\mathbf{1.0})$;
    If $(\boldsymbol{r} < \boldsymbol{P} \wedge \boldsymbol{\rho}_{\boldsymbol{c}_i} \geq \bar{\boldsymbol{\rho}}_{\boldsymbol{c}}) \vee (\boldsymbol{r} \geq \boldsymbol{P} \wedge \boldsymbol{\rho}_{\boldsymbol{c}_i} < \bar{\boldsymbol{\rho}}_{\boldsymbol{c}})$
        $\boldsymbol{\rho}_{\boldsymbol{m}_i} = \boldsymbol{\rho}_{\boldsymbol{m}_i} + \bar{\boldsymbol{\rho}}_{\boldsymbol{m}}$;
    End If
End For

Test case 1 comprise of $\bar{\rho}_c = \bar{\rho}_m = 25\%$ and test case 2 comprise of $\bar{\rho}_c = \bar{\rho}_m = 45\%$. For test case 1, distribution values of CPU and memory requirement lies between [0, 50]. For test case 2, distribution values of CPU and memory requirement lies between [0, 90]. Note that the threshold value of physical servers is set to 90%. For both the test cases, probability P is set to 0.00, 0.25, 0.50, 0.75, and 1.0. For test case 1, the obtained correlation coefficients are −0.754, −0.348, −0.072, 0.371 and 0.755 for respective probabilities. These correlation coefficients correspond to strong-negative, weak-negative, no, weak-positive, and strong-positive correlations. Similarly, for test case 2, the obtained correlation coefficients are −0.755, −0.374, −0.052, 0.398 and 0.751.

Resource wastage of VMPDA is compared with VMPACS, VMPGA, VMPABC and VMPBPSO algorithms. In Table 3, resource wastage is computed for these algorithms after 70,000 function evaluations. It has been observed, from the table, that VMPDA algorithm leads to least resource wastage as compared to VMPACS, VMPGA, VMPABC and VMPBPSO for all correlations in both the test cases. VMPDA possess high exploration making it suitable for discovering the promising regions of the binary search space. Abrupt change, in the flying path of dragonflies, makes it suitable in avoiding local optima and forces it to move towards global optima.

Genetic algorithm does not guarantee optimal solution as it largely depends upon the initial population. In VMPGA, poor initial population leads to highest resource wastage as compared to other algorithms. This results in highest resource wastage for VMPGA among other compared algorithms in all the cases. It can also be observed that VMPBPSO results in less resource wastage as compared to other algorithms except VMPDA. VMPBPSO responds to quality and diversity of the solutions as here only $\overrightarrow{gbest}$ shares the information with others. So, overall behavior of

Table 3 Resource wastage comparison of various algorithms

Reference value	Correlation coefficient	Algorithm	Resource wastage for small set of VMs	Resource wastage for large set of VMs
$\bar{\rho}_c = \bar{\rho}_m = 25\%$	−0.754	VMPACS	3.61	2.10
		VMPGA	5.43	2.54
		VMPABC	4.98	2.32
		VMPBPSO	3.12	1.92
		VMPDA	**2.88**	**1.84**
	−0.348	VMPACS	2.86	1.94
		VMPGA	5.12	2.42
		VMPABC	4.81	2.20
		VMPBPSO	2.54	1.71
		VMPDA	**2.49**	**1.63**
	−0.072	VMPACS	2.78	1.79
		VMPGA	4.86	2.21
		VMPABC	4.37	2.07
		VMPBPSO	2.62	1.65
		VMPDA	**2.53**	**1.50**
	0.371	VMPACS	2.47	1.68
		VMPGA	4.70	2.09
		VMPABC	3.98	1.91
		VMPBPSO	1.97	1.55
		VMPDA	**1.82**	**1.39**
	0.755	VMPACS	2.31	1.45
		VMPGA	4.10	1.92
		VMPABC	3.64	1.73
		VMPBPSO	1.89	1.32
		VMPDA	**1.76**	**1.20**

Table 3 (continued)

Reference value	Correlation coefficient	Algorithm	Resource wastage for small set of VMs	Resource wastage for large set of VMs
$\bar{\rho}_c = \bar{\rho}_m = 45\%$	− 0.755	VMPACS	14.72	12.80
		VMPGA	18.21	15.42
		VMPABC	17.24	13.17
		VMPBPSO	13.83	11.43
		VMPDA	**12.90**	**10.02**
	− 0.374	VMPACS	13.66	11.74
		VMPGA	17.19	14.42
		VMPABC	16.04	12.00
		VMPBPSO	11.97	10.13
		VMPDA	**10.32**	**9.20**
	− 0.052	VMPACS	12.85	10.84
		VMPGA	16.48	13.11
		VMPABC	15.44	11.50
		VMPBPSO	11.91	9.15
		VMPDA	**11.05**	**8.60**
	0.398	VMPACS	11.35	9.87
		VMPGA	15.92	12.76
		VMPABC	14.83	11.93
		VMPBPSO	10.66	8.10
		VMPDA	**9.91**	**7.54**
	0.751	VMPACS	10.32	8.64
		VMPGA	15.01	11.84
		VMPABC	14.22	10.78
		VMPBPSO	9.74	7.56
		VMPDA	**8.99**	**6.12**

Bold values indicate the findings of proposed algorithm VMPDA

the above mentioned algorithms for resource wastage order is summarized as VMP GA > VMPABC > VMPACS > VMPBPSO > VMPDA.

In order to show the convergence behavior of all five algorithms, mean cost curve is drawn as shown in right half of Figs. 3, 4, 5, 6, 7, 8, 9, 10, 11, 12. Mean value of resource wastage cost, for each function evaluation, is represented with the curves. Statistical distribution of these costs are shown with boxplot. Middle value of costs is represented by median in the boxplots which is evaluated from 70,000 function

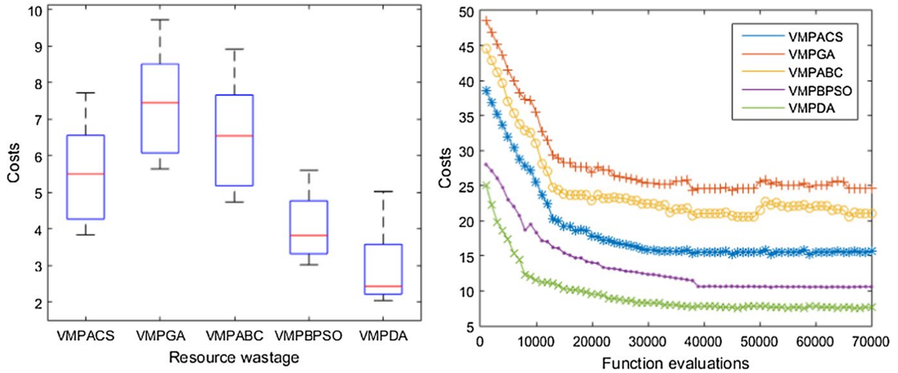

Fig. 3 Boxplot and mean cost curve for resource wastage in $\bar{\rho}_c$=$\bar{\rho}_m$= 25% Corr. = − 0.754

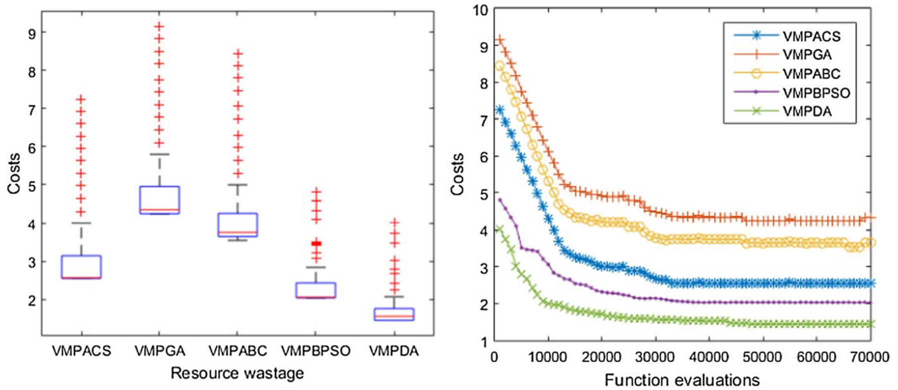

Fig. 4 Boxplot and mean cost curve for resource wastage in $\bar{\rho}_c$=$\bar{\rho}_m$= 25% Corr. = − 0.348

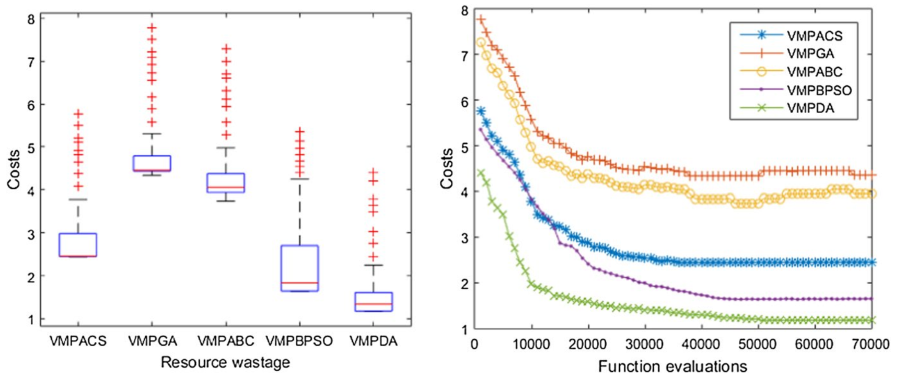

Fig. 5 Boxplot and mean cost curve for resource wastage in $\bar{\rho}_c$=$\bar{\rho}_m$= 25% Corr. = − 0.072

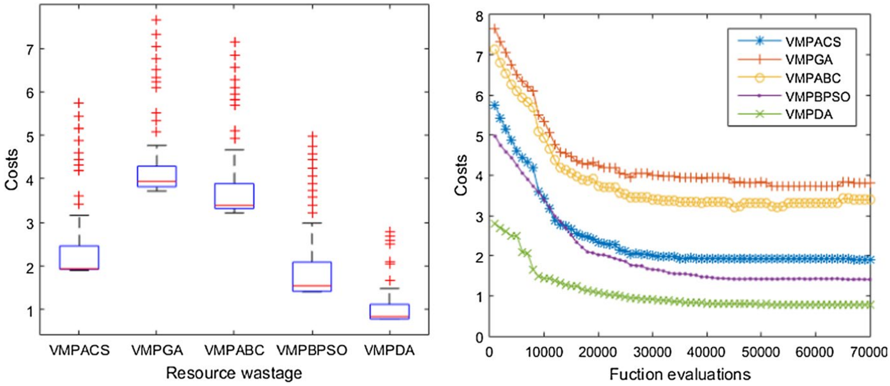

Fig. 6 Boxplot and mean cost curve for resource wastage in $\bar{\rho}_c$=$\bar{\rho}_m$= 25% Corr. = 0.371

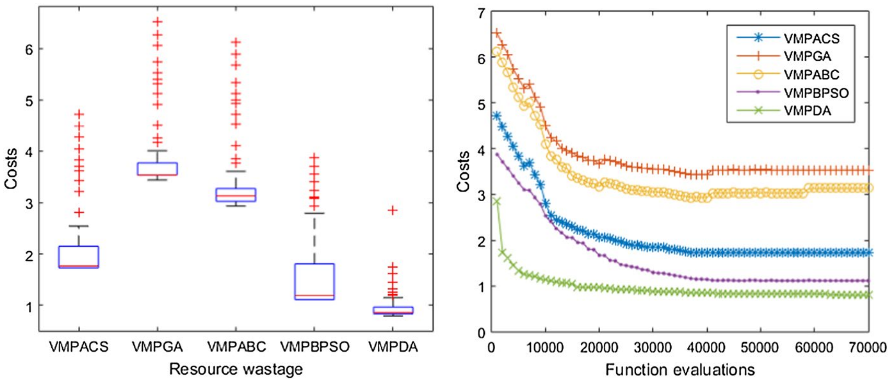

Fig. 7 Boxplot and mean cost curve for resource wastage in $\bar{\rho}_c$=$\bar{\rho}_m$= 25% Corr. = 0.755

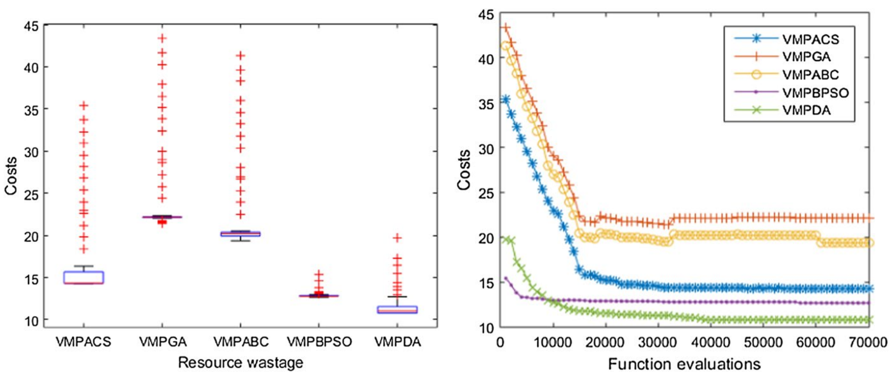

Fig. 8 Boxplot and mean cost curve for resource wastage in $\bar{\rho}_c$=$\bar{\rho}_m$= 45% Corr. = − 0.755

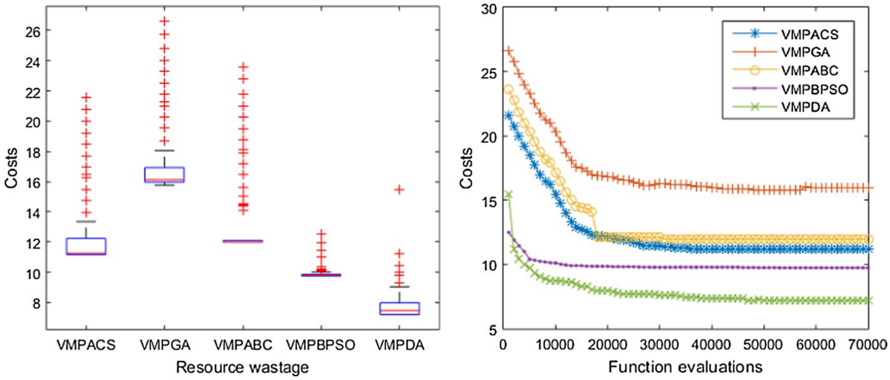

Fig. 9 Boxplot and mean cost curve for resource wastage in $\bar{\rho}_c$=$\bar{\rho}_m$= 45% Corr. = − 0.374

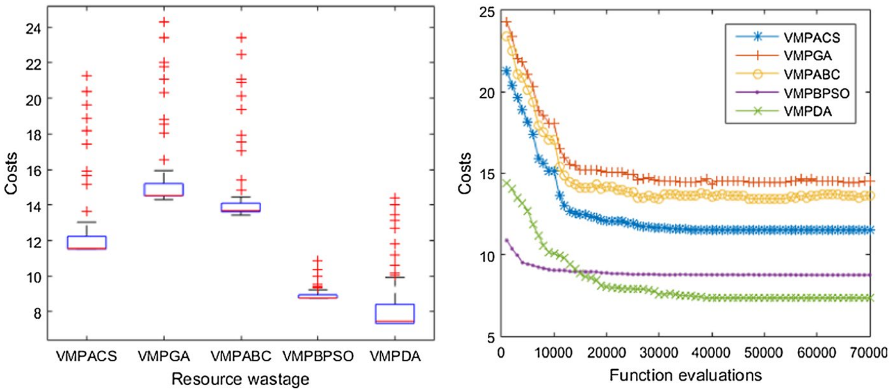

Fig. 10 Boxplot and mean cost curve for resource wastage in $\bar{\rho}_c$=$\bar{\rho}_m$= 45% Corr. = − 0.052

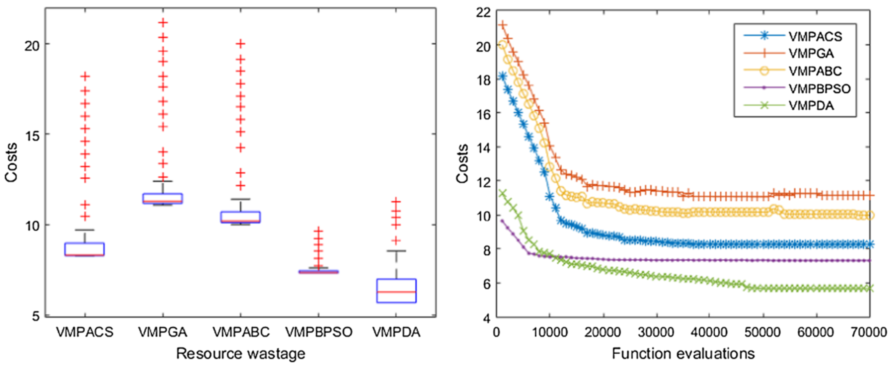

Fig. 11 Boxplot and mean cost curve for resource wastage in $\bar{\rho}_c$=$\bar{\rho}_m$= 45% Corr. = 0.398

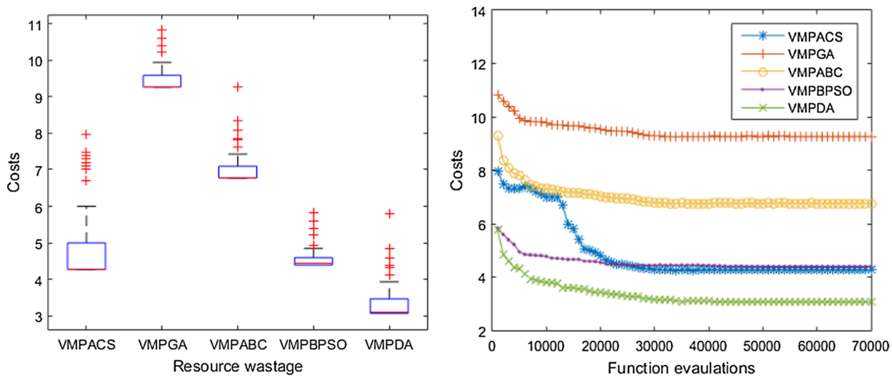

Fig. 12 Boxplot and mean cost curve for resource wastage in $\bar{\rho}_c$=$\bar{\rho}_m$= 45% Corr. = 0.751

evaluations for synthesized data. All figures, on mean cost curve, indicate that VMPDA converges faster than VMPACS, VMPGA, VMPABC and VMPBPSO. The order of resource wastage at different function evaluations is VMPDA < VMPBPSO < VMPACS < VMPABC < VMPGA. VMPDA, at successive function evaluations, provide better solutions with better resource utilization. It is because VMPDA exhibits high exploration and exploitation as it incorporates V-shaped transfer function. The problem of local optima in VMPGA leads to mapping of the VMs onto more number of PMs. Therefore, VMPGA performs worst in all the cases. For higher reference value i.e. $\bar{\rho}_c$=$\bar{\rho}_m$= 45%, it is also observed that at lower function evaluations VMPBPSO exhibits lower resource wastage as compared to other algorithms though this behavior is visible for very short duration. After few more function evaluations, VMPDA outperforms VMPBPSO.

Box plots, of all five algorithms for both test cases, are also shown at the left half of Figs. 3, 4, 5, 6, 7, 8, 9, 10, 11, 12. Median values of all the boxes can be relatively analyzed for the performance study of the VM placement algorithms into consideration. By analyzing the boxplots, it can be inferred that most of the mean cost values of VMPDA are comparatively smaller than minimum values of VMPGA, VMPABC, VMPACS and VMPBPSO. Figures 8, 9, 10, 11 for test case 2, infer that initially VMPBPSO performs better than rest of the algorithms but after some function evaluation VMPDA outperforms others. This behavior is shown for all correlation coefficients. The reason for this behavior being VMPBPSO more prone to local convergence but after sufficient function evaluation, it shows the actual cost. VMPDA facilitates tracking of position of dragonflies during the optimization process. Therefore, it is possible to monitor the values of parameters and tune it to balance the exploitation and exploration of the search space. This property results in lower resource wastage while placing the VMs.

6.2 Comparison Based on Placement Time

Placement time of VMPDA is compared with VMPACS, VMPGA, VMPABC and VMPBPSO as represented in Fig. 13. It reflects the faster convergence of VMPDA

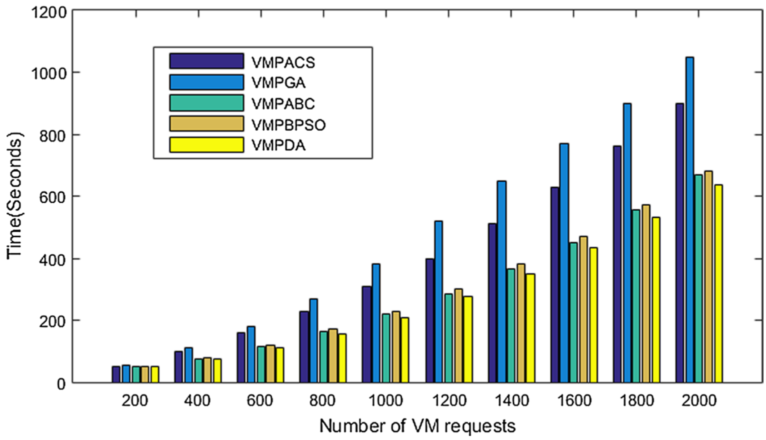

Fig. 13 Placement time on number of VMs

than all other algorithms i.e. time taken to place the VMs in VMPDA is smaller than other four. Use of *V*-shaped transfer function do not limit dragonflies to take values 0 or 1 and therefore it provides an efficient exploitation of search space leading to quick convergence. Despite the fact that VMPGA is very simple, it highly depends upon the initial population. Poor initial population leads to longer run time to find the optimal solution.

6.3 Comparison of VMPDA with Original Binary Dragonfly Algorithm

The performance of the proposed VMPDA algorithm is compared with VM placement algorithm based on original binary dragonfly algorithm VMPODA. This

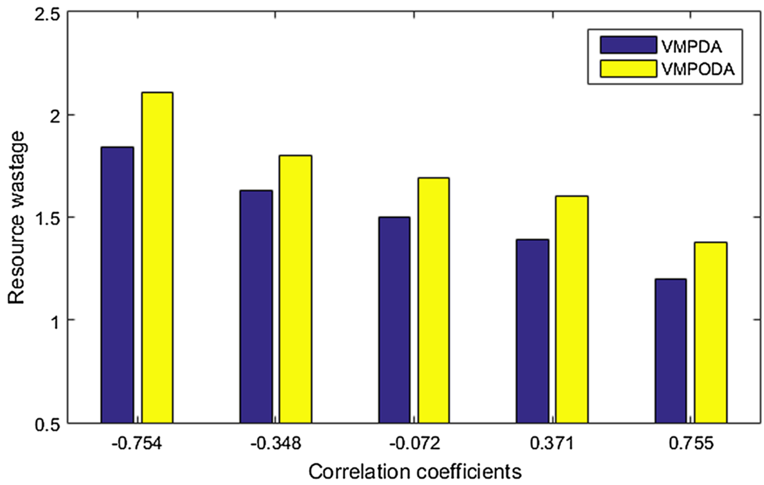

Fig. 14 Comparison of VMPDA and VMPODA for resource wastage

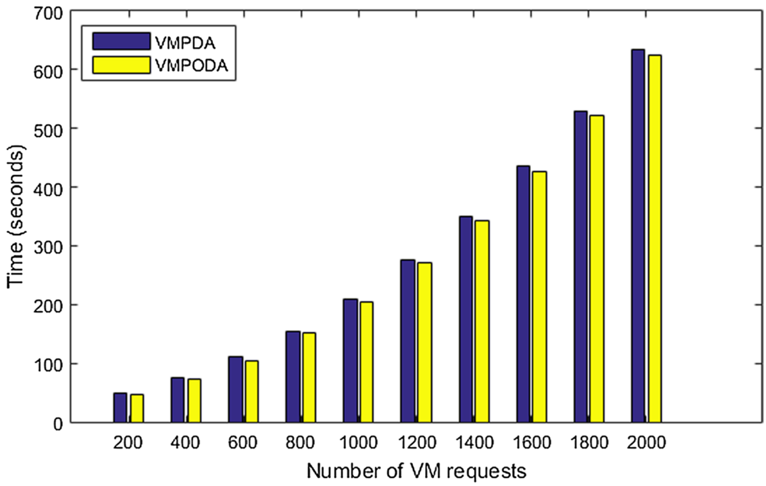

Fig. 15 Comparison of VMPDA and VMPODA for placement time

comparison is performed for the reference value of 25% and for large set of VM placement problem. Figure 14 shows resource wastage of these two algorithms at different correlation coefficient which shows that VMPDA performs better as compared to original binary dragonfly algorithm. Since the step vector is linked with the momentum of the individual dragonfly, it should be very small when dragonfly is close to the best solution. Transfer function affects the step vector hence is also responsible for the position update. VMPODA uses eq. 16 as transfer function and eq. 17 for position update. At the beginning, exploration rate should be more than exploitation rate which is not so in case of eq. 16. Therefore, some promising areas may not have been explored efficiently. Further, VMPDA with transfer function in eq. 18 and position updating rule in eq. 20 provides better solution. Figure 15 shows that time taken to place VMs in VMPDA is slightly more than VMPODA.

7 Conclusion

The resource utilization is very important in a cloud data center for which proper VM placement is a key activity. This work proposes an effective VM placement technique based on dragonfly algorithm to minimize the resource wastage. Few modifications, such as use of time varied V-shaped transfer function, solution generator function etc. are incorporated in simple Dragonfly algorithm to suit it well for the VM placement problem. The proposed model is simulated and extensive experimentation is done on real as well as synthetic data sets. The performance of the proposed VMPDA algorithm is also compared with state of the art e.g. VMPACS, VMPGA, VMPABC and VMPBPSO.

It has been concluded, through simulation experiments, that the proposed model using modified Dragonfly algorithm performs verywell for VM placement problem. In most of the cases, the proposed VMPDA outperforms mainly due to its greater coordination and its capacity to maintain a good balance between exploration and exploitation. VMPDA also results in quick convergence and offers least placement time as compared to others. We foresee the applicability of the VMPDA algorithm in a real cloud environment. The proposed model will be extended to incorporate the dynamic behavior of the cloud environment where, with time, new VM requests will be generated and after some time few VMs may get terminated.

Authors would like to acknowledge the editors and the anonymous reviewers for their useful suggestions resulting in the quality improvement of the paper. Also to acknowledge, MGCU Bihar, IIIT Kota and JNU for their support.

References

1. Zhang, Q., Cheng, L., Boutaba, R.: Cloud Coimputing: state-of-the-art and research challenges. In: J Internet Serv, pp. 626–631. Springer Verlag, IEEE, (2010)
2. Rhoton, J.: Cloud computing explained: implementation handbook for enterprises (2009)
3. Addya, S.K., Turuk, A.K., Sahoo, B., Sarkar, M., Biswash, S.K.: Simulated annealing based VM placement strategy to maximize the profit for Cloud Service Providers. Eng. Sci. Technol. Int. J. **20**, 1249–1259 (2017). https://doi.org/10.1016/j.jestch.2017.09.003
4. Békési, J., Galambos, G., Kellerer, H.: A 5/4 linear time bin packing algorithm. J. Comput. Syst. Sci. **60**, 145–160 (2000). https://doi.org/10.1006/jcss.1999.1667
5. Grit, L., Irwin, D., Yumerefendi, A., Chase, J.: Virtual machine hosting for networked clusters: Building the foundations for "autonomic" orchestration. In: VTDC 2006 2nd International Workshop on virtualization technology in distributed computing; held in conjunction with SC06. IEEE Computer Society, pp. 1–7 (2006)
6. Speitkamp, B., Bichler, M.: A mathematical programming approach for server consolidation problems in virtualized data centers. IEEE Trans. Serv. Comput. **3**, 266–278 (2010). https://doi.org/10.1109/TSC.2010.25
7. Cardosa, M., Korupolu, MR., Singh. A .: Shares and utilities based power consolidation in virtualized server environments. In: 2009 IFIP/IEEE International Symposium on integrated network management, IM 2009, pp. 327–334. IEEE, New York (2009)
8. Bichler, M., Setzer, T., Speitkamp, B.: Capacity planning for virtualized servers. In: Workshop on information technologies and systems, Milwaukee, Wisconsin. Milwaukee, Wisconsin, USA (2006)
9. Srikantaiah, S., Kansal, A., Zhao, F,: Energy aware consolidation for cloud computing. In: Proceedings of the 2008 Conference on power aware computing and systems (HotPower) (2008)
10. Verma, A., Ahuja, P., Neogi, A.: pMapper: Power and migration cost aware application placement in virtualized systems. Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), pp. 243–264. Springer-Verlag, New York Inc (2008)
11. Li, B., Li, J., Huai, J., Wo, T., Li, Q., Zhong, L.: EnaCloud: an energy-saving application live placement approach for cloud computing environments. In: CLOUD 2009–2009 IEEE International Conference on cloud computing, pp. 2009. IEEE, New York (2009)
12. Verma, A., Ahuja, P. Neogi, A .: Power-aware dynamic placement of HPC applications. In: Proceedings of the 22nd annual international conference on ACM, pp. 175–184 (2008)
13. Lin, J.W., Chen, C.H., Lin, C.Y.: Integrating QoS awareness with virtualization in cloud computing systems for delay-sensitive applications. Futur. Gener. Comput. Syst **37**, 478–487 (2014). https://doi.org/10.1016/j.future.2013.12.034
14. Liao, X., Jin, H., Liu, H.: Towards a green cluster through dynamic remapping of virtual machines. Futur. Gener. Comput. Syst **28**, 469–477 (2012). https://doi.org/10.1016/j.future.2011.04.013
15. Van, H.N., Tran, F.D., Menaud, J.M.: Performance and power management for cloud infrastructures. In: Proceedings 2010 IEEE 3rd International Conference on cloud computing, CLOUD 2010, pp. 329–336. IEEE, New York. (2010)

16. Feller, E., Rilling, L., Morin, C.: Energy-aware ant colony based workload placement in clouds. In: Proceedings 2011 12th IEEE/ACM International Conference on grid computing, Grid 2011. IEEE Computer Society, pp. 26–33 (2011)
17. Jeyarani, R., Nagaveni, N., Ram, R.V.: Self adaptive particle swarm optimization for efficient virtual machine provisioning in Cloud. In: International Journal of intelligent information technologies, pp. 88–107. IGI Global, Pennsylvania (2011)
18. Mi, H., Wang, H., Yin, G., Zhou, Y., Shi, D., Yuan, L.: Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers.In:Proceedings 2010 IEEE 7th International Conference on services computing, SCC 2010, pp.514–521. IEEE, New York (2010)
19. Xu, J., Fortes, J.A.B.: Multi-objective virtual machine placement in virtualized data center environments. In: Proceedings 2010 IEEE/ACM International Conference on green computing and communications, GreenCom 2010, 2010 IEEE/ACM International Conference on cyber, physical and social computing, CPSCom 2010, pp. 179–188. IEEE, New York (2010)
20. Gao, Y., Guan, H., Qi, Z., Hou, Y., Liu, L.: A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J. Comput. Syst. Sci. **79**, 1230–1242 (2013). https://doi.org/10.1016/j.jcss.2013.02.004
21. Jeyarani, R., Nagaveni, N., Vasanth Ram, R.: Design and implementation of adaptive power-aware virtual machine provisioner (APA-VMP) using swarm intelligence. Futur. Gener. Comput. Syst **28**, 811–821 (2012). https://doi.org/10.1016/j.future.2011.06.002
22. Wood, T., Shenoy, P., Venkataramani, A., Yousif, M.: Black-box and Gray-box strategies for virtual machine migration. In: 4th USENIX Symposium on networked systems design and implementation, pp. 229–242 (2007)
23. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. Futur. Gener. Comput. Syst **28**, 755–768 (2012). https://doi.org/10.1016/j.future.2011.04.017
24. Hermenier, F., Lorca, X., Menaud, J.M., Muller, G., Lawall, J.: Entropy: a Consolidation Manager for Clusters. In: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on virtual execution environments VEE'09. ACM, pp 41–50 (2009)
25. Duong-Ba, T.H., Nguyen, T., Bose, B., Tran, T.T.: A dynamic virtual machine placement and migration scheme for data centers. IEEE Trans. Serv, Comput (2018)
26. Tripathi, A., Pathak, I., Vidyarthi, D.P.: Energy efficient VM placement for effective resource utilization using modified binary PSO. Comput. J. **61**, 832–846 (2018). https://doi.org/10.1093/comjnl/bxx096
27. Zheng, Q., Li, R., Li, X., Shah, N., Zhang, J., Tian, F., Chao, K.M., Li, J.: Virtual machine consolidated placement based on multi-objective biogeography-based optimization. Futur. Gener. Comput. Syst. **54**, 95–122 (2016). https://doi.org/10.1016/j.future.2015.02.010
28. Abdel-Basset, M., Abdle-Fatah, L., Sangaiah, A.K.: An improved Lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment. Cluster Comput (2018). https://doi.org/10.1007/s10586-018-1769-z
29. Satpathy, A., Addya, S.K., Turuk, A.K., Majhi, B., Sahoo, G.: Crow search based virtual machine placement strategy in cloud data centers with live migration. Comput. Electr. Eng. **69**, 334–350 (2018).https://doi.org/10.1016/j.compeleceng.2017.12.032
30. Singh, A., Korupolu, M., Mohapatra, D.:Server-storage virtualization: Integration and load balancing in data centers. In: 2008 SC International Conference for high performance computing, networking, storage and analysis, SC 2008, pp. 1–12. IEEE, New York (2008)
31. Ghribi, C., Hadji, M., Zeghlache, D.: Energy efficient VM scheduling for cloud data centers: Exact allocation and migration algorithms. In: Proceedings 13th IEEE/ACM International Symposium on cluster, cloud, and grid computing, CCGrid 2013. pp. 671–678. IEEE, New York (2013)
32. Wang, M., Meng, X., Zhang, L.: Consolidating virtual machines with dynamic bandwidth demand in data centers. In: Proceedings IEEE INFOCOM, pp. 71–75. IEEE, New York (2011)
33. Alahmadi, A., Alnowiser, A., Zhu, M.M., Che, D., Ghodous, P.: Enhanced first-fit decreasing algorithm for energy-aware job scheduling in cloud. In: Proceedings 2014 International Conference on computational science and computational intelligence, CSCI 2014, pp. 69–74 (2014)
34. Chen, W., Hu, Z.-H., You-Gan, W.: Exact algorithms for energy-efficient virtual machine placement in data centers. Futur. Gener. Comput. Syst **106**, 77–91 (2020). https://doi.org/10.1016/j.future.2019.12.043

35. Ponraj, A.: Optimistic virtual machine placement in cloud data centers using queuing approach. Futur. Gener. Comput. Syst. **93**, 338–344 (2019). https://doi.org/10.1016/j.future.2018.10.022
36. Mirjalili, S.: Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput. Appl. **27**, 1053–1073 (2016). https://doi.org/10.1007/s00521-015-1920-1
37. Reynolds, C.W.: Flocks, herds, and schools: a distributed behavioral model, in computer graphics. ACM SIGGRAPH Comput. Graph **21**, 25–34 (1987)
38. Yang, X.-S.: Nature-inspired metaheuristic algorithms (2010)
39. Mirjalili, S., Lewis, A.: S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization. Swarm. Evol. Comput. **9**, 1–14 (2013). https://doi.org/10.1016/j.swevo.2012.09.002
40. Mirjalili, S., Wang, G.G., dos Coelho L, S.: Binary optimization using hybrid particle swarm optimization and gravitational search algorithm. Neural Comput. Appl. **25**, 1423–1435 (2014). https://doi.org/10.1007/s00521-014-1629-6
41. Mafarja, M., Aljarah, I., Heidari, A.A., Faris, H., Fournier-Viger, P., Li, X., Mirjalili, S.: Binary dragonfly optimization for feature selection using time-varying transfer functions. Knowledge Based Syst **161**, 185–204 (2018). https://doi.org/10.1016/j.knosys.2018.08.003
42. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. **41**, 23–50 (2011). https://doi.org/10.1002/spe.995
43. Amazon.: EC2 Instance types–Amazon Web Services (AWS). Amazon, Seattle (2019). http://aws.amazon.com/ec2/instance-types

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutionalaffiliations.

Atul Tripathi is an Assistant Professor in Department of Computer Science and Information Technology at Mahatma Gandhi Central University, Bihar, India. He had submitted his Ph.D. at the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India. His research interests include cloud computing, network virtualization, soft computing etc.

Isha Pathak is an Assistant Professor in Department of Computer Science and Engineering, Indian Institute of Information Technology, Kota, India. She did her Ph.D. in Computer Science & Tecnology from Jawaharlal Nehru University, New Delhi, India. She has published research papers in various peer reviewed International Journals (Springer, Wiley, Oxford University Press etc.) and in proceedings of various peer-reviewed conferences. Her research interests include network virtualization, computer networks, resource provisioning in cloud computing.

Deo Prakash Vidyarthi is Professor in the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi. Dr. Vidyarthi has published more than 100 research papers in various peer reviewed International Journals and Transactions (including IEEE, Elsevier, Springer, Wiley, World Scientific etc.) and around 50 research papers in proceedings of various peer-reviewed conferences in India and abroad. Dr. Vidyarthi has three books (research monograph) to his credit. One entitled "Auction Based Resource Provisioning in Cloud Computing" published by Springer in 2018 and another entitled ''Technologies and Protocols for the Future Internet Design: Reinventing the Web'' published by IGI-Global (USA) released in Feb. 2012. The third book is entitled ''Scheduling in Distributed Computing Systems: Design, Analysis and Models'' published by Springer in 2009. He has contributed chapters in many edited books and volumes. He is in the editorial board and in the reviewer's panel of many International Journals. Dr. Vidyarthi is the senior member of IEEE, International Association of Computer Science and Information Technology (IACSIT), Singapore, International Society of Research in Science and Technology (ISRST), USA and International Association of Engineers (IAENG). Research interest includes Parallel and Distributed System, Grid and Cloud Computing, Mobile Computing, Internet of Things and Evolutionary Computing.