PeerJ Computer Science

A multi-objective algorithm for virtual machine placement in cloud environments using a hybrid of particle swarm optimization and flower pollination optimization

Sara Mejahed and M Elshrkawey

Information System Department Faculty of Computers and Information, Suez Canal University, Suez Canal University, Ismailia, Egypt

ABSTRACT

The demand for virtual machine requests has increased recently due to the growing number of users and applications. Therefore, virtual machine placement (VMP) is now critical for the provision of efficient resource management in cloud data centers. The VMP process considers the placement of a set of virtual machines onto a set of physical machines, in accordance with a set of criteria. The optimal solution for multi-objective VMP can be determined by using a fitness function that combines the objectives. This paper proposes a novel model to enhance the performance of the VMP decision-making process. Placement decisions are made based on a fitness function that combines three criteria: placement time, power consumption, and resource wastage. The proposed model aims to satisfy minimum values for the three objectives for placement onto all available physical machines. To optimize the VMP solution, the proposed fitness function was implemented using three optimization algorithms: particle swarm optimization with Lévy flight (PSOLF), flower pollination optimization (FPO), and a proposed hybrid algorithm (HPSOLF-FPO). Each algorithm was tested experimentally. The results of the comparative study between the three algorithms show that the hybrid algorithm has the strongest performance. Moreover, the proposed algorithm was tested against the bin packing best fit strategy. The results show that the proposed algorithm outperforms the best fit strategy in total server utilization.

Subjects Algorithms and Analysis of Algorithms, Autonomous Systems, Computer Networks and Communications, Distributed and Parallel Computing, Optimization Theory and Computation
Keywords Cloud computing, Virtual machine placement, Multi-objectives, Particle swarm optimization, Flower pollination optimization

Submitted 16 June 2021
Accepted 6 December 2021
Published 12 January 2022

Corresponding author
Sara Mejahed,
sara.mongy@ci.suez.edu.eg

Academic editor
Yilun Shang

Additional Information and Declarations can be found on page 22

DOI **10.7717/peerj-cs.834**

INTRODUCTION

Virtualization is one of the most significant technologies in cloud computing systems. It allows the distribution of required resources to multiple users *via* multiple virtual machines (VMs) (*Singh, 2018*). Cloud computing infrastructure incorporates a large number of data centers (DCs) that can communicate over the internet. Each DC holds a vast number of VMs that are hosted on different physical machines (PMs). Hence, each user request is examined and modeled to determine the VM resources required to host and execute the

requested task (*Rashid & Chaturvedi, 2019*). The allocation of VMs onto PMs is known as virtual machine placement (VMP) (*Alashaikh & Alanazi, 2019*). Cloud service providers attempt to exploit VMP methods to maximize the number of VMs placed within each PM, therefore maximizing the number of VMs that a DC can host. As such, VMP is an important topic in the research area of cloud computing (*Gupta & Pateriya, 2014*). Several factors make VMP a complex problem. These include scalability with respect to users requests, PM heterogeneity, and resource multi-dimensionality (*Ghobaei-Arani, Shamsi & Rahmanian, 2017*). Hence, VMP methodology should account for the specific requirements of a given DC. The efficacy of a VMP methodology can be measured by power consumption, cost, resource utilization, and load balancing (*Masdari, Nabavi & Ahmadi, 2016*). Varied research has been undertaken on metaheuristic algorithms (*Alboaneen, Tianfield & Zhang, 2016*; *Donyagard Vahed, Ghobaei-Arani & Souri, 2019*) including particle swarm optimization (PSO), genetic algorithms (GA), ant colony optimization(ACO), and the flower pollination algorithm (FPA). These algorithms were employed to find the optimal PM among all possible PMs in a DC. Optimal selection may be undertaken according to the aforementioned objectives. Metaheuristic algorithms are more advanced than exact strategies, which require high computational costs (*Kumaraswamy & Nair, 2019*). Consequently, metaheuristic algorithms are the superior choice for VMP. This is because the placement problem can be treated as an optimization problem that evaluates the best solution for placement among all candidate solutions.

Generally, the optimization decision of each metaheuristic algorithm is made based on the proposed fitness function. Each fitness function is formulated from parameters that refer to the aforementioned objectives. Hence, each algorithm formulates the optimal solution based on the combined values of the parameters which form the appropriate fitness function, corresponding to the overall objective.

The fitness functions used in existing research consider objectives such as reduction in power consumption, maximization of resource utilization, cost minimization, and load balancing (*Adamuthe, Pandharpatte & Thampi, 2013*). They do not consider placement time: the time consumed by placing a VM onto a cloud server. Specifically, placement time is the difference in time between a VM being requested, and the VM being placed onto a PM. This is an important metric for cloud service providers and potential users. An increase in placement time can violate the service level agreement (*Addya et al., 2015*). Such an infringement can upset users due to the violation of their technical requirements. Hence, placement time should be considered to avoid any deficiencies caused by placement time exclusion.

This paper proposes a novel model to address the problem of VMP by generating an optimal solution through three objectives. These objectives are minimization of the total time required to place each VM onto an appropriate PM, reduction in power consumption of the PMs in the DC, and the minimization of wasted resources. Hence, a novel multi-objective fitness function is proposed that incorporates three parameters that represent these objectives. According to the importance of the three objectives, the intended fitness function is the sum of three parameters of equal weight.

The proposed fitness function is implemented using three algorithms: particle swarm optimization with Lévy flight (PSOLF), flower pollination optimization (FPO), and a proposed hybrid algorithm (HPSOLF-FPO). The particle swarm optimization (PSO) algorithm has the capability to find the optimal solution by searching the local neighborhood for the current best solutions. However, the algorithm can become trapped in local optima even through the evaluation of successive iterations. In addition, particle velocities decrease rapidly, and particle positions converge and search within the same limited area to find the optimal solution from the candidate solutions (*Schmitt & Wanka, 2015*). Thus, the deduced optimal solution may be imprecise, as further possible solutions that may include the precise optimal solution are excluded. The addition of Lévy flight to update particle velocities is not sufficient for the algorithm to avoid becoming trapped in local optima, as the particle velocities decrease after several iterations and the problem reoccurs (*Qingxi & Xiaobo, 2016*). In addition, experiments show that the merging of Lévy flight with PSO causes a large number of VMs to be allocated onto a single PM. Accordingly, this allocation leads to substantial congestion on the selected PM.

The FPO algorithm is a modern optimization method derived from the pollination behavior of flowers, and uses both local and global search. It explores the local neighborhood of each of the best solutions such that the search spaces are guaranteed to be explored more efficiently (*Nabil, 2016*). However, due to the nature of the random search mechanism, over multiple iterations FPO struggles to find a compromise between global and local search. This deficiency reduces the exploitation capability of the algorithm and as such it tends towards fast convergence which limits its ability to find the optimal solution. Hence, the independent implementation of FPO is inefficient (*Hoang, Bui & Liao, 2016*).

This paper proposes the HPSOLF-FPO algorithm to overcome the deficiencies of the independent implementations of the PSO and FPO algorithms. The HPSOLF-FPO algorithm is designed to combine the exploration capabilities of FPO with the exploitation capabilities of PSO. Using the exploration capabilities of FPO, HPSOLF-FPO can move out of any local optima. Using the exploitation capabilities of PSO, the optimal solution can be obtained by intensifying the search around the local neighborhoods of the current best solutions. The hybrid algorithm produces superior results to the separate implementations of PSO or FPO. Furthermore, this combination gives rise to a search mechanism and accuracy which improves the placement decisions for the VMs.

The paper is organized as follows. 'Related Work' discusses the related work of VMP in a cloud computing environment. 'VMP System Model and Problem Formulation' details the mathematical formulation and system model of the optimization problem. 'The Implementation of the Proposed Fitness Function Using PSO, FPO and HPSOLF-FPO' describes the proposed VMP optimization algorithms: PSO, FPO, and HPSOLF-FPO. 'Simulation Evaluation' demonstrates and discusses the simulations and experimental results. Finally, 'Conclusion and Future Work' summarizes and concludes the paper and discusses possible future works.

RELATED WORK

The placement of VMs onto appropriate PMs is a critical topic in cloud computing. Diverse algorithms have been proposed to resolve the issues and challenges facing this problem. Each algorithm attempts to distribute VMs onto PMs according to specific objectives. These objectives represent reduced power consumption, increased resource utilization, load balancing, cost minimization, and reduced service level agreement penetration (*Gahlawat & Sharma, 2014*; *Pires & Barán, 2015*; *Zhao, Zhou & Li, 2019*; *Silva Filho et al., 2018*). Hence, according to the implemented objectives, VMP algorithms may be categorized into two principal approaches: single-objective algorithms and multi-objective algorithms. Single-objective algorithms are deployed using just one of the aforementioned objectives. *Saeedi & Shirvani (2021)* propose a resource skewness-aware VM consolidation based on the improved thermodynamic simulated annealing algorithm, and a system framework with different modules that allow VMP to be modeled as an integer linear programming problem. The algorithm outperforms two heuristics and two metaheuristics in minimization of the number of used servers and reduction in data center resource wastage. Multi-objective algorithms are implemented by combining a selection of the aforementioned objectives into a single fitness function. *Farzai, Shirvani & Rabbani (2020)* propose a hybrid multi-objective genetics-based optimization solution for VMP by considering three objectives: reduced power consumption, reduced resource wastage, and reduced bandwidth usage in consideration of the data center topology for co-hosting dependent VMs.

Generally, VMP algorithms may be classified into four principal classes (*Attaoui & Sabir, 2018*). These classes are heuristic, metaheuristic, exact, and approximate. Exact algorithms include constraint programming (*Van, Tran & Menaud, 2010*; *Mann, 2016*), integer linear programming (*López, Kushik & Zeghlache, 2019*), mixed integer linear programming (*Regaieg et al., 2018*), and pseudo-boolean optimization (*Ribas et al., 2013*). Although these algorithms generate optimal solutions, they suffer from exponential time complexity. *Coffman et al. (2013)* develop multiple approximate algorithms. Upon inspection, these algorithms are capable of resolving one-dimensional bin packing. However, it has been verified that two-dimensional bin packing algorithms cannot be examined in polynomial time. It is therefore unlikely that an exact algorithm or an effective approximate algorithm is suitable (*Mann, 2015*).

In consideration of this, recent research has been directed towards heuristic or metaheuristic algorithms (*Mollamotalebi & Hajireza, 2017*; *Chang et al., 2018*; *Satpathy et al., 2018*; *Masdari et al., 2019*). The first fit decreasing algorithm (*Keller et al., 2012*) places the PMs into a successive list and sorts the VMs in descending order according to their resource demand. When a VM is hosted, it selects the first PM that has adequate resources. The best fit decreasing algorithm (*Varasteh & Goudarzi, 2015*) imitates the first fit decreasing algorithm by placing the VMs in descending order. Subsequently, the VM is allocated to a PM which has the minimum remaining resources adequate for this VM. The modified best fit decreasing algorithm (*Esfandiarpoor, Pahlavan & Goudarzi, 2015*) arranges the VMs according to their CPU requirement. Following this, a VM is assigned to the server which produces the smallest energy increment for the DC. For this reason, the

preferred server is selected from the servers that are neither empty nor fully utilized. The objective of this algorithm is the minimization of power consumption by the DC. However, it does not consider the increase in resource utilization. The integer quadratic program with linear and quadratic constraints (*Vakilinia, 2018*) aims to optimize power in the DC. Additionally, server power consumption, migration cost, and network communication *via* VMP are optimized. However, the approach is not consistently successful as the system scales up in size. The resource aware VMP algorithm (*Gupta & Amgoth, 2018*) aims to minimize the number of active PMs to reduce power consumption and resource wastage. The approach uses a new concept: resource usage factor. This is used to balance resource optimization on active PMs. The algorithm performs no operations on the VM and PM lists, but instead computes the resource usage factor of each PM according to the requirements of the VM resource. Following this, the calculated factor is used to select a suitable PM for the VM such that resources are optimized. Although this approach maximizes resource optimization and reduces resource wastage, it does not consider the power consumed. The GA proposed by *Jamali & Malektaji (2014)* is based on the vector packing approach, and aims to minimize power consumption by maximizing resource usage and reducing the number of active PMs. The algorithm successfully reduces power consumption, however resources are used inefficiently and reduction in resource wastage is not accomplished.

The improved Lévy-based whale optimization algorithm presented by *Abdel-Basset, Abdle-Fatah & Sangaiah (2019)* is used to allocate VMs according to the current cloud computing bandwidth. The proposed approach generates an optimal balance for network load. However, the initial placement of VMs is not a feasible approach to load balancing due to the continuous increase in task number and cloud size. Recently, an approach based on an ant colony system (*Alharbi et al., 2019*) was proposed for dynamic VMP to minimize the power consumption in DCs. The approach uses a novel heuristic such that the PM with minimum power usage is designated to host VMs. This reduces DC power consumption but leads to a long execution time which is problematic for large-scale DCs. Furthermore, the effect on resource wastage was not measured. A VMP approach based on multi-cloud flower pollination optimization (*Usman et al., 2018*) aims to maximize resource utilization and reduce DC power consumption. The approach supports the use of clustering, migrations, and power effectiveness techniques. However, implementations of the approach have not successfully maximized resource utilization for the multi-cloud. The framework introduced by *Usman et al. (2019)* obtains optimal VM placements onto appropriate PMs by applying a flower pollination optimization algorithm. The approach uses a strategy known as dynamic switching probability, which aims to efficiently acquire an optimal placement solution and improves the performance of cloud DCs. However, this strategy scales poorly. *Fatima et al. (2018)* resolve the VMP problem by merging an improved Lévy-based particle swarm optimization algorithm and a variable-sized bin packing algorithm that uses the best-fit strategy. The method initializes with the standard operations of basic PSO: the swarm is applied in research space, and local and global best solutions are predicted. The remaining steps are guided based on a probability value. If the probability value is greater than 0.5, a basic PSO method is used to update the particle velocities. Otherwise, the particle velocities are updated by Lévy flight. However, this

method tends to allocate a large number of VMs onto a single PM which causes substantial congestion on the selected PM. The improved PSO approach of *Jensi & Jiji (2016)* exploits Lévy flight to obtain new particle velocities. The proposed algorithm has been verified by 21 well-known test functions, and is shown to improve the aptitude of comprehensive search and increase the efficiency of convergence. However, the algorithm is limited to linear problems.

A hybrid genetic wind-driven algorithm is proposed by *Javaid et al. (2017)*. A controller is inserted into a smart grid to manage energy for a residential area by use of a heuristic algorithm that balances the load in the grid area network. The approach is applicable to single or multiple homes, although a high request rate introduces delay. The discrete three-phase hybrid PSO algorithm proposed by *Shirvani (2020)* solves parallelizable scheduling on heterogeneous computing systems. The proposed algorithm merges discrete PSO with the hill climbing technique to avoid the local optima problem. In *Kumar & Mandal (2017)*, a cloud model of VMP is simulated using three optimization algorithms: PSO, GA, and hybrid GA PSO. The proposed model reduces the number of active physical servers, lessens power consumption, and reduces cloud resource wastage. However, the proposed model lacks the means to update the velocity and position of each particle. This capability is necessary to make long jumps toward an optimal solution. In addition, the proposed model requires substantial adjustments to achieve load balance.

VMP SYSTEM MODEL AND PROBLEM FORMULATION

This section demonstrates the VMP prefaces and applied architecture. In addition, the problem formulation is presented. Moreover, the proposed fitness function, which is central to the implementation of the proposed optimization algorithms, is presented. The fitness function is a multi-objective function which is designed to obtain the optimal values for placement time, power consumption, and resource wastage of the PMs within the cloud DC.

VMP model

The architecture of the implemented VMP model is depicted in Fig. 1. It comprises four key components: the cloud DCs, cloud information service, system mediator (including cloud broker and VMP scheduler), and the cloud users' devices. The cloud information service registers the status of each PM within the cloud DCs. In addition, it continuously notifies the cloud broker of every status update concerning the PMs that have adequate resources to host requested VMs. The status should include the service rate offered by the PM in the DC, and the expected waiting time in the queue of each DC.

Consequently, the cloud broker has two roles. First, it receives all user VM requests. Second, it converts the gathered information concerning available PMs and requested VMs into apposite vector forms and submits them to the VMP scheduler. The scheduler uses these vectors to make the placement decisions for the requested VMs using the proposed fitness function and optimization algorithms. In accordance with the scheduler decisions, the requested VMs are placed onto the appropriate PMs.

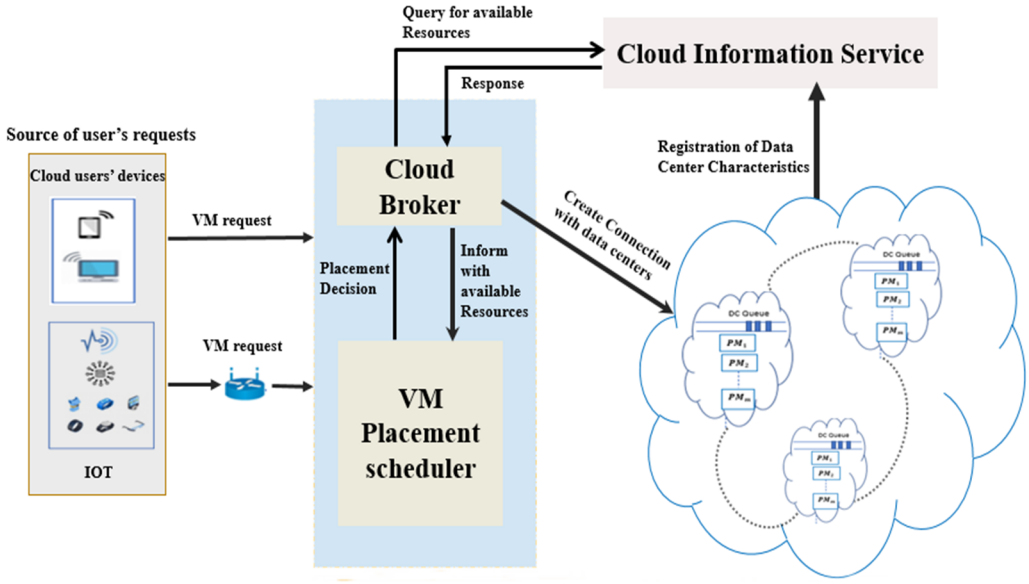

Figure 1 **The system model of VMP in cloud DCs.**

Full-size DOI: 10.7717/peerjcs.834/fig-1

Problem formulation

To streamline the formulation of the VMP model, the mathematical formulation of the VMP problem is as follows. Let $P = \{p_1, p_2, p_3, \ldots, p_N\}$ be a set of N available PMs within a given DC. We assume that the resource capacity of p_i is completely defined by its CPU and memory. Let $C = \{c_1, c_2, c_3, \ldots, c_N\}$ and $M = \{m_1, m_2, m_3, \ldots, m_N\}$ be two sets denoting respectively the CPU and memory resources of the PMs. Let $V = \{v_1, v_2, v_3, \ldots, v_M\}$ be a vector of M requested VMs. In addition, the CPU and memory requirements of these VMs can be expressed respectively as $C' = \{c_1', c_2', c_3', \ldots, c_M'\}$ and $M' = \{m_1', m_2', m_3', \ldots, m_M'\}$. To establish the mapping between the M requested VMs and the N available PMs, an $N \times M$ placement matrix $\mathbb{M}$, which defines the possible assignments of the mapping, is defined as follows:

$$\mathbb{M} = P^T \times V, \tag{1}$$

where P^T is the transpose of the row vector P. Each element $e_{ij} = p_i\ v_j \in \mathbb{M}$ defines a possible placement for each VM v_j on a single PM p_i, $1 \leq i \leq N$, $1 \leq j \leq M$. Each VM may have possible placements on multiple PMs. Hence, the decision binary variable e_{ij} can be precisely identified as:

$$e_{ij} = \begin{cases} 1, & \text{if } v_j \text{ placed on } p_i \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Let us define X to be the vector of PMs that have a probability to host at least one VM. Each decision variable $x_k \in X$, $1 \leq k \leq N$ can be expressed as:

$$x_k = \begin{cases} 1, & \sum_{j=1}^{M} e_{kj} \geq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

However, the values of each element $e_{ij} \in \mathbb{M}$ and $x_k \in X$ are set to a binary decision variable which equals 1 if and only if the following two constraints are fulfilled:

$$\frac{C^{'}}{C} \leq 1 \tag{4}$$

and

$$\frac{M^{'}}{M} \leq 1. \tag{5}$$

The normalized CPU utilization U_i^{cpu} offered by the PM p_i to all hosted VMs can be defined as follows:

$$U_i^{cpu} = \sum_{j=1}^{M} \frac{e_{ij} c_j^{'}}{c_i}. \tag{6}$$

The normalized RAM utilization U_i^{ram} offered by the PM p_i to all hosted VMs can be defined as follows:

$$U_i^{ram} = \sum_{j=1}^{M} \frac{e_{ij} m_j^{'}}{m_i}. \tag{7}$$

The role of the VMP scheduler is to obtain the optimal placement solution with respect to three objectives: minimization of placement time, minimization of power consumption, and minimization of resource wastage. The following subsections introduce the mathematical formulas for these three components.

Modeling placement time

This subsection derives a mathematical expression for the placement time T_j of a VM v_j. The placement time T_j is measured from the instant at which v_j is requested to the expected instant at which v_j is to be placed onto a specified PM p_i. From this definition, T_j consists of two components.

1. The search time S_j is the time required by v_j to find all the available PMs $x_k \in X$ that fulfill its requirements, where $|X| = N$. Hence, the value of S_j can be written as
$$S_j = \sum_{k=1}^{N} x_k \left(T \left(\frac{c_j^{'}}{c_k} + \frac{m_j^{'}}{m_k} \right) + \delta_k \right), \tag{8}$$
where T is the time required to find appropriate PMs on which to place the requested VMs, and δ_k is the search time factor representing the time delay for each VM cycle.
2. The expected waiting time $E[W_j]$ of v_j in the cloud DC consists of the VM queuing time and the VM service time. Let the arrival process of a given VM at the cloud DC

be a Poisson process with arrival rate λ. Let the service time of a VM be exponentially distributed with expectation $\frac{1}{\mu}$. We assume that the DC has an infinite queue for the hosting of all arrived VMs on N PMs. From this physical description, the system dynamics (*i.e.,* arrival, service, and departure) can be modeled as the well-known multi-server queuing system M/M/N, where $\lambda < \mu$ is the steady state condition. The expected waiting time $E[W_j]$ of M/M/N can be written as (*White, 2012*)

$$E[W_j] = \frac{\mu\left(\frac{\lambda}{\mu}\right)^N p_0}{(N-1)!(N\mu-\lambda)^2} + \frac{1}{\mu}, \tag{9}$$

where p_0 is the probability of an empty queue and is given by

$$p_0 = \left[\sum_{n=0}^{N-1} \frac{(\frac{\lambda}{\mu})^n}{n} + \frac{(\frac{\lambda}{\mu})^n}{N!(1-\frac{\lambda}{N\mu})}\right]^{-1}. \tag{10}$$

Using Equations Eqs. (8) and (9), the placement time t_j of v_j is given as

$$t_j = S_j + E[W_j]. \tag{11}$$

The total placement time T_j can be calculated as follows:

$$T_j = \sum_{i=0}^{N} e_{ij} t_j. \tag{12}$$

Finally, the total placement time T for all VMs can be calculated using Eq. (12):

$$T = \sum_{j=0}^{M} T_j. \tag{13}$$

Modeling power consumption

Jin et al. (2020) accurately modeled power consumption linearly as a function of the resource utilization of the PMs in the cloud DC. In this model, power consumption is based only on CPU utilization. Additionally, unused PMs are deactivated to save power. Hence, power consumption can be computed for each PM $p_i \in P$ as follows. Let u_i^{max} and u_i^{idle} be the power consumption of PM p_i at maximum CPU utilization and idle state, respectively. The power consumption PU_i of the PM p_i due to the VM v_j is given as

$$PU_i = u_i^{idle} + \left(u_i^{max} - u_i^{idle}\right)\frac{e_{ij}c_j^{'}}{c_i}. \tag{14}$$

The power consumption $PU_i^{'}$ of the PM p_i due to all VMs is given using Equation (6) as

$$PU_i^{'} = u_i^{idle} + \left(u_i^{max} - u_i^{idle}\right)U_i^{cpu}. \tag{15}$$

From Eqs. (14) and (15), the total power consumption PU of all PMs in a DC can be computed as follows:

$$PU = \sum_{i=0}^{N} x_i \times PU_i^{'}. \tag{16}$$

Modeling resource wastage

The residual resources offered by each physical machine may vary according to the VMP strategy. To maximize the utilization of multiple resources such as CPU and RAM, the resource wastage W_i of the PM p_i can be calculated as follows:

$$W_i = \frac{\left| r_i^{cpu} - r_i^{ram} \right|}{U_i^{cpu} + U_i^{ram}} + \varepsilon, \tag{17}$$

where r_i^{cpu} and r_i^{ram} represent the normalized residual CPU and memory resources; the values of U_i^{cpu} and U_i^{ram} are given in Eqs. (6) and (7); and $\varepsilon = 0.001$, a small positive real number (*Gupta & Amgoth, 2016*). This model and its corresponding objective are included to make best use of the resources of all PMs and achieve balance between multiple residual resources. The total resource wastage W of all PMs in a DC is given by

$$W = \sum_{i=1}^{N} x_i \frac{\left| \left(U_i^{cpu} - \sum_{j=1}^{M} e_{i,j} c_j^{'} \right) - \left(U_i^{ram} - \sum_{j=1}^{M} e_{i,j} m_j^{'} \right) \right| + \varepsilon}{\sum_{j=1}^{M} e_{i,j} c_j^{'} + \sum_{j=1}^{M} e_{i,j} m_j^{'}}. \tag{18}$$

Finally, the placement problem can be formulated:

$$Minimize \quad T. \tag{19}$$

$$Minimize \quad PU. \tag{20}$$

$$Minimize \quad W. \tag{21}$$

Subject to the constraints

$$\sum_{i=1}^{N} e_{i,j} = 1, \qquad \forall j = 1, 2, \ldots, M, \tag{22}$$

$$\sum_{j=1}^{M} e_{i,j} c_j^{'} < c_i, \qquad \forall i = 1, 2, \ldots, N, \tag{23}$$

$$\sum_{j=1}^{M} e_{i,j} m_j^{'} < m_i, \qquad \forall i = 1, 2, \ldots, M, \tag{24}$$

and

$$x_i, e_{i,j} \in \{0, 1\}, \qquad \forall i = 1, 2, \ldots, N, j = 1, 2, \ldots, M. \tag{25}$$

From Eq. (22), each VM can be placed only on a single PM. Equations (23) and (24) stipulate that the summation of CPU and RAM respectively of all VMs hosted on a given PM must not surpass that PMs CPU and RAM capacity. Finally, Eq. (25) formalizes the domains of the variables M and N. Hence, there are M^N possible solutions for the placement problem.

The fitness function

The primary goal of this work is to obtain the optimal solution for VMP. This is accomplished by the use of a fitness function, which allows the optimizer to select the optimal solution as a function of the three objectives described in Eqs. (19)–(21). The fitness function is created from the three objective functions by using the scalarization method (*Gunantara, 2018*):

$$F(x) = w_1 f_1(x) + w_2 f_2(x) + \cdots + w_n f_n(x), \tag{26}$$

where $F(x)$ is the total fitness function and $w_1, w_2, \ldots, w_n$ represent the weight values given to each of the objective functions. Each weight value is determined by the priority of its corresponding objective function within the total fitness function. The weight value has a vital role in governing the performance of the corresponding objective function within the total fitness function (*Giagkiozis & Fleming, 2015*). Hence, the weight values should be determined prior to optimization. In this paper, equal weighting is assigned to each component of the fitness function. This approach is used to equalize priorities for each objective: minimization of placement time, minimization of power consumption, and minimization of resource wastage.In this case each weight can be calculated as $w_i = 1/n$, where n is the number of objective functions. Accordingly, the equal weight value of each objective function is $w_i = 1/3$.

THE IMPLEMENTATION OF THE PROPOSED FITNESS FUNCTION USING PSO, FPO AND HPSOLF-FPO

This section presents the proposed nature-inspired algorithms for cloud computing VMP: PSO, FPO, and PSOLF-FPO. These algorithms use the proposed fitness function to find the optimal placements for a set of VMs on available PMs within a cloud DC while minimizing total placement time, total power consumption, and resource wastage.

The PSO algorithm

The PSO algorithm is modeled on the social behavior of fish and bird swarms (*Clerc, 2010*). The following subsections describe the creation and configuration of the set of particles that embody the swarm. In addition, the parameters of PSO are described in detail.

Particle encoding

Each particle represents a candidate solution for the placement of a set of VM requests on a set of PMs. The placement matrix $\mathbb{M}$, defined in Eq. (1), is used to initiate the particles. In this manner, each created particle represents one of the M^N possible solutions to the placement problem.

For example, consider two different particles created from $\mathbb{M}$ in the initial swarm as shown in Fig. 2. Each particle is created in a two-dimensional scheme that uses one-to-many maps between each PM and its hosted VMs within the particle.

Particle evaluation

The optimal solution for placement is obtained through successive iterations of swarm regeneration. A new swarm is generated whenever the position and velocity of the particles

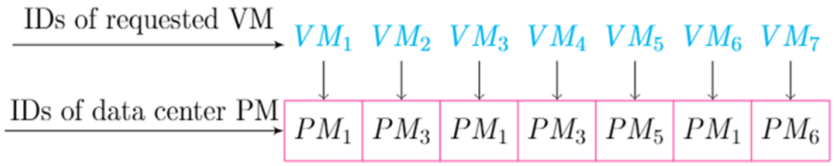

Figure 2 **An example of particle encoding.**

Full-size DOI: 10.7717/peerjcs.834/fig-2

are updated. The equations for particle position and velocity as implemented in the standard PSO and PSOLF algorithms are as follows. Using PSO, the velocity of each particle is updated as

$$V_i(t+1) = \omega \times V_i(t) + \rho_1 \times c_1 \times (P_{lbest} - P_i(t)) + \rho_2 \times c_2 \times (P_{gbest} - P_i(t)), \tag{27}$$

where i is the particle number within the swarm, ω is the inertia weight which determines the impact of the previous velocity on the current velocity, V_i(t) is the current particle velocity, V_i(t + 1) is the new particle velocity, P_i(t) is the current position of the particle within the swarm, C_1 and C_2 are learning factors of the particle and swarm, and ρ_1 and ρ_2 are uniformly distributed random variables between 0 and 1. Using PSO, the position of each particle is updated as

$$P_i(t+1) = P_i(t) + V_i(t+1), \tag{28}$$

where $P_i(t+1)$ is the new position of the particle within the search space, and $P_i(t)$ is the current position. Standard PSO has a defect known as premature convergence that pushes the particles to converge too early and become trapped within local optima (*Nakisa et al., 2014*). To overcome this problem, Lévy flight can be combined with PSO to produce PSOLF. This reduces early particle convergence by updating velocities in a manner that causes the particles to take a long step towards the optimal solution (*Haklı & Uğuz, 2014*). When using Lévy flight, the particle positions and velocities are updated as follows. Using PSOLF, the velocity of each particle is updated as (*Jensi & Jiji, 2016*)

$$V_i(t+1) = \omega \times L \times V_i(t) + \rho_1 \times c_1 \times (P_{lbest} - P_i(t)) + \rho_2 \times c_2 \times (P_{gbest} - P_i(t)), \tag{29}$$

where L is a step size emulating the large distance. This can be calculated using Lévy flight:

$$L(s,a) \sim \frac{\lambda \times \Gamma(\lambda) \times \sin\frac{(\pi\lambda)}{2}}{\pi}, \tag{30}$$

where $\Gamma(\lambda)$ is the gamma function with index λ, $a = 1$ is a control parameter of the distribution, and S is a large step which is given by

$$S = \frac{U}{|V|^{(\lambda-1)}}. \tag{31}$$

The parameters U and V are drawn from a Gaussian normal distribution and are given by

$$U \sim N(0, \sigma_u^2), V \sim N(0, \sigma_v^2), \tag{32}$$

where

$$\sigma_u = \left[\frac{\Gamma(1+\lambda)}{\lambda\Gamma(\frac{(1+\lambda)}{2})} \times \frac{\sin(\frac{\pi\lambda}{2})}{2^{\frac{(\lambda-1)}{2}}} \right]^{\frac{1}{\lambda}}. \tag{33}$$

The size of each step is calculated using *Jensi & Jiji (2016)*

$$stepsize = 0.01 * S. \tag{34}$$

Using PSOLF, the position of each particle is updated as

$$P_i(t+1) = V_i(t+1). \tag{35}$$

The PSOLF-based VMP algorithm stages

Before describing the proposed PSOLF-based VMP algorithm, there are some preparatory PSO principles that should be clarified and described. In the search space, each particle has position P_i and velocity V_i. Both should be computed at each iteration of the swarm evaluation. In addition, each particle has a key criterion known as the local best solution P_{l-best}. During each iteration, P_{l-best} is updated with the particles computed position value if the computed value is less than P_{l-best}. The swarm itself has an analogous criterion known as the global best solution P_{g-best}. Generally, the global best solution is given by the minimum value of the local best solution among all particles.

The proposed algorithm consists of three principal operations: swarm initialization, swarm evaluation, and termination.

- **Swarm initialization.** The swarm is initialized by generating a particle vector P from the placement matrix $\mathbb{M}$. All initiated particles are restricted by the constraints in Eqs. (22)–(25). The number of particles in the swarm is N_{swarm}. For each particle, the initial position and velocity are generated randomly based on the particles index within the vector P. The position of each particle is set to its corresponding local best solution P_{l-best}. The minimum value of P_{l-best} among all particles is assigned to the global best P_{g-best} of the swarm. In addition, the required PSO parameters are defined, including the learning factors c_1, c_2, inertia weight coefficient ω, and the random variables ρ_1, ρ_2 $\in$ [0,1]. Moreover, the random function *rand*() is defined to generate random numbers between [0,1], and the maximum number of iterations is set to Max_{iter}. Finally, the fitness function of all particles is computed using Eq. (26) to obtain the local fitness of each particle.
- **Swarm evaluation.** During each iteration, the particle velocities and positions are updated by either PSO or PSOLF. The choice between algorithms is dependent upon the value generated by *rand*(). If the value of *rand*() is less than 0.5, the particle velocities and positions are updated as in Eqs. (29) and (35). Else, the particle velocities and positions are updated as in Eqs. (27) and (28). Subsequently, the fitness values associated with the updated particle position P_u and the position P_{l-best} are compared. If the fitness value of P_u is less than the fitness value of P_{l-best}, then P_{l-best} is set to P_u. Furthermore, if the

fitness value of P_{l-best} is less than the fitness value of the global best position P_{g-best}, then P_{g-best} is set to P_{l-best}.

- **Swarm termination.** Swarm evaluation continues iteratively until the maximum number of iterations Max_{iter} is reached. Following the final iteration, the particle corresponding to P_{g-best} is selected to represent the optimal fitness value for the VMP solution.

The pseudo-code for the PSOLF algorithm is presented below.

Algorithm 1: The proposed PSOLF-based virtual machine placement algorithm

Input: N_{swarm}, Max_{iter}, ρ_1, ρ_2, ω
Output: $P_{gbest} \leftarrow (optimal_{Particles(PM)}$ and $optimal_{fitness})$

```
Possible particles P ← Placement matrix;
for i = 1 to N_swarm do
    P_position ←Random position (P) ;
    P_velocity ← Random velocity (P) ;
    Calculate fitness function using Equation 26;
    Fitness ← Record fitness value
    P_best ←P_Position;
end
P_gbest ← Min(Fitness)
for t=1 to Max_iter
    for i = 1 to N_swarm do
        if rand()<0.5 then
            P_velocity ←Update velocity using Equation 29;
            P_position ←Update position using Equation 35;
        else
            P_velocity ←Update velocity using Equation 27;
            P_position ←Update position using Equation 28;
        end
        P_u ←P_position
        if P_u.fitness <= P_best.fitness then
            P_best ←P_u;
        end
        if P_best.fitness <= P_gbest.fitness then
            P_gbest ←P_best;
        end
    end
end
```

The FPO algorithm

Developed by *Yang (2012)*, FPO is a fascinating algorithm based on the process of flower pollination in flowering plants. It employs features of the pollination process to evaluate both global pollination and local pollination (*Yang, Karamanoglu & He, 2014*).

Pollen encoding

Each pollen grain encodes a possible VMP solution. The placement matrix $\mathbb{M}$ can be used to initialize the pollen as in the case of the particle swarm. An example of the pollen

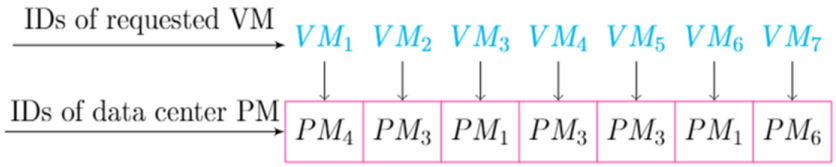

Figure 3 **An example of pollen encoding.**
Full-size DOI: 10.7717/peerjcs.834/fig-3

initialization is shown in Fig. 3. Each pollen grain is created in a two-dimensional scheme that uses one-to-many maps between each PM and its hosted VMs within the pollen grain.

Pollen evaluation

The two principal steps of the FPO algorithm are represented by two pollination methods: global pollination and local pollination. The method used is selected based on a switching probability value β. The optimal placement solution is obtained through a series of iterations. During each iteration a random number between [0,1] is generated. This number is compared with the switch probability β. If the random number is less than β, global pollination will be used and updated using the Lévy distribution. Else, local pollination is used. When using global pollination, the pollen vectors are updated as

$$S_i^{(t+1)} = S_i^t + L \times (S_i^t - g_{best}^*), \tag{36}$$

where S_i^t is the ith pollen vector (solution) at iteration t, g_{best}^* is the global best solution at the current iteration, and L is the pollination strength and is calculated in the same manner as the step size using Eqs. (30)–(34). When using local pollination, the pollen vectors are updated as

$$S_i^{(t+1)} = S_i^t + \varepsilon[S_{r1}^t - S_{r2}^t], \tag{37}$$

where S_{r1}^t and S_{r2}^t are a random selection of pollen grains (solutions) for local pollination.

The FPO-based VMP algorithm stages

The FPO algorithm consists of three principal operations: pollen initialization, population evaluation, and termination.

- **Pollen initialization.** To initialize the pollen, the placement matrix $\mathbb{M}$ is used to generate all possible placement solutions S_i. All solutions must obey the placement constraints in Eqs. (22)–(25). Each solution S_i is assigned randomly to a pollen grain. In addition, the fitness function for each pollen grain is computed using Eq. (26). The minimum fitness value among the pollen is selected as g_{best}^*. Furthermore, the population size is set to N_{pop}, and the maximum number of iterations is set to N_{iter}. Finally, the switching probability is defined as $\beta \in [0,1]$.
- **Population evaluation.** During each iteration a random number $\in [0,1]$ is generated and compared with the switching probability β. If the random number is less than β then global pollination and Eq. (36) are used. Else, local pollination and Eq. (37) are used. Subsequently, the results of the new solutions S_u are evaluated using the pre-computed

fitness functions. If the fitness value of the new solution is less than the fitness value of g^*_{best}, then g^*_{best} is set to the new solution.

- **Termination.** Population evaluation continues iteratively until the maximum number of iterations N_{iter} is reached. Following the final iteration, g^*_{best} is selected as the optimal VMP solution.

The pseudo-code for the FPO algorithm is presented below.

Algorithm 2: The proposed FPO-based virtual machine placement algorithm

Input: N_{pop}, N_{iter}, $\beta \in [0,1]$
Output: $g^*_{best} \leftarrow$ (Optimal $_{pollen(PM)}$, $Optimal_{fitness}$)

```
Possible solutions S ← Placement matrix;
for i = 1 to N_pop do
    S_i ←Random solution(S)
    Calculate fitness function using Equation 26;
    Find the current best among the initial population:
    g*_best ←Min_fitness;
end
for t=1 to N_iter
    for i = 1 to N_pop do
        rand ← ∈ [0,1]
        if rand< β then
            Calculate a (d-dimensional) Lévy distribution step vector L using Equation 30;
            Update the i^th solution according to global pollination using Equation 36;
        else
            Create ε from a uniform distribution in[0,1]
            Randomly choose s_r1 and s_r2 from the current population
            Update the i^th solution according to local pollination using Equation 37;
        end
        Evaluate the new solutions S_u
        if S_u.fitness < g*_best.fitness then
            g*_best ←S_u
        end
    end
end
```

The hybrid PSOLF-FPO algorithm

The trap of local optima is the primary obstacle to the use of PSO. This problem causes particle velocities to decrease quickly over successive iterations. Consequently, the particle positions will converge and search the local area of the same peak. Thus, the resulting solution may not be optimal, due to the exclusion of other possible solutions further away from the local peak. Despite the use of Lévy flight to update the particle velocities (*Hariya et al., 2015*), the problem remains, as the Lévy flight implementation ensures that the particle velocities will decrease after several iterations. Consequently, the local optima may disappear for several iterations before reappearing. In addition, the PSOLF algorithm tends to allocate a large number of VMs onto a single PM (*Fatima et al., 2018*). Such an allocation causes substantial congestion on the PM. Hence, this subsection proposes a hybrid VMP algorithm, HPSOLF-FPO, to overcome the local optima problem. The proposed algorithm improves the optimal solution discovery process, as the FPO algorithm is capable of updating solutions in the search space to continuously valued positions. The

hybrid algorithm accomplishes three objectives. First, it achieves the required exploitation and exploration capabilities. Second, it enhances the search accuracy. Finally, it increases the probability of finding the global best solution to the problem. The algorithm combines the advantages of PSO local search with FPO global search. This overcomes the weakness of the global search capability of PSO and PSOLF, guaranteeing better placement decisions. The hybrid algorithm consists of three principal operations: population initialization, population evaluation, and termination.

- **Population initialization** As for swarm initialization ('The PSOLF-based VMP algorithm stages'), the N_{swarm} particles are derived from the placement matrix $\mathbb{M}$. The initial particle position and velocity are set, along with the particle fitness functions, particle local best P_{l-best} and the swarm global best P_{g-best}. During each iteration, the equations used to update the particle velocities are determined by the value of $\beta \in [0,1]$, where the switching probability β is used in place of the constant switching value 0.5, to select between Lévy flight equations or basic PSO equations. The use of β improves results. Furthermore, β can be controlled and changed for different experiments. As for pollen initialization ('The FPO-based VMP algorithm stages'), the pollen grains that represent the possible solutions S_i are derived from the placement matrix $\mathbb{M}$. The initial pollen parameters are set: position S_i, fitness function, new solution S_u and best fitness value g^*_{best}. Consequently, all possible solutions represented by the N_{swarm} particles of the PSO module have corresponding pollen grains in the FPO module.
- **Population evaluation.** Across successive iterations, the population is evaluated across three modules: the PSO module, the FPO module, and the update module. During each iteration, the PSO module will compute all updated particle parameters. However, the particle parameters are not updated. Instead, the updated parameters are moved to the update module. In addition, the new particle positions P_u are input to the FPO module. According to the value of the switching probability, the FPO module performs either global pollination or local pollination. Consequently, all corresponding pollen parameters are updated. Within the update module, for each solution the updated position S_u generated by the FPO module is compared to the corresponding solution P_u generated by the PSO module. The lowest of the two is selected to update the corresponding position of both solutions in the PSO module and FPO module. Based on the updated position for each new solution, the remaining parameters are updated. They include the fitness function, P_{l-best}, and P_{g-best}. The merging of the updated solutions from both the PSO module and the FPO module successfully overcomes the local optima problem.
- **Termination** Population evaluation continues iteratively until the maximum number of iterations Max_{iter} is reached. Following the final iteration, P_{g-best} is selected as the optimal VMP solution.

The pseudo-code for the HPSOLF-FPO algorithm is presented below.

Algorithm 3: The proposed HPSOLF-FPO based virtual machine placement algorithm

Input: N_{swarm}, Max_{iter}, ρ_1, ρ_2, ω
Output: $P_{gbest} \leftarrow (optimal_{Particles(PM)}$ and $optimal_{fitness})$

Possible placement solutions $S \leftarrow$ Placement matrix
for i = *1 to* N_{swarm} **do**
  $P_{position} \leftarrow$Random(S) ;
  $P_{velocity} \leftarrow$ Random(S);
  Evaluate fitness function using Equation 26;
  Fitness $\leftarrow$ Record fitness values
  $P_{l-best} \leftarrow$P$_{Position}$;
end
$P_{gbest} \leftarrow$ Min(Fitness)
Determine the switch probability $\beta \in [0, 1]$
for t=1 to Max_{iter}
  for i = *1 to* N_{swarm} **do**
    if $rand() < \beta$ **then**
      Calculate Lévy distribution step vector L using Equation 30;
      $P_{velocity} \leftarrow$Update velocity using Equation 29;
      $P_{position} \leftarrow$Update position using Equation 35;
    else
      $P_{velocity} \leftarrow$Update velocity using Equation 27;
      $P_{position} \leftarrow$Update position using Equation 28;
    end
    $P_u \leftarrow$Record updated solutions
  end
  \\Hybrid flower pollination algorithm block \\.
  for *each* P_u **do**
    if $rand() < \beta$ **then**
      Compute Lévy distribution step vector L using Equation 30;
      $S_i \leftarrow$update i^{th} solution by global pollination using Equation 36;
    end
    else
      Create ε from a uniform distribution in[0,1]
      Randomly choose s_{r1} and s_{r2} from the current population
      $S_i \leftarrow$update i^{th} solution by local pollination using Equation 37;
    end
    Evaluate updated solution S_i;
    $S_u \leftarrow$ Record updated solutions.
  end
  \\End of hybrid flower pollination algorithm block \\.
  if $S_u.fitness < P_u.fitness$ **then**
    $P_u \leftarrow$S$_u$
  end
  if $P_u.fitness <= P_{lbest}.fitness$ **then**
    update $P_{lbest} = P_u$;
  end **if** $P_{lbest}.fitness <= P_{gbest}.fitness$ **then**
    update $P_{gbest} = P_{lbest}$;
  end
end

SIMULATION EVALUATION

This section compares the results of experiments with the three proposed algorithms: PSOLF, FPO, and HPSOLF-FPO. The three algorithms are simulated using a MATLAB tool to generate 2000 VM requests. The remaining implementation parameters are listed in Table 1. The results from the three algorithms are compared to measure their performance

Table 1 Parameter settings of the proposed algorithms.

Parameter name	Value
Number of iterations Max_{iter}	100
Population size N_{pop}	100
ω	2
C_1	2
C_2	2
ρ_1	U[0–1]
ρ_2	U[0–1]
Switch probability β	U[0–1]

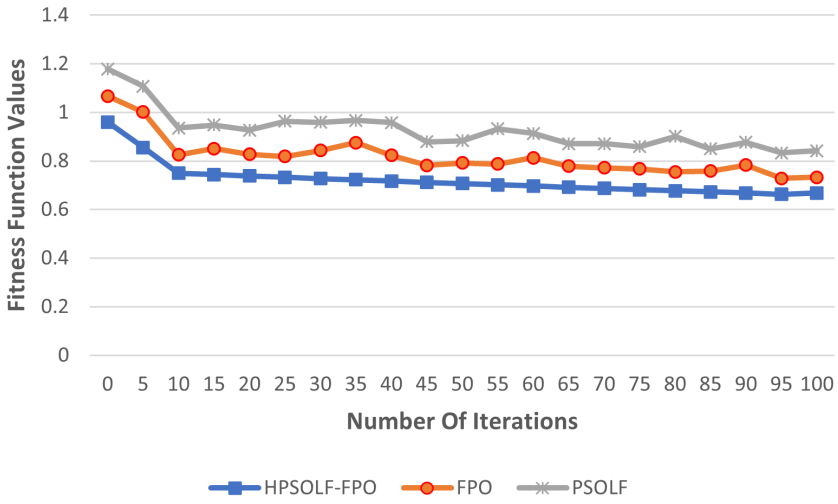

Figure 4 The fitness function values of the proposed VMP algorithms.
Full-size DOI: 10.7717/peerjcs.834/fig-4

in terms of placement time, power consumption, resource utilization (inferred from the number of active servers), and resource wastage.

Fitness function performance

Figure 4 shows the values of the proposed fitness function against the experimental iterations for the three algorithms. The results show that the HPSOLF-FPO algorithm outperforms the other two algorithms in terms of the rate at which the fitness value decreases. In addition, the HPSOLF-FPO algorithm obtained the lowest fitness value in the smallest number of iterations.

Placement time performance

Figure 5 shows the average placement time for the three algorithms. The average placement times increase as the number of VM requests increase for each algorithm. However, the highest average placement times are generated by the PSO algorithm, while the lowest average are produced by the HPSOLF-FPO algorithm. Hence, the HPSOLF-FPO algorithm effectively reduces placement times.

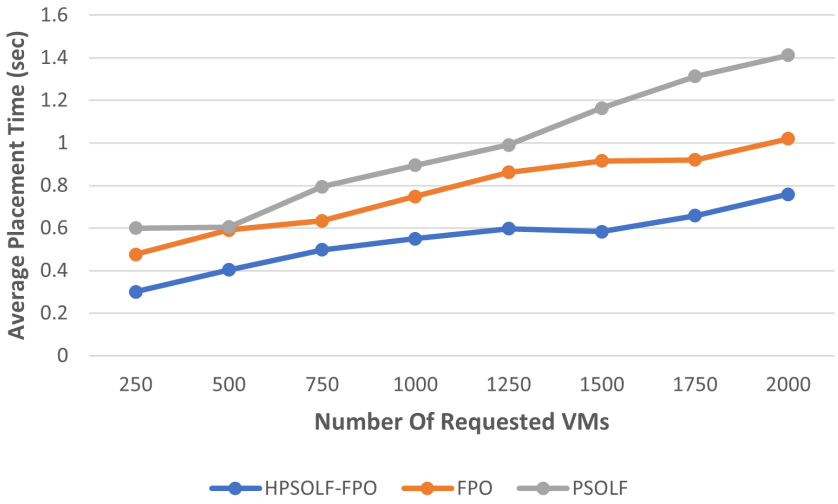

Figure 5 **The average placement time against the number of requested VMs.**
Full-size DOI: 10.7717/peerjcs.834/fig-5

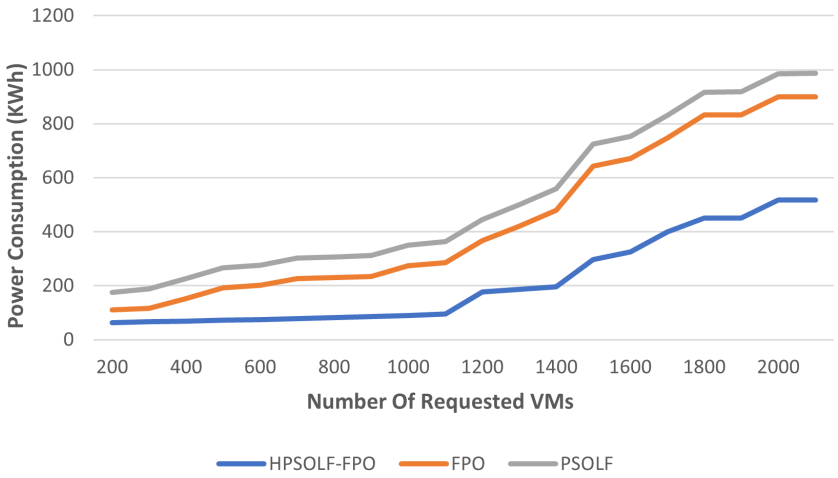

Figure 6 **The power consumption against the number of requested VMs.**
Full-size DOI: 10.7717/peerjcs.834/fig-6

Power consumption performance

The total power consumption of all active servers for the three proposed algorithms is shown in Fig. 6. Power consumption increases as the number of VM requests increases for all algorithms. However, as for the number of requested VMs, the HPSOLF-FPO algorithm always has lower power consumption than the other algorithms. The placement strategy when using the PSO algorithm produces the highest power consumption.

Resource utilization performance

To evaluate the performance of the algorithms in term of resource wastage, two different experiments were performed. The first experiment used each of the three algorithms in addition to a best-fit bin packing (BP) strategy. During the experiment the number of active

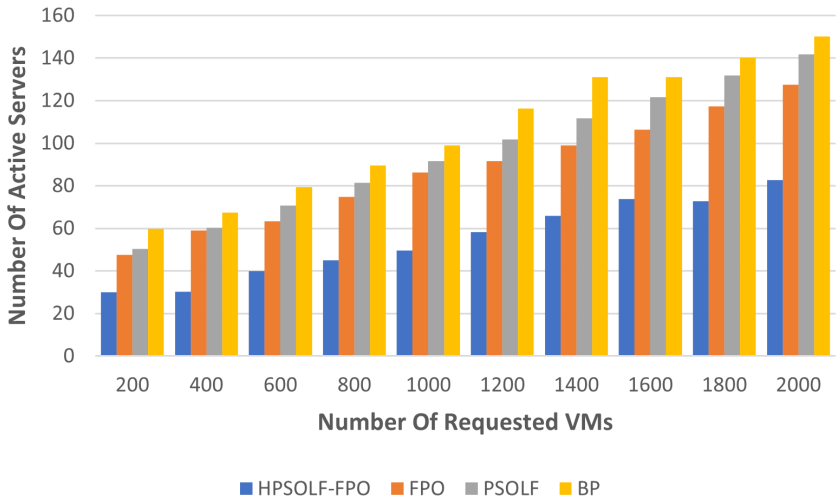

Figure 7 **Active servers against the number of requested VMs.**
Full-size DOI: 10.7717/peerjcs.834/fig-7

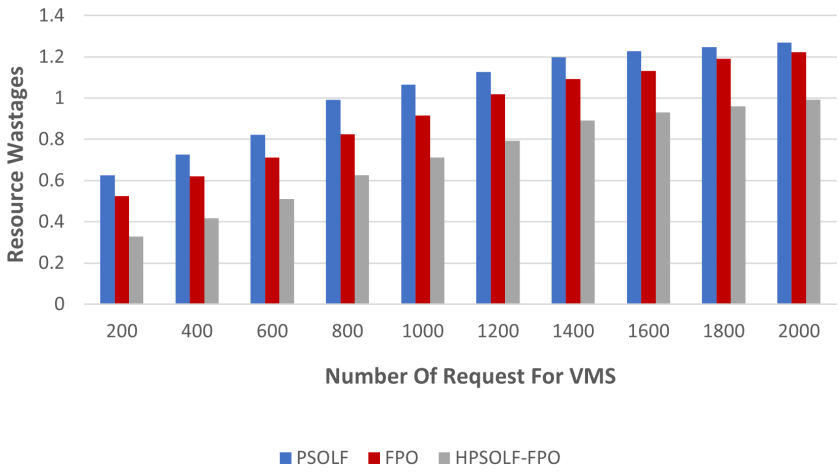

Figure 8 **Resource wastage against the number of requested VMs.**
Full-size DOI: 10.7717/peerjcs.834/fig-8

servers was used to indicate the utilization rate of the PMs. Figure 7 shows the experimental results comparing all algorithms when using the same number of user requests. The three optimization algorithms outperform the best-fit bin packing strategy. The HPSOLF-FPO algorithm outperforms all other algorithms and has the minimum number of active servers. Hence, it effectively maximizes resource utilization in the cloud DC.

The second experiment was performed using the three optimization algorithms to measure resource wastage. Figure 8 demonstrates the results. It can be observed that, with an increasing number of VM requests, the HPSOLF-FPO algorithm wastes fewer resources than the PSOLF and FPO algorithms. These results are obtained due to the decision making

process of the HPSOLF-FPO algorithm which accounts for available remaining resources and uses them in a balanced manner to achieve minimum resource wastage.

CONCLUSION AND FUTURE WORK

This paper presented an efficient multi-objective VMP strategy for the cloud computing environment. The proposed model aims to determine the optimal VMP solution among all possible VMP solutions. The HPSOLF-FPO algorithm was developed to combine the exploration capabilities of FPO with the exploitation capabilities of PSO. The algorithm can move between FPO and PSO as needed. When trapped in local optima, the algorithm utilizes FPO to move away. As the local optima disappear, the algorithm returns to using PSO to improve its ability to obtain an optimal solution. The optimal VMP solution was evaluated based on a fitness function that combines the values of three criteria: the total placement time of requested VMs, power consumption, and resource wastage in the cloud DC. The HPSOLF-FPO, PSOLF, and FPO algorithms were evaluated based on the proposed fitness function. The experimental results of the simulation evaluations showed that the proposed HPSOLF-FPO algorithm is more efficient than the PSOLF or FPO algorithms. Furthermore, when the server utilization was measured, the HPSOLF-FPO algorithm outperformed the bin packing best-fit strategy. In future work, the proposed algorithms may be used to fulfil additional VMP objectives such as load balancing, live migration, and cost minimization. In addition, the proposed algorithm in combination with machine learning techniques will be employed to serve real-time or non-real-time tasks in a cloud computing environment.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

The authors received no funding for this work.

Competing Interests

The authors declare there are no competing interests.

Author Contributions

- Sara Mejahed and M Elshrkawey conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The raw data and code are available in the Supplemental File.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.834#supplemental-information.

REFERENCES

Abdel-Basset M, Abdle-Fatah L, Sangaiah AK. 2019. An improved Lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment. *Cluster Computing* **22(4)**:8319–8334 DOI 10.1007/s10586-018-1769-z.

Adamuthe AC, Pandharpatte RM, Thampi GT. 2013. Multiobjective virtual machine placement in cloud environment. In: *2013 international conference on cloud & ubiquitous computing & emerging technologies*. Piscataway: IEEE, 8–13.

Addya SK, Turuk AK, Sahoo B, Sarkar M. 2015. A hybrid queuing model for virtual machine placement in cloud data center. In: *2015 IEEE international conference on advanced networks and telecommuncations systems (ANTS)*. Piscataway: IEEE, 1–3.

Alashaikh AS, Alanazi EA. 2019. A survey of preferences in virtual machine placement. ArXiv preprint. arXiv:1907.07778.

Alboaneen DA, Tianfield H, Zhang Y. 2016. Metaheuristic approaches to virtual machine placement in cloud computing: a review. In: *2016 15th international symposium on parallel and distributed computing (ISPDC)*. Piscataway: IEEE, 214–221.

Alharbi F, Tian Y-C, Tang M, Zhang W-Z, Peng C, Fei M. 2019. An ant colony system for energy-efficient dynamic virtual machine placement in data centers. *Expert Systems with Applications* **120**:228–238 DOI 10.1016/j.eswa.2018.11.029.

Attaoui W, Sabir E. 2018. Multi-criteria virtual machine placement in cloud computing environments: a literature review. ArXiv preprint. arXiv:1802.05113.

Chang Y, Gu C, Luo F, Fan G, Fu W. 2018. Energy efficient resource selection and allocation strategy for virtual machine consolidation in cloud datacenters. *IEICE Transactions on Information and Systems* **101(7)**:1816–1827.

Clerc M. 2010. Particle swarm optimization. Vol. 93. London, UK: John Wiley & Sons.

Coffman EG, Csirik J, Galambos G, Martello S, Vigo D. 2013. Bin packing approximation algorithms: survey and classification. In: *Handbook of combinatorial optimization*. New York, NY: Springer New York, 455–531 DOI 10.1007/978-1-4419-7997-1_35.

Donyagard Vahed N, Ghobaei-Arani M, Souri A. 2019. Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: a comprehensive review. *International Journal of Communication Systems* **32(14)**:e4068 DOI 10.1002/dac.4068.

Esfandiarpoor S, Pahlavan A, Goudarzi M. 2015. Structure-aware online virtual machine consolidation for datacenter energy improvement in cloud computing. *Computers & Electrical Engineering* **42**:74–89 DOI 10.1016/j.compeleceng.2014.09.005.

Farzai S, Shirvani MH, Rabbani M. 2020. Multi-objective communication-aware optimization for virtual machine placement in cloud datacenters. *Sustainable Computing: Informatics and Systems* **28**:100374.

Fatima A, Javaid N, Sultana T, Hussain W, Bilal M, Shabbir S, Asim Y, Akbar M, Ilahi M. 2018. Virtual machine placement via bin packing in cloud data centers. *Electronics* **7(12)**:389 DOI 10.3390/electronics7120389.

Gahlawat M, Sharma P. 2014. Survey of virtual machine placement in federated clouds. In: *2014 IEEE international advance computing conference (IACC)*. Piscataway: IEEE, 735–738.

Ghobaei-Arani M, Shamsi M, Rahmanian AA. 2017. An efficient approach for improving virtual machine placement in cloud computing environment. *Journal of Experimental & Theoretical Artificial Intelligence* **29(6)**:1149–1171 DOI 10.1080/0952813X.2017.1310308.

Giagkiozis I, Fleming PJ. 2015. Methods for multi-objective optimization: an analysis. *Information Sciences* **293**:338–350 DOI 10.1016/j.ins.2014.08.071.

Gunantara N. 2018. A review of multi-objective optimization: methods and its applications. *Cogent Engineering* **5(1)**:1502242 DOI 10.1080/23311916.2018.1502242.

Gupta MK, Amgoth T. 2016. Resource-aware algorithm for virtual machine placement in cloud environment. In: *2016 ninth international conference on contemporary computing (IC3)*. Piscataway: IEEE, 1–6.

Gupta MK, Amgoth T. 2018. Resource-aware virtual machine placement algorithm for IaaS cloud. *The Journal of Supercomputing* **74(1)**:122–140 DOI 10.1007/s11227-017-2112-9.

Gupta RK, Pateriya R. 2014. Survey on virtual machine placement techniques in cloud computing environment. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)* **4(4)**:1–7.

Haklı H, Uğuz H. 2014. A novel particle swarm optimization algorithm with Levy flight. *Applied Soft Computing* **23**:333–345 DOI 10.1016/j.asoc.2014.06.034.

Hariya Y, Kurihara T, Shindo T, Jin'no K. 2015. Lévy flight PSO. In: *2015 IEEE congress on evolutionary computation (CEC)*. Piscataway: IEEE, 2678–2684.

Hoang N-D, Bui DT, Liao K-W. 2016. Groutability estimation of grouting processes with cement grouts using differential flower pollination optimized support vector machine. *Applied Soft Computing* **45**:173–186 DOI 10.1016/j.asoc.2016.04.031.

Jamali S, Malektaji S. 2014. Improving grouping genetic algorithm for virtual machine placement in cloud data centers. In: *2014 4th international conference on computer and knowledge engineering (ICCKE)*. Piscataway: IEEE, 328–333.

Javaid N, Javaid S, Abdul W, Ahmed I, Almogren A, Alamri A, Niaz IA. 2017. A hybrid genetic wind driven heuristic optimization algorithm for demand side management in smart grid. *Energies* **10(3)**:319 DOI 10.3390/en10030319.

Jensi R, Jiji GW. 2016. An enhanced particle swarm optimization with levy flight for global optimization. *Applied Soft Computing* **43**:248–261 DOI 10.1016/j.asoc.2016.02.018.

Jin C, Bai X, Yang C, Mao W, Xu X. 2020. A review of power consumption models of servers in data centers. *Applied Energy* **265**:114806 DOI 10.1016/j.apenergy.2020.114806.

Keller G, Tighe M, Lutfiyya H, Bauer M. 2012. An analysis of first fit heuristics for the virtual machine relocation problem. In: *2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm)*. Piscataway: IEEE, 406–413.

Kumar D, Mandal T. 2017. Bi-objective virtual machine placement using hybrid of genetic algorithm and particle swarm optimization in cloud data center. *International Journal of Applied Engineering Research* **12(22)**:12044–12051.

Kumaraswamy S, Nair MK. 2019. Bin packing algorithms for virtual machine placement in cloud computing: a review. *International Journal of Electrical and Computer Engineering* **9(1)**:512.

López J, Kushik N, Zeghlache D. 2019. Virtual machine placement quality estimation in cloud infrastructures using integer linear programming. *Software Quality Journal* **27(2)**:731–755 DOI 10.1007/s11219-018-9420-z.

Mann ZÁ. 2015. Approximability of virtual machine allocation: much harder than bin packing. *Available at http://www.cs.bme.hu/~manusz/publications/Japan-2015/Mann_Japan_2015.pdf* .

Mann ZÁ. 2016. Multicore-aware virtual machine placement in cloud data centers. *IEEE Transactions on Computers* **65(11)**:3357–3369 DOI 10.1109/TC.2016.2529629.

Masdari M, Gharehpasha S, Ghobaei-Arani M, Ghasemi V. 2019. Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions. In: *Cluster Computing*. Cambridge: Academic Press, 1–31.

Masdari M, Nabavi SS, Ahmadi V. 2016. An overview of virtual machine placement schemes in cloud computing. *Journal of Network and Computer Applications* **66**:106–127 DOI 10.1016/j.jnca.2016.01.011.

Mollamotalebi M, Hajireza S. 2017. Multi-objective dynamic management of virtual machines in cloud environments. *journal of Cloud Computing* **6(1)**:1–13 DOI 10.1504/IJCC.2017.083901.

Nabil E. 2016. A modified flower pollination algorithm for global optimization. *Expert Systems with Applications* **57**:192–203 DOI 10.1016/j.eswa.2016.03.047.

Nakisa B, Ahmad Nazri MZ, Rastgoo MN, Abdullah S. 2014. A survey: Particle swarm optimization based algorithms to solve premature convergence problem. *Journal of Computer Science* **10(9)**:1758–1765 DOI 10.3844/jcssp.2014.1758.1765.

Pires FL, Barán B. 2015. A virtual machine placement taxonomy. In: *2015 15th IEEE/ACM international symposium on cluster, cloud and grid computing*. Piscataway: IEEE, 159–168.

Qingxi W, Xiaobo G. 2016. Particle swarm optimization algorithm based on Levy flight. *Application Research of Computers* **33(9)**:2588–2591.

Rashid A, Chaturvedi A. 2019. Virtualization and its role in Cloud Computing environment. *International Journal of Computer Sciences and Engineering* **7(4)**:1131–1136 DOI 10.26438/ijcse/v7i4.11311136.

Regaieg R, Koubàa M, Osei-Opoku E, Aguili T. 2018. Multi-objective mixed integer linear programming model for vm placement to minimize resource wastage in a heterogeneous cloud provider data center. In: *2018 tenth international conference on ubiquitous and future networks (ICUFN)*. Piscataway: IEEE, 401–406.

Ribas BC, Suguimoto RM, Montano RA, Silva F, Castilho M. 2013. Pbfvmc: A new pseudo-boolean formulation to virtual-machine consolidation. In: *2013 Brazilian conference on intelligent systems*. Piscataway: IEEE, 201–206.

Saeedi P, Shirvani MH. 2021. An improved thermodynamic simulated annealing-based approach for resource-skewness-aware and power-efficient virtual machine consolidation in cloud datacenters. *Soft Computing* **25(7)**:5233–5260 DOI 10.1007/s00500-020-05523-1.

Satpathy A, Addya SK, Turuk AK, Majhi B, Sahoo G. 2018. Crow search based virtual machine placement strategy in cloud data centers with live migration. *Computers & Electrical Engineering* **69**:334–350 DOI 10.1016/j.compeleceng.2017.12.032.

Schmitt M, Wanka R. 2015. Particle swarm optimization almost surely finds local optima. *Theoretical Computer Science* **561**:57–72 DOI 10.1016/j.tcs.2014.05.017.

Shirvani MH. 2020. A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems. *Engineering Applications of Artificial Intelligence* **90**:103501 DOI 10.1016/j.engappai.2020.103501.

Silva Filho MC, Monteiro CC, Inácio PR, Freire MM. 2018. Approaches for optimizing virtual machine placement and migration in cloud environments: a survey. *Journal of Parallel and Distributed Computing* **111**:222–250 DOI 10.1016/j.jpdc.2017.08.010.

Singh M. 2018. Virtualization in cloud computing-a study. In: *2018 international conference on advances in computing, communication control and networking (ICACCCN)*. Piscataway: IEEE, 64–67.

Usman MJ, Ismail AS, Chizari H, Abdul-Salaam G, Usman AM, Gital AY, Kaiwartya O, Aliyu A. 2019. Energy-efficient virtual machine allocation technique using flower pollination algorithm in cloud datacenter: a panacea to green computing. *Journal of Bionic Engineering* **16(2)**:354–366 DOI 10.1007/s42235-019-0030-7.

Usman MJ, Ismail AS, Gital AY, Aliyu A. 2018. Energy-aware distributed multi-cloud flower pollination optimization scheme. In: *2018 Seventh ICT international student project conference (ICT-ISPC)*. Piscataway: IEEE, 1–5.

Vakilinia S. 2018. Energy efficient temporal load aware resource allocation in cloud computing datacenters. *Journal of Cloud Computing* **7(1)**:1–24 DOI 10.1186/s13677-017-0102-3.

Van HN, Tran FD, Menaud J.-M. 2010. Performance and power management for cloud infrastructures. In: *2010 IEEE 3rd international conference on cloud computing*. Piscataway: IEEE, 329–336.

Varasteh A, Goudarzi M. 2015. Server consolidation techniques in virtualized data centers: a survey. *IEEE Systems Journal* **11(2)**:772–783.

White JA. 2012. *Analysis of queueing systems*. Amsterdam: Elsevier.

Yang X.-S. 2012. Flower pollination algorithm for global optimization. In: *International conference on unconventional computing and natural computation*. Springer, 240–249.

Yang X-S, Karamanoglu M, He X. 2014. Flower pollination algorithm: a novel approach for multiobjective optimization. *Engineering optimization* **46(9)**:1222–1237 DOI 10.1080/0305215X.2013.832237.

Zhao D-M, Zhou J-T, Li K. 2019. An energy-aware algorithm for virtual machine placement in cloud computing. *IEEE Access* **7**:55659–55668 DOI 10.1109/ACCESS.2019.2913175.