electronics

MDPI

Article

Optimal Virtual Machine Placement Based on Grey Wolf Optimization

Ammar Al-Moalmi [1], Juan Luo [1,*], Ahmad Salah [1,2,3] and Kenli Li [1,3]

[1] College of Computer Science and Electrical Engineering, Hunan University, Changsha 410082, China; al_moalmi@hnu.edu.cn (A.A.-M.); ahmad@hnu.edu.cn (A.S.); lkl@hnu.edu.cn (K.L.)
[2] College of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt
[3] The National Supercomputing Center in Changsha, Changsha 410082, China
* Correspondence: juanluo@hnu.edu.cn

Received: 4 February 2019; Accepted: 28 February 2019; Published: 4 March 2019

Abstract: Virtual machine placement (VMP) optimization is a crucial task in the field of cloud computing. VMP optimization has a substantial impact on the energy efficiency of data centers, as it reduces the number of active physical servers, thereby reducing the power consumption. In this paper, a computational intelligence technique is applied to address the problem of VMP optimization. The problem is formulated as a minimization problem in which the objective is to reduce the number of active hosts and the power consumption. Based on the promising performance of the grey wolf optimization (GWO) technique for combinatorial problems, GWO-VMP is proposed. We propose transforming the VMP optimization problem into binary and discrete problems via two algorithms. The proposed method effectively minimizes the number of active servers that are used to host the virtual machines (VMs). We evaluated the proposed method on various VM sizes in the CloudSIM environment of homogeneous and heterogeneous servers. The experimental results demonstrate the efficiency of the proposed method in reducing energy consumption and the more efficient use of CPU and memory resources.

Keywords: virtual machine placement; cloud computing; grey wolf optimization

1. Introduction

Cloud computing has transformed traditional IT into a promising paradigm in which the cloud is used as a utility [1,2]. Service-on-demand is a common cloud computing service model, in which the user can dynamically scale up or down the reserved resources and pay for the exact resource usage. Cloud computing provides its services via three models: Software as a Service (SaaS) for applications, Infrastructure as a Service (IaaS) for hardware resources, and Platform as a Service (PaaS) for runtime environments [3].

In IaaS, cloud computing offers an unlimited amount of heterogeneous resources with high elasticity of use via a virtualization technique [4]. A virtual machine (VM) is created to host an application according to the customer requirements for resources such as CPU, memory, storage, and bandwidth [5,6]. The virtualization technique enables multiple VMs to share the physical resources on the same physical machine (PM). This technique facilitates the efficient exploitation of the physical resources via VM consolidation, which places as many VMs as possible on the minimal number of PMs [7].

The rapid growth of cloud computing service demands has increased the power consumption of cloud data centers, where the power consumption and carbon dioxide emission are the largest challenges and hamper the promotion of cloud computing [8]. Power savings and emissions reduction can be effectively realized by minimizing the number of active hosts and shutting idle servers

down [9,10]. Therefore, reducing the energy consumption of servers is vital to decreasing the total power of a data center [11].

Virtual machine placement (VMP) optimization is a process of selecting the minimal number of PMs that can supply the required resources for hosting a specified number of VMs with the lowest possible power consumption. VMP optimization increases the energy efficiency and resource utilization of cloud data centers by introducing a solution in which VMs are hosted in the minimal number of active PMs. Moreover, VMP optimization can prolong the stability of the datacenter before the reallocation of VMs becomes an urgent issue [12,13].

VMP optimization is an NP-hard combinatorial problem. The problem can be addressed with diverse, conflicting objectives [14,15]. The VMP problem has been solved for several objectives, i.e., a linear programming problem (LP) is used to minimize the cost of hosting VMs in PMs [16] and heuristic data to consolidate VMs on a minimal number of PMs [17].

Evolutionary computation algorithms have been used to reduce the power consumption and increase the resource utilization, i.e., genetic algorithm (GA). In [18], an improved genetic algorithm has been introduced for maximizing the multidimensional resource usage and minimizing the communication traffic. In [19,20], the proposed method used multicapacity bin packing to find the optimal assignment for the VMP problem. In [21], a method for optimizing a neural network that forecasts the power consumption via GA was proposed.

Grey wolf optimization (GWO) is an evolutionary algorithm. GWO yields promising results compared to the well-known heuristics, such as evolution strategy (ES), the gravitational search algorithm (GSA), differential evolution (DE), evolutionary programming (EP), and particle swarm optimization (PSO) [22]. Recently, the binary grey wolf optimization (BGWO) approach for feature selection was proposed in [23].

In this paper, we develop a GWO-based method for addressing the VMP optimization problem as a combinatorial problem. We formulate the VMP task as binary and discrete problems. Then, we adapt the GWO method for each of these two problems. The performance of the proposed methods is evaluated via a set of experiments on homogeneous and heterogeneous data center environments. The major contributions of this paper are as follows:

1. To the best of the authors' knowledge, this is the first time that GWO has been utilized to address the problem of optimal VM placement; we refer to this method as GWO-VMP. The proposed method reduces the energy consumption of cloud computing by allocating VMs into the minimal number of active PMs.
2. The proposed work formulated the VMP optimization problem as discrete and binary GWO problems. The binary approach is more efficient.
3. We proposed a method for correcting infeasible solutions (RIS) to accelerate the convergence of the proposed algorithms.
4. We performed an extensive experimental study to evaluate the effectiveness and efficiency of the proposed algorithms. The proposed methods performed competitively compared to the state-of-the-art methods.

2. Background and Related Work

2.1. VMP Problem Formulation

Virtualization is the key technology that powers cloud computing. The largest benefit of virtualization is server consolidation, where resources of a single server can be split for multiple VMs. Virtualization reduces the operating cost and increases the utilization efficiency of the cloud data center. Application that are demanded by customers are hosted on VMs according to the customer requirements (i.e., operating system and hardware specifications).

Then, the VMP strategy assigns the VMs to a sufficient number of physical servers according to various objectives [24,25]. One of the most important objectives is power consumption reduction

because of its impacts on the operating cost and environmental effects. In this paper, we introduce a new VMP strategy for minimizing the number of active PMs and reducing the power consumption.

Consider a cloud data center that contains n PMs and m VMs. P represents a set of PMs, where P_i represents the i^{th} PM, i belongs to $[1 \ldots n]$, and $P_i \in P$. Similarly, V represents a set of VMs, where V_j represents the j^{th} VM, j belongs to $[1 \ldots m]$, and $V_j \in V$.

In this study, we focus on CPU and memory resources. The required computational power and memory of V_j are represented as $Vcpu_j$ and $Vram_j$, respectively. Likewise, the capacities of P_i for CPU and memory are represented as $Pcpu_i$ and $Pram_i$, respectively.

We suppose that each PM has sufficient capacity to host any single VM. Thus, a single VM can be hosted on one and only one PM. The placement solution, which is denoted by S, is represented by a zero-one adjacency matrix, where $x_{ij} = 1$ if VM_j is assigned to P_i and $x_{ij} = 0$ otherwise. Similar to the constraints that were proposed in [26], the optimal VMP, which has the minimal number of active PMs in a cloud data center, can be formulated as:

$$min \sum_{i=1}^{n} y_i \tag{1}$$

which is subject to the following constraints:

$$x_{ij} = \begin{cases} 1, & if\ V_j\ is\ assigned\ to\ P_i,\ \forall i \in P\ and\ \forall j \in V \\ 0, & otherwise \end{cases} \tag{2}$$

$$y_i = \begin{cases} 1, & if \sum_{j=1}^{N} x_{ij} \geqslant 1\ \ \forall i \in P \\ 0, & otherwise \end{cases} \tag{3}$$

$$\sum_{i=1}^{N} x_{ij} = 1 \quad \forall j \in V \tag{4}$$

$$\sum_{i=1}^{M} Vcpu_j.x_{ij} \leqslant Pcpu_i.y_i \ \ \forall i \in P \tag{5}$$

$$\sum_{i=1}^{M} Vram_j.x_{ij} \leqslant Pram_i.y_i \ \ \forall i \in P \tag{6}$$

According to Equation (3), P_i is active if $y_i = 1$ and P_i is idle if $y_i = 0$. Equation (4) ensures that V_j is submitted to only one of the PMs. Equations (5) and (6) specify the P_i capacity constraints, which the CPU and memory should not exceed.

The power that is consumed by an active PM without a load is approximately 50% to 70% of the power consumption of a fully utilized PM [27]. The power consumption has a linear relationship with the CPU load, as demonstrated in [28]. Consequently, shutting down inactive PMs is vital for minimizing the total power consumption in the cloud data center. Therefore, we defined the power consumption as a linear function of the CPU utilization as follows:

$$PC\left(Pcpu_i\right) = \left\{ \begin{array}{cc} P_{idle} + \left(P_{full} - P_{idle}\right) & \\ \times\ Pcpu_i, & if\ Pcpu_i > 0 \\ 0, & otherwise \end{array} \right\} \tag{7}$$

where P_{idle} is the power that is consumed in an idle state, P_{full} is the power that is consumed in a fully utilized CPU, and $Pcpu_i \in [0, 1]$. In this paper, we have assumed that the power consumption in an idle state of the PM is 60% of the energy that is consumed in a fully utilized state. An effective way to reduce power consumption is to increase the resource utilization of active hosts and decrease the number of active PMs. Therefore, full CPU utilization will lead to consuming the total power of active

PMs regardless of other types of resource. Consequently, we have considered the CPU resource in a power model as it is the most important resource reducing the power consumption [29–31].

2.2. Related Work

VMP optimization is an NP-hard problem because of the diverse, conflicting objectives [14]. The VMP problem has been solved for various objectives in different ways. The stochastic integer problem, which is a linear programming problem (LP), is solved via a mathematical optimization technique, which is used to minimize the cost of hosting VMs in PMs [16]. In [32], the resource demand, which is estimated as a correlation-aware value and an aggregate of essential demands, is used to assign VMs to PMs, where the probability of the server load exceeding its capacity is p.

In [17], the authors proposed minimizing the server cost by VM using integer LP model using a heuristic data to consolidate VMs on a minimum number of PMs. In [33], the power consumption has been decreased by non-bypass IP/wavelength division multiplexing core network model. They used the DEER-CD model with comparable power efficiency to develop energy efficiency. The concept is to place the small VMs in a proximity to their users.

The meta-heuristic approach is another method which effectively solves the VMP problem. The main difference, compared to the heuristic algorithm, is that meta-heuristic algorithms are designed for a general purpose problem and can efficiently avoid local optima [34]. In [26], X.F. Liu et al. proposed an ACS-based approach, OEMACS, to assign m VMs to n PMs. The VMs were grouped depending on historical experience of packing the VMs together before using artificial ants to search for fittest place for hosting. The infeasible solution of the update was revised every search iteration to reduce the convergence time. The solution space was reduced while the iteration number grows. The solution is revised to turn infeasible solution to feasible one by ordering exchange process and migrate VMs on overloaded PMs which reduces the convergence time. In [35], Kansal and Chana proposed an energy-aware model based on artificial bee colony algorithm to schedule jobs to the minimum resources in a cloud environment. This model minimized the energy consumption and execution time of applications.

In [19], a GA-based method, namely, RGGA, for addressing the problem of VMP, is proposed. The authors proposed using multicapacity bin packing to identify the optimal assignment for the VMP problem. RGGA produces solutions via a crossover process in which the PMs of the previous generation are sorted according to resource utilization and PMs that have heavy loads are selected to host VMs. The unassigned VMs are arranged in decreasing order of CPU and RAM and submitted to new PMs.

In [36], the authors proposed a novel method based on Simulated Annealing (SA) for addressing the problem of VMP. The method includes a proposed searching technique for finding better SA configurations. The acceptance criteria of the new configurations include two conditions. The new configuration, to be accepted, must be feasible and have a lower energy consumption than the previous state. In addition, a temperature scheduling technique is discussed for the purpose of avoiding searching far from the optimal solution.

Cho et al. proposed a hybrid meta-heuristic algorithm based on a combination of ant colony optimization and particle swarm optimization [37]. The algorithm schedules VMs to PMs according to the load prediction for the new demand and rejects the unsatisfied demand to reduce the computing time of the scheduling. Wen et al. proposed a meta-heuristic algorithm (ACO-VMM) for migrating VMs to PMs that aims at finding a near-optimal solution [38]. The monitoring data of resource utilization and traversal strategies are used by ants to identify the mapping between VMs and PMs that has the minimal number of VM migrations. In [39], Feller et al. utilized ant colony optimization to minimize the number of active PMs in a cloud data center by consolidating VMs in the minimal number of PMs based on CPU utilization. However, this method considers only a single resource.

In [40], Tawfeek et al. addressed VM consolidation for the only one-dimensional resource which gives a better result than FFD. As well, Suseela and Jeyakrishnan used a new hyper version of the

ant colony and particle swarm to consolidate the VMs for minimum power consumption without the direct aim of reducing the number of active PMs, which is reported to have good results [41].

In addition, several works that are based on meta-heuristic algorithms utilize particle swarm optimization (PSO), which is applied for the VMP problem in [42]. The authors adapted the PSO method to the VMP problem with the objective of realizing the low power consumption. In PSO for VMP, the problem of submitting VMs to PMs is represented as a matrix, namely, $[m; n]$, where n is the number of VMs and m is the number of PMs. The particles and the velocities of the particles in the initial solution are randomly distributed. The solutions are evaluated according to the VMP constraints. The best local and global results are obtained based on the fitness function, which aims at minimizing the power consumption. In the beginning, the position of each particle is set to its local best position. Then, the best global result corresponds to the particle that has the minimum overall power consumption. In each iteration, each particle updates its position according to its velocity. If the corresponding bit in the velocity matrix is equal to one, the binary bit in the particle matrix is revised. Then, the fitness function determines whether the particle updates its position to the new value or saves the old position as the local best position. At the end of the iteration, the global best solution is selected as the best solution for the VMP problem. Similarly, Braiki et al. proposed a multiobjective PSO algorithm that seeks to maximize the packing efficiency while minimizing the energy consumption [43].

2.3. Grey Wolf Optimization

Grey wolves live in packs and have a hierarchical governing system that imposes very strict rules. According to [22], the wolves in a pack are categorized into four levels. The first level contains the alphas, who are males or females who make decisions and lead the pack. The wolves in the remaining levels of the wolf pack should obey the alphas' instructions. The second level contains the betas, who are at a lower level than the alphas and work as consultants and help make decisions. The betas are the best candidates to be alphas. The third level contains deltas, who work as elders, hunters, sentinels, scouts, and caretakers. They submit information to alphas and betas and dominate the omegas. Elders are experienced wolves who are candidates to be alphas or betas.

Hunters are responsible for helping alphas and betas hunt prey and provision food for the pack. Sentinels are responsible for guarding and guaranteeing the security of the pack. Scouts are responsible for monitoring the boundaries and alerting the pack of any danger. Finally, caretakers care for the weak, ill, and injured wolves in the pack. The fourth level contains the omegas, who must submit to all the dominant levels in the pack and serve as scapegoats. They are allowed to eat only after all the other wolves have finished eating.

The hunting of the grey wolves is guided by alpha (α), beta (β), and delta (δ) wolves. Consequently, the best solutions are produced by alpha (α) wolves, followed by beta (β) and delta (δ) wolves. The remaining solutions correspond to omega (ω) wolves. The hunting in GWO consists of two stages: encircling and attacking. The update procedure is mathematically formulated as follows:

$$\vec{X}(t+1) = \vec{X}_p(t) + \vec{A} \,.\, \vec{D}, \tag{8}$$

where $\vec{D}$ is defined in Equation (9), t is the number of iterations, $\vec{A}$ is the coefficient vector, $\vec{X}_p$ is the position of the prey, and $\vec{X}$ is the position of the grey wolf.

$$\vec{D} = |\vec{C} \,.\, \vec{X}_p(t) - \vec{X}(t)|, \tag{9}$$

where $\vec{C}$ is coefficient vector. The $\vec{A}$, $\vec{C}$ vectors are calculated as following:

$$\vec{A} = 2a.\vec{r_1} - a \tag{10}$$

$$\vec{C} = 2\vec{r_2}, \tag{11}$$

where a is linearly decreased over the course of the iteration from 2 to 0 and r_1 and r_2 are randomly generated in [0, 1]. The alphas normally guide the grey wolves' hunting; however, the betas and deltas might also occasionally participate in hunting.

2.4. Binary Grey Wolf

Binary grey wolf optimization (BGWO) has been proposed in a recent study [23]. Unlike continuous grey wolf optimization (CGWO), where the wolves update their positions to any point in the space, BGWO solutions are limited to binary $\{0, 1\}$ values. At any time, each solution is represented in binary form and located on the corner of a hypercube. According to the BGWO algorithm, the wolves follow the same approach to updating their positions while restricting the solutions to binary values. In GWO, all wolves estimate the positions of prey and the positions of the best wolves relative to the prey guide the other wolves to update their positions randomly around those of the best wolves toward the prey.

3. GWO for Virtual Machine Placement

3.1. Adjusting GWO for VMP

The energy consumption of a cloud data center can be reduced effectively by minimizing the number of active PMs. Therefore, we adjust the GWO algorithm to fit the VMP problem to obtain a solution that maps the VMs to a minimal number of active PMs. We choose the best feasible solution S, which has a minimal number of active PMs. However, since the optimal mapping of VMs to PMs is unknown at the initial state, we start with the best solution that has been generated, which is denoted as S_B, in the initial state. The m VMs are randomly distributed on n PMs, where each VM is submitted to a single PM. Consequently, there are m^n possible distributions of m over n.

The wolves continuously update their positions to search for prey (an optimal solution). During the search process, to encircle the prey, the wolves are guided by the best solutions, which are denoted as α, β, and δ, and update their positions according to the α, β, and δ wolves' positions. Therefore, the VMP solution can be constructed as shown in Figure 1.

Figure 1 presents an example of four feasible solutions, each in a different color, for hosting five VMs, where $n = 4$ PMs and $m = 5$ VMs. The solution that is shown as a black line represents the initial solution for four PMs. The solution that is shown as a yellow line represents another enhanced solution for submitting five VMs to three PMs. The solution that is shown as a red line produces the same number of active PMs as the solution that is shown as a yellow line, but in a different way. The optimum solution, which is represented a blue line, uses only two PMs to host the five VMs.

3.2. Solution Construction

After the randomly distributed solution step has been completed for all wolves in the pack, GWO-VMP generates new solutions by updating the existing solutions for every wolf to search for an optimum distribution of VMs on PMs. In each iteration t, the wolves update their locations according to the best three solutions that have been obtained so far. The best solutions are the solutions that utilize the minimal number of active PMs. The best solutions (α, β, and δ) maintain their locations without any update to guide other wolves to the optimum solution.

Figure 1 shows four wolves' solutions for assigning five VMs to the minimal number of the existing five PMs, where $n = 4$ and $m = 5$. The four wolves' solutions can be represented by the following set: $S = \{S_\alpha, S_\beta, S_\delta, S_\omega\}$. These solutions are sorted in ascending order of the number of active PMs. The first-, second-, and third-best solutions are denoted as S_α, S_β, and S_δ, respectively. Each of these solutions represents a single solution. In Figure 1, these solutions are $(x_{31}, x_{12}, x_{33}, x_{14}, x_{15})$, $(x_{21}, x_{12}, x_{13}, x_{34}, x_{25})$, and $(x_{11}, x_{22}, x_{13}, x_{14}, x_{35})$, respectively. The remaining solution(s) are denoted as S_ω, $(x_{11}, x_{22}, x_{33}, x_{44}, x_{35})$. S_ω represents all solutions except the first three best solutions. Thus, it represents a set of solutions. Consequently, the number of wolves that are involved in hunting

should exceed three. If two wolves introduce the same number of active PMs, in a heterogeneous environment, the algorithm sorts them according to the power consumption value, where the lower the power consumption is, the better the solution. At the last iteration, solution α is reported as the best solution, which represents the best placement of the VMs on the PMs that was obtained via GWO.

To reduce the number of active PMs, each PM must host as many VMs as possible, which increases the resource utilization of each PM. Consequently, updating the location of the set of VMs to the same PM is necessary for minimizing the number of active PMs. In each iteration, the VM location is updated to any available PM: i $(1 \leq i \leq n)$. The available PM, namely, P_i, is defined as

$$P_i = \left\{ \begin{array}{l} \sum_{j=1}^{m} x_{ij}.Vcpu_j + Vcpu_l \leq Pcpu_i \; and \\ \\ \sum_{j=1}^{m} x_{ij}.Vram_j + Vram_l \leq Pram_i \end{array} \right\} \tag{12}$$

where $Vcpu_j$ and $Vram_j$ are the sums of the CPU and memory capacities of the already submitted VMs on P_i, respectively. $Vcpu_l$ and $Vram_l$ are the CPU and memory capacities of the unscheduled VM. $Pcpu_i$ and $Pram_i$ are the capacities of the PM. This equation represents the constraints on the capacities, which facilitate the selection of a suitable PM from the set of PMs. The methods that are used to update the discrete and binary locations of VMs on PMs will be discussed in detail in the following sections (E, F).

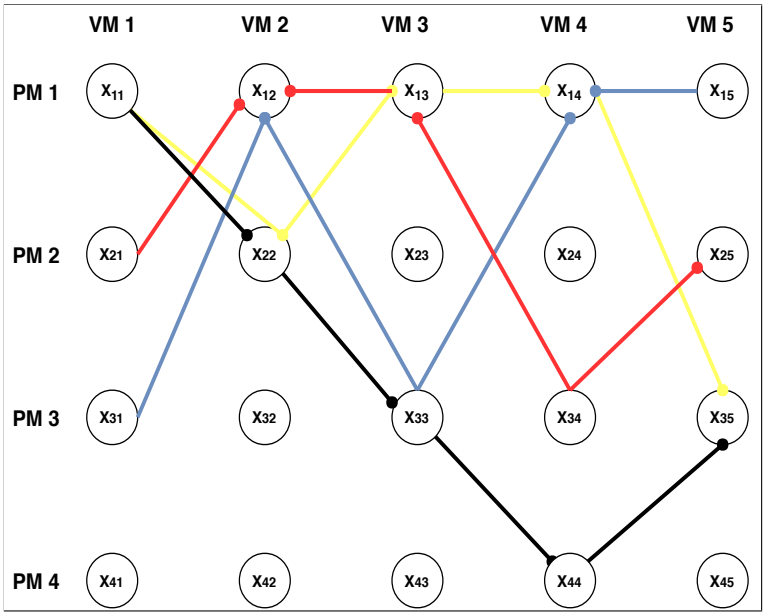


Figure 1. Example of feasible solutions.

3.3. Revising the Infeasible Solutions (RIS)

After all wolves have updated their positions, not all solutions are feasible according to the constraints in Equations (2)–(6). Each infeasible solution must be transformed into a feasible solution before α, β, and δ are chosen. We propose an approach, namely, RIS, for correcting the updated positions according to the constraints of Equations (2)–(6). RIS consists of three procedures: (1) eliminating the duplicate assignments, (2) obviating the overload assignments, and (3) reassigning the unallocated VMs. These procedures correct VMs' positions and obviate the overloaded PMs. If all solutions become feasible solutions, the α, β, and δ can be chosen.

3.3.1. Eliminating the Duplicate Assignments

This operation is used to determine whether a solution has a single VM assigned to more than one PM and to change the solution so that each VM is hosted by a single PM according to Equation (4).

There are several approaches to eliminating the problem of a VM being hosted by more than one PM. One possible approach is to fix a random number i and set $x_{ij} = 1$ and the values of the other entries to zero. Another approach is to keep the first PM that is hosting the VM and set the VM as not assigned to the other PMs.

3.3.2. Obviating the Overload Assignments

This operation is used to determine whether the load of a PM satisfies the constraints in Equations (5) and (6). For each overloaded PM i, we decrease the load by moving some of the VMs that are assigned to PM i to another PM. Several approaches are available for selecting which VM to remove from the overloaded PM.

To maximize the resource utilization and minimize the number of active PMs, we can select the VM that has the lowest resource requirements among the VMs that are hosted by the overloaded PM. This would obviate the overload and maximize the resource utilization. After the VM has been selected, it is reassigned to another PM. Another way of selecting the VM is by sorting the VMs that are hosted by the overloaded PM and selecting the worst-balanced resource utilization according to the absolute difference between the CPU and RAM requirements of these VMs.

In addition, we can select the first VM that violates the constraints in Equations (5) and (6) during the evaluation process. This approach would decrease the computational burden and obviate the overload. However, the resource utilization might not be balanced or maximized because the size of the eliminated VM is not considered. After selecting a VM to reassign to avoid an overload, if PM i is still overloaded, then another VM is selected for reassignment. Once a VM j that is hosted by an overloaded PM i is selected for reassignment, the variable x_{ij} is set to zero.

3.3.3. Reassigning the Unallocated VMs

This operation is used to reassign a VM that was not allocated during updating, duplication removal, or overload obviation. An infeasible solution might contain an unallocated VM, which should be reassigned. After the wolves' positions have been updated, the VM constraint in Equation (4) is evaluated. If the sum of the value $x_{ij} = 0$ that corresponds to VM j was not assigned to any PM during updating, duplication removal, or overload obviation, we must reassign the VM to a nonoverloaded PM that has sufficient residual resources to meet the resource requirements of this VM.

There are many approaches to implementing this procedure. The unhosted VM can be reassigned to a random PM under the constraints of Equations (5) and (6), or we can assign the unhosted VM to a PM that has sufficient resources via the first fit (FF) greedy algorithm. To maximize the resource utilization and accelerate the convergence of the solution, we prefer to reassign the missing VM via the best fit (BF) greedy algorithm. The best PM is the PM that has a minimum sufficient residual capacity that satisfies the resource requirements of this VM.

To maximize the resource utilization and balance the use of resources (CPU and memory), we can reassign the VM to the PM that will have the minimum estimated absolute difference between CPU and memory resource utilizations after the VM has been added to the PM, where the absolute difference between CPU and memory utilizations would be calculated after adding the VM requirements to the utilized PM resources. Then, the minimum absolute difference between the resources would be selected.

3.4. Objective Function

After the wolves have updated their positions and RIS has transformed the infeasible solutions to feasible solutions, the proposed method evaluates the fitness of the solutions to determine the α, β, and δ solutions. k wolves represent k solutions; only three wolves are selected as α, β, and δ and keep their positions, or solutions, without any update in the next iteration to serve as references to other wolves through the updating process. To select the best three solutions, there are two objectives, which are expressed as follows:

$$f_1(S) = \begin{cases} \sum_{i=1}^{n} y_i, \forall i \in P,\ y_i\ satisfies \\ the\ constraints \quad in\ Equations\ (5)\ and\ (6) \\ \\ n+1, \quad otherwise \end{cases} \tag{13}$$

$$f_2(S) = \sum_{i=1}^{n} PC(Pcpu_i) \times y_i. \tag{14}$$

where n is the number of PMs that are available in the data center to host m VMs and y_i indicates whether P_i is used in this solution S or not.

Equation (13) calculates the number of active PMs that are used in the feasible solution. If an infeasible solution remains after the RIS process, we distinguish it by assigning the value $n+1$. Consequently, the infeasible solutions would be at the end of the results of this function once they have been sorted in ascending order. The first three values would be considered as α, β, and δ. The other solutions would be considered as ω; thus, they must update their solutions in the next iteration.

The infeasible solutions must also update their solutions. RIS cannot transform an infeasible solution to a feasible solution because it utilizes only active PMs that are available in this solution at the current iteration. If two solutions have the same number of active PMs, we compare their f_2 values, which are calculated via Equation (14), and select the one that has the smaller value. Therefore, the three solutions with the fewest active PMs and the lowest power consumption are selected as the best solutions to be α, β, and δ. A flowchart of general GWO-VMP is shown in Figure 2 and discussed in the following section.

3.5. BGWO-VMP Algorithm

In the BGWO-VMP, the solution in which the VMs are submitted to the minimal number of active PMs is presented as a matrix of binary values. In BGWO for the VMP problem, every wolf in the pack represents a solution in which m VMs are mapped to n PMs as a matrix $[n; m]$, where m is the number of VMs and n is the number of PMs. Therefore, a wolf's position is represented as follows:

$$W_k = \begin{bmatrix} x_k^{11}, x_k^{12}, \ldots, & x_k^{1m} \\ x_k^{21}, x_k^{22}, \ldots, & x_k^{1m} \\ \ldots\ldots & \\ x_k^{n1}, x_k^{n2}, \ldots, & x_k^{nm} \end{bmatrix} \tag{15}$$

where W_k is the matrix position of wolf k in the pack. The corresponding bit x_k^{ij} equals one if $V_j \in V$ is assigned to $P_i \in P$; otherwise, the bit value equals zero.

The solution matrix of $[n; m]$ for every wolf is updated according to the solution matrices of α, β, and δ. The algorithm starts updating the solution matrix for each column, where each column contains the submitted value of V_j to P_i under the constraints in Equation (4). Every column in the solution matrix is updated according to every column in the α, β, and δ solutions. The allocation of V_j is updated bit by bit in each iteration.

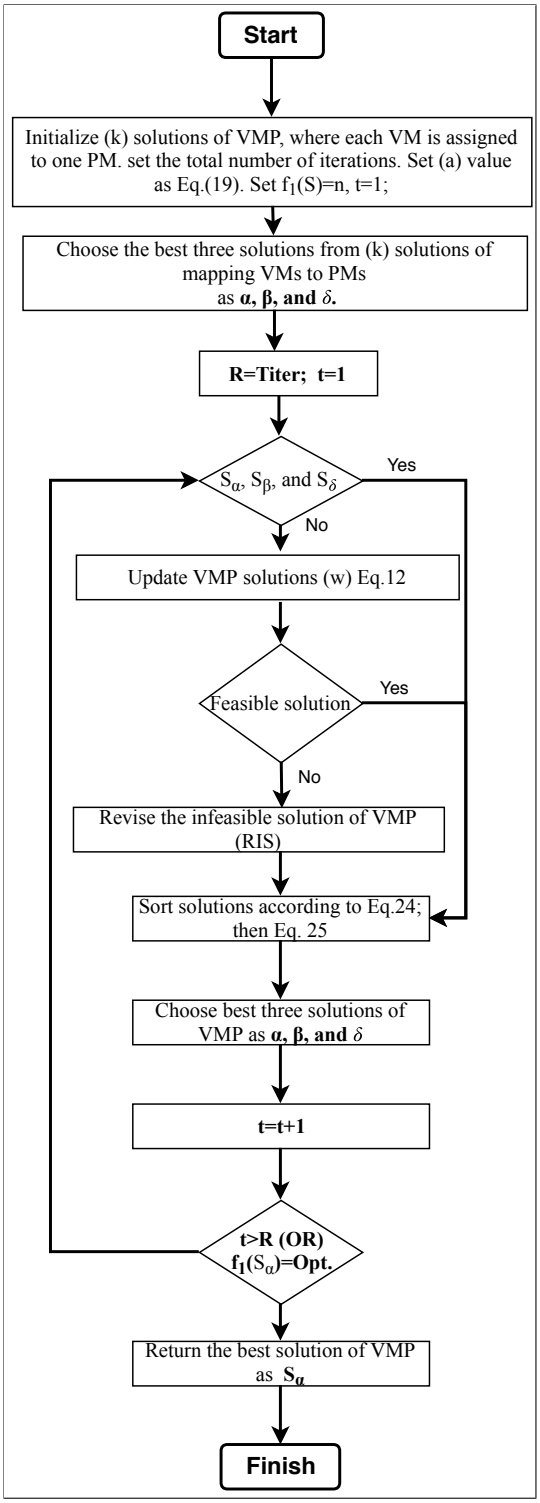


Figure 2. Flowchart of GWO-VMP.

Alpha (α) represents the best solution, beta (β) represents the second-best solution, and delta (δ) represents the third-best solution, where it is assumed that they have substantial amounts of information about the possible location of prey. Consequently, the first three best solutions that have been obtained so far are considered as (α), (β), and (δ) among all search agents, including all grey wolf levels (α, β, δ, and ω). Therefore, all other wolves update their positions according to the best search

agents (α, β, and δ). Based on [23], the BGWO algorithm only forces the updated grey wolf position vector to be a binary vector and the main updating formula is calculated as follows:

$$x^{ij}(t+1) = \begin{cases} 1, & if\ sigmoid(\frac{x_1^{ij}+x_2^{ij}+x_3^{ij}}{3}) \geqslant random \\ 0 & otherwise \end{cases} \tag{16}$$

where $x^{ij}(t+1)$ is the updated binary position at iteration t; $x_1^{ij}, x_2^{ij}, x_3^{ij}$ are calculated according to Equations (18)–(20); $random$ is a random number $\in [0,1]$; and function $sigmoid(a)$ is formulated as follows:

$$sigmoid(a) = \frac{1}{1+e^{-10(x-0.5)}} \tag{17}$$

$$x_1^{ij} = |x_\alpha^{ij} - \vec{A}_1 \,.\, \vec{D}_\alpha|, \tag{18}$$

$$x_2^{ij} = |x_\beta^{ij} - \vec{A}_2 \,.\, \vec{D}_\beta|, \tag{19}$$

$$x_3^{ij} = |x_\delta^{ij} - \vec{A}_3 \,.\, \vec{D}_\delta|, \tag{20}$$

where $x_\alpha^{ij}, x_\beta^{ij}, x_\delta^{ij}$ are the corresponding bits of the best three solutions that are obtained in the pack in every iteration t; $\vec{A}_1, \vec{A}_2, \vec{A}_3$ are calculated via Equation (10); and $\vec{D}_\alpha, \vec{D}_\beta, \vec{D}_\delta$ are calculated as follows via Equations (21)–(23), respectively.

$$\vec{D}_\alpha = |\vec{C}_1 \,.\, x_\alpha^{ij} \ - \ x^{ij}|, \tag{21}$$

$$\vec{D}_\beta = |\vec{C}_2 \,.\, x_\beta^{ij} \ - \ x^{ij}|, \tag{22}$$

$$\vec{D}_\delta = |\vec{C}_3 \,.\, x_\delta^{ij} \ - \ x^{ij}|, \tag{23}$$

where $\vec{C}_1, \vec{C}_2, \vec{C}_3$ are calculated via Equation (11) and x^{ij} represents the current corresponding bit that must be updated. The parameter a, which controls the balance between $exploration$ and $exploitation$, is updated in every iteration. It is linearly decreased from 2 to 0 and is calculated as follows:

$$a = 2 - t\frac{2}{NumOfIter} \tag{24}$$

The main strategy of GWO is that the wolves update their positions toward the prey according to the positions of the best wolves. The update location is near the positions of the best wolves and closer to the prey. In addition, to minimize the number of active PMs and reduce the power consumption for the cloud data center, we must improve the solution on current active PMs and try to decrease their quantity. Consequently, updating the bits of every VM on all PMs is costly and enlarges the search space; hence, the wolves' locations move toward the prey and sometimes in the opposite direction in a binary search space.

Based on this general description of the GWO algorithm, we proposed that the corresponding value x_{ij} of submitted V_j to P_i be updated if the corresponding bit of α, β, or δ equals one. Hence, the search space is exclusive to the active PMs and the effort that is required of RIS to correct infeasible solutions is reduced. Based on the GWO algorithm and the new adjustment of its operations, BGWO-VMP is described in Algorithm 1.

3.6. DGWO-VMP Algorithm

In the discrete GWO-VMP algorithm, the solution of VMP is represented as a one-dimensional matrix of size $[1 \times m]$. Each entry represents a VM and its value is the index of the hosting PM. Thus, the entries of this matrix are integers that are in the range $[1, n]$, where n is the number of PMs. For

instance, matrix component $x_j = i$ denotes that VM j is hosted by PM i. Consequently, Equation (15) is reformulated as follows:

$$W_k = \left[x_k^{i1}, x_k^{i2}, ..., \quad x_k^{im}\right] \tag{25}$$

where W_k is the matrix position of wolf k in the pack. The corresponding variable, namely, x_k^{i1}, equals the discrete value i, $\forall i \in [1, n]$. If $V_j \in V$ is assigned to $P_i \in P$, the corresponding value equals a number in $[1, n]$.

Every wolf updates its solution according to α, β, and δ via Equation (26).

$$x^{ij}(t+1) = \frac{x_1^{ij} + x_2^{ij} + x_3^{ij}}{3}, \tag{26}$$

where $x_1^{ij}, x_2^{ij}, x_3^{ij}$ are calculated as in Equations (18)–(20).

Applying DGWO-VMP on the example that is illustrated in Figure 1 yields the following: the α solution is represented as $S = \{3, 1, 3, 1, 1\}$, which indicates that V_1 is submitted to P_3, V_3 is submitted to P_1 and so on. According to the α solution, only two PMs are active for hosting the five VMs, with consolidation $P_3 = \{V_1, V_3\}$ and $P_1 = \{V_2, V_4, V_5\}$. Similarly, according to the β solution, only three PMs are active for hosting the five VMs, with consolidation $P_2 = \{V_1, V_5\}$, $P_1 = \{V_2, V_3\}$, $P_3 = \{V_4\}$. According to the δ solution, three PMs are active for hosting the five VMs, with consolidation $P_1 = \{V_1, V_3, V_4\}$, $P_2 = \{V_2\}$, $P_3 = \{V_5\}$. Finally, according to the ω solution, four PMs are active for hosting the five VMs, with consolidation $P_1 = \{V_1\}$, $P_2 = \{V_2\}$, $P_3 = \{V_3, V_5\}$, $P_4 = \{V_4\}$. The ω solution path must be updated according to α, β, and δ based on Equation (26), where the locations of the VMs are updated one by one. However, some of the update locations of the VMs may be outside the boundary $[1, n]$. In such cases, the algorithm recalculates the allocation of VMs to guarantee that the PMs are inside the boundary as follows:

$$x^{ij}(t+1) = \begin{cases} x^{ij}(t+1) \pmod n, & if\ x^{ij}(t+1) \notin [1, n] \\ x^{ij}(t+1), & otherwise \end{cases} \tag{27}$$

Based on the GWO algorithm and the new adjustment of its operations, DGWO-VMP is described in Algorithm 1.

Algorithm 1: GWO-VMP.

Input: the number of VMs, the number of PMs

Output: VM allocation map α

Step 1: Initialization. Set parameter a via Equation (24). Set the number of wolves as k, which are considered as the search agents. Set the total number of iterations *Titer* and the iteration number $it = 1$.

Step 2: Let the k wolves construct the k solutions. Then, select the α, β, and δ solutions.

Step 3: Update all solutions based on the solutions of α, β, and δ and calculate the updated values according to Equation (16) or Equation (27) for the binary and discrete algorithms, respectively.

Step 4: Perform RIS if there is an infeasible solution.

Step 5: Evaluate the fitness values of the k solutions and identify the best three solutions so far, which are set as α, β, and δ for the current iteration.

Step 6: Termination detection. If the current iteration number exceeds the maximum number of iterations or the number of active PMs equals the preset optimal number of active PMs, then the algorithm terminates. Otherwise, increase it by 1 and return to step (3) for the next iteration.

Step 7: Return a solution α as the best solution.

4. Experiment and Comparisons

Experimental tests are conducted in this section to evaluate the performances of BGWO-VMP and DGWO-VMP. The proposed algorithms have been implemented in Java. We have used CloudSIM [44], which supports many features of IaaS, such as provisioning on-demand resources and power-aware solutions. We have used the 3.1 version toolkit of CloudSIM to create the PMs and VMs and to initially submit VMs to PMs. The experiments were performed on a computer with a 2.4 GHz Intel Xeon CPU E5-2680 v4 and 32 GB of RAM. The OS that was utilized is 64-bit Linux.

We evaluate the performances of the algorithms under homogeneous and heterogeneous cloud data center environments. The two algorithms, namely, BGWO-VMP and DGWO-VMP, are compared to other algorithms, namely, FFD [45], OEMACS [26], RGGA [19], ACO [39], MACO [40], HACOPSO [41], and PSO [42], in terms of efficiency. FFD is classified as a deterministic algorithm and yields a result that is equal to or less than 11/9* OPT+1. Therefore, FFD can represent both heuristic and deterministic solutions to the NP-hard problems. A comparison of SA and GA for solving VMP is proposed in [46]. The authors show that the GA-based method slightly outperforms the SA-based method. Thus, we compared our approach against a GA-based algorithm, RGGA [19], not an SA-based algorithm, [36]. In addition, GWO is a population-based algorithm as well as GA, where SA is a single solution based algorithm.

The related parameters for BGWO-VMP and DGWO-VMP are $a = 2$, which is linearly decreased over the course of the iterations, and $r1$ and $r2$, which are random vectors in $[0, 1]$. Few parameters must be set for BGWO-VMP and DGWO-VMP; this is one of their main advantages. The parameters of the other algorithms have been set according to their original literature. We suppose that the resource utilization can reach 100%. For BGWO-VMP, the proposed implementation uses 100 iterations with an early stop of five iterations. For DGWO-VMP, the proposed implementation uses 100 iterations; there is no early stop condition, as the results of this algorithms may be not improved by many successive iterations.

4.1. Bottleneck of a Resource Homogeneous Environment

To evaluate the efficiency of BGWO-VMP and DGWO-VMP, we adopt the dataset that was proposed in [26]. The dataset of the VMs and PMs in the cloud data center has a bottleneck resource. This test is more interesting and difficult than the typical test, in which the ratio of the total requirements of CPU and memory is 10:9. Eight instances have been created, which have sizes that range from 100 to 2000 and are numbered from A1 to A8. Each PM has a 16-core CPU and 32 GB RAM. Each VM has a CPU requirement of 1-4 cores and a memory requirement of 1-8 GB, which are randomly generated from discrete uniform distributions.

The probability of 4-core VM is 0.25, and that of 7 or 8 GB VM is 0.125. In this case, the CPU is the bottleneck resource and the ratio of the CPU and RAM requirements is approximately 10:9. The optimal solution in a random generation is unknown. Consequently, a lower bound on the optimum solution can be estimated as the maximum ratio of the sum of the VM requirements to the sum of the PM capacities, which depends on the dataset that is generated by the discrete uniform distributions.

Table 1 lists the results of the algorithms. BGWO-VMP, DGWO-VMP, and OEMACS yield the best solutions. However, BGWO-VMP and DGWO-VMP yield the best result for most of the instances. Specifically, BGWO-VMP and DGWO-VMP yield the best results for problems of larger size, which is observed for instances A6, A7, and A8. Hence, BGWO-VMP and DGWO-VMP outperform the other compared methods.

Table 1. Experimental results for the bottleneck resource test in a homogeneous environment (the best results are in bold).

No.	M	BGWO	DGWO	OEMACS	RGGA	ACO	MACO	HACOPSO	PSO	FFD
A1	100	**16**	**16**	**16**	**16**	**16**	**16**	16.93	17	18
A2	200	**32**	**32**	33	33	33	33	34	34	37
A3	300	48	48	**46**	46.03	46.63	47	47	50	53
A4	400	**63**	**63**	64	64.3	64.33	65.9	65.1	66	74
A5	500	78	78	**77**	77.63	78.93	81.93	80.5	85	91
A6	600	**94**	**94**	95	95.77	96.73	99.16	99.2	101	111
A7	1000	**158**	**158**	159	161.23	161.86	168.86	174.66	164	184
A8	2000	**316**	**316**	317	319.64	322.73	339.33	345.66	357	366

Figure 3 shows the number of active PMs that are used to host the largest two sets of VMs: A7 and A8. BGWO-VMP and DGWO-VMP yield the best results.

Figure 4 shows the mean utilizations of CPU and RAM for active PMs on instance A8. The bottleneck resource is the CPU, of which the utilization is approximately 100% by BGWO-VMP and DGWO-VMP, compared to 90% utilization of RAM. Hence, the generated VMs are balanced on PMs. Thus, both resources are utilized to maximum capacity. OEMACS have the same utilizations of CPU and RAM, where the CPU and RAM are almost fully utilized. BGWO-VMP and DGWO-VMP yield an efficient distribution compared to other algorithms.

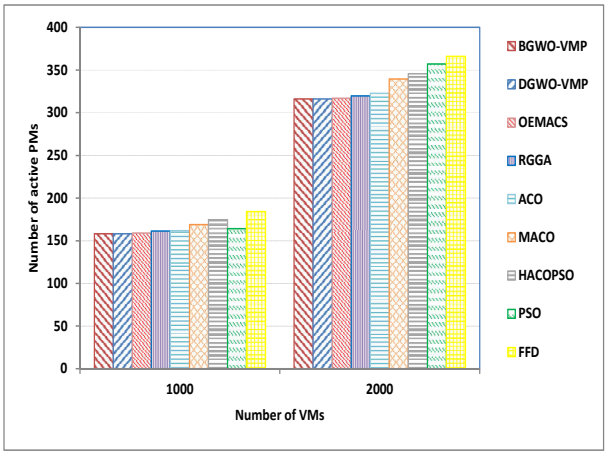


Figure 3. Numbers of active physical machines (PMs) for homogeneous instances of large-sized problems A7 and A8.

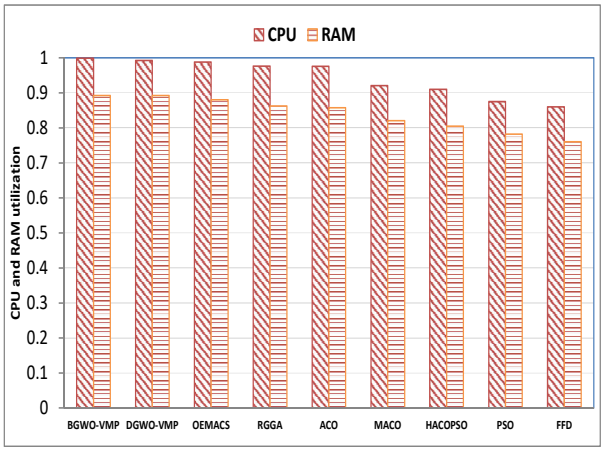


Figure 4. Mean utilizations of CPU and RAM for all active PMs on A8.

4.2. Large-Scale Heterogeneous Environmentt

In a real cloud computing data center, all PMs are often heterogeneous. In contrast to the first test, in which the PMs were homogeneous, this test considers heterogeneous PMs and CPU-intensive and RAM-intensive VMs. Two types of PMs, namely, type t_0, which has a 16-core CPU, 32 GB RAM, and $P_{max} = 215$ W, and type t_1, which has a 32-core CPU, 128 GB RAM, and $P_{max} = 300$ W, are used. We generate $9m/10$ PMs of type t_0 (m VMs) and $m/10$ PMs of type t_1. The rationale behind this is to force the placement strategy to use both types because the number of PMs of type t_1 is not sufficient for hosting all VMs. Problem instances of five sizes, which are numbered from B1 to B5, are generated by discrete uniform distributions over $[1, 8]$ for CPU and $[1, 32]$ for memory. The bottleneck resource in this datacenter is the memory.

In Table 2, the poorest result among all heuristic algorithms is obtained by FFD for all instances except B8, where the worst result is obtained by the MACO algorithm.

The MACO algorithm obtains the second worse result after the FFD for all instances; hence, this algorithm is not suitable for the heterogeneous environment. Moreover, when the size of the problem increases, the result of MACO decreases in quality. DGWO-VMP produces a satisfactory result; however, this result is worse than those of RGGA, OEMACS, and BGWO-VMP.

The BGWO-VMP algorithm yields the best result for hosting the VMs on PMs comparing to OEMACS and RGGA. BGWO-VMP yields the best result on (B2, B3, B5), where only 43, 64, and 105 PMs are used to host 200, 300, and 500 VMs, respectively. According to the average number of active PMs for thirty independent runs on each instance, the BGWP-VM algorithm typically yields the best results. The design of the BGWO-VMP algorithm and its update stage provide a satisfactory diversity of solutions, where the VM visits most of the possible PMs, which facilitates the hosting of the VMs on the minimal number of PMs with minimal waste of resources.

Table 2. Number of active physiacl machines (PMs) for the bottleneck resource test in the heterogeneous environment (the best results are in bold).

No.	M	BGWO	DGWO	OEMACS	RGGA	ACO	MACO	HACOPSO	PSO	FFD
B1	100	**19**	20.2	**19**	19.33	23.13	28.267	24.56	21	32
B2	200	**43.2**	44	45	45.26	55.26	68.26	57.76	57.2	75
B3	300	**64**	66	68	68	79.13	106.86	79.36	79.4	102
B4	400	85.5	89	**81**	82.36	103.06	137.16	114.4	105	131
B5	500	**105.7**	110.3	107	107.96	127.06	178.36	133.66	130	167

Figure 5 shows the number of active PMs for the largest two instances, namely, B4 and B5, in a heterogeneous environment. The best results for B4 were obtained by OEMACS, where only 81 active PMs are used to host 400 VMs. For B5, the best results were obtained by BGWO-VMP, followed by OEMACS, where only 105 and 107 PMs are used to host 500 VMs.

Figure 6 shows the average utilizations of all active PMs and of type 1 and type 2 PMs. The FFD algorithm has the lowest resource utilization, where FFD assigns VMs without maximizing the resource utilization or balancing the resource utilization for active PMs. According to the gap between the CPU and RAM utilizations, the distribution of FFD is poor. PSO assigns the VMs in the same way and produce substantial residual capacity for both types of PMs. The ACO, MACO, and HACOPSO algorithms yield better improvements than the FFD and PSO algorithms. However, the gap between the CPU and RAM utilizations can be used to decrease the number of active PMs with a satisfactory distribution for VMs.

BGWO-VMP, OEMACS, and RGGA produce satisfactory distributions by exploiting the resources to a high level. However, for the BGWO-VMP algorithm, the CPU and RAM utilizations of type 1 PMs are the highest with a very good balance between them compared to OEMACS and RGGA. In addition, the CPU and RAM utilizations of type 2 PMs are lower compared to OEMACS and RGGA; hence, the BGWO-VMP algorithm prefers to host VMs on type 1 PMs to the maximum extent and use

the type 2 PMs to host the remaining VMs. The gap between the CPU and RAM utilizations of type 2 PMs for the BGWO-VMP algorithm is due to the RAM being the bottleneck resource that limits the consolidation and the balance level. In other words, BGWO-VMP obtains the highest consolidation and a better balance of VMs, which reduces the number of active PMs.

For DGWO-VMP, the CPU utilization of type 1 PMs is higher compared to the OEMACS algorithm, while the CPU utilization of type 2 PMs is lower compared to OEMACS. DGWO-VMP obtains a good balance between the CPU and RAM utilizations but leaves more residual capacity than OEMACS. This results in an increase in the number of active PMs for the DGWO-VMP algorithm compared to OEMACS.

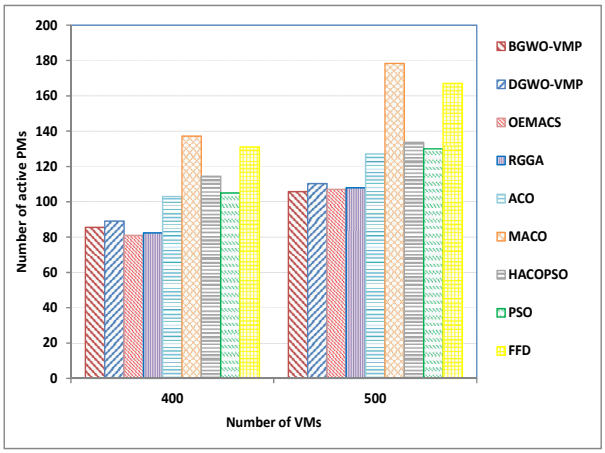


Figure 5. Number of active PMs for heterogeneous instances of large-sized problems B4 and B5.

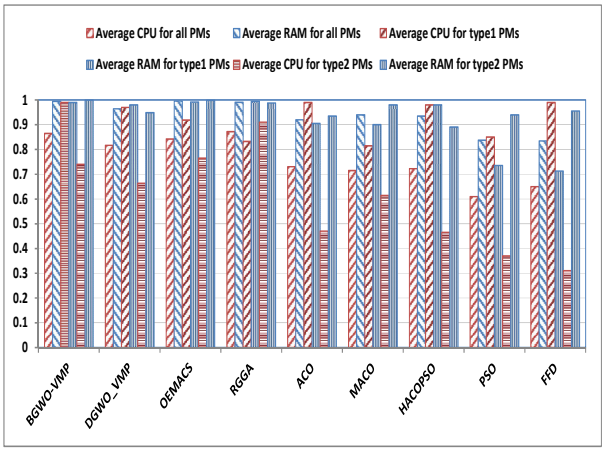


Figure 6. Average utilizations of CPU and RAM of active PMs of type 1 and type 2 on B5.

4.3. Power Consumption

Figure 7 shows the power consumption in the homogeneous environment. BGWO-VMP and DGWO-VMP obtained the lowest power consumptions comparing to the other algorithms. Compared with FFD on A8, BGWO-VMP and DGWO-VMP reduced the power consumption effectively by approximately 10.75 kW. Compared with OEMACS on A8, BGWO-VMPand DGWO-VMP reduced the power consumption by approximately 0.215 kW. Improving resource utilization by BGWO-VMP and DGWO-VMP to the maximum extent, as shown in Figure 4, by consolidating the VMs and shutting down the idle PMs reduced the power consumption efficiently.

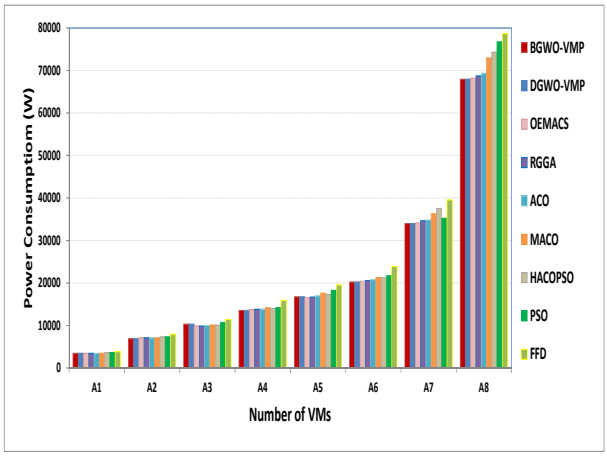


Figure 7. Power consumptions of various algorithms in the homogeneous environment.

Figure 8 depicts the energy consumption on the heterogeneous environment. BGWO-VMP has the lowest power consumption among the algorithms. Compared with FFD on B5, BGWO-VMP and DGWO-VMP reduced the power consumption effectively by approximately 8.43 kW and 6.38 kW, respectively. Compared with OEMACS on B5, BGWO-VMP reduced the power consumption by approximately 239.7 W, while OEMACS consumed less than DGWO-VMP by approximately 1.8 kW. In the heterogeneous bottleneck resource environment test, DGWO-VMP succeeded in reducing the power consumption by improving the resource utilization, as shown in Figure 6; however, it does not yield the same satisfactory result as BGWO-VMP, which produces an average power consumption that exceeds those of BGWO-VMP and OEMACS. However, the power consumption by DGWO-VMP is still less than that of MACO, HACOPSO, PSO, and FFD.

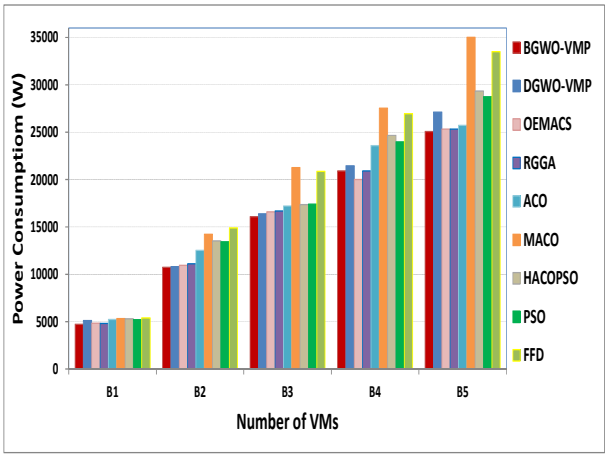


Figure 8. Power consumptions of various algorithms in the heterogeneous environment.

4.4. Further Analysis of BGWO-VMP and DGWO-VMP

In this section, we study the influences of the number of wolves involved in hunting the prey and the number of iterations on the quality of the solution. We begin our investigation with the number of wolves. We vary the number of wolves from 5 to 30 with a step size of 5. The number of iterations is set to 60. We chose A1 and A8 from the homogeneous environment tests and B1 and B5 from the heterogeneous environment tests. The average number of active PMs for each number of wolves is shown in Figure 9 for both BGWO-VMP and DGWO-VMP.

Figure 9a,b shows the number of active PMs for the BGWO-VMP and DGWO-VMP algorithms, which remain constant with the increase in the number of wolves for A1. Both algorithms yield the best results with the minimal number of wolves. For A8, DGWO yields the best result faster than

BGWO-VMP with 5 wolves, and the result is slightly improved when we increase the number of wolves. BGWO-VMP starts with a large number of active PMs with 5 wolves, improves its result to a minimum number of active PMs with 20 wolves, and becomes stable. The discrete improvement is faster than the binary, but the binary is more robust.

Figure 9c,d shows the numbers of active PMs for the BGWO-VMP and DGWO-VMP algorithms in the heterogeneous environment. For B1, both algorithms start with nearly the same result with 5 wolves and DGWO-VMP converges faster than BGWO-VMP. However, the results of both algorithms continuously improve as the number of wolves increases. For B5, BGWO-VMP obtains its best result with 5 wolves, which is superior to that of DGWO-VMP. DGWO-VMP improves its result faster than BGWO-VMP, which gradually improves its result as the number of wolves increases. With 20 wolves, the result of BGWO-VMP utilizes 105 PMs and that of DGWO-VMP utilizes 110 PMs.

We conclude that the results of both of the algorithms improve as the number of wolves increases; however, BGWO-VMP is more stable than DGWO-VMP. Moreover, the results are quickly improved from 5 to 20 wolves and change slightly from 20 to 30 wolves.

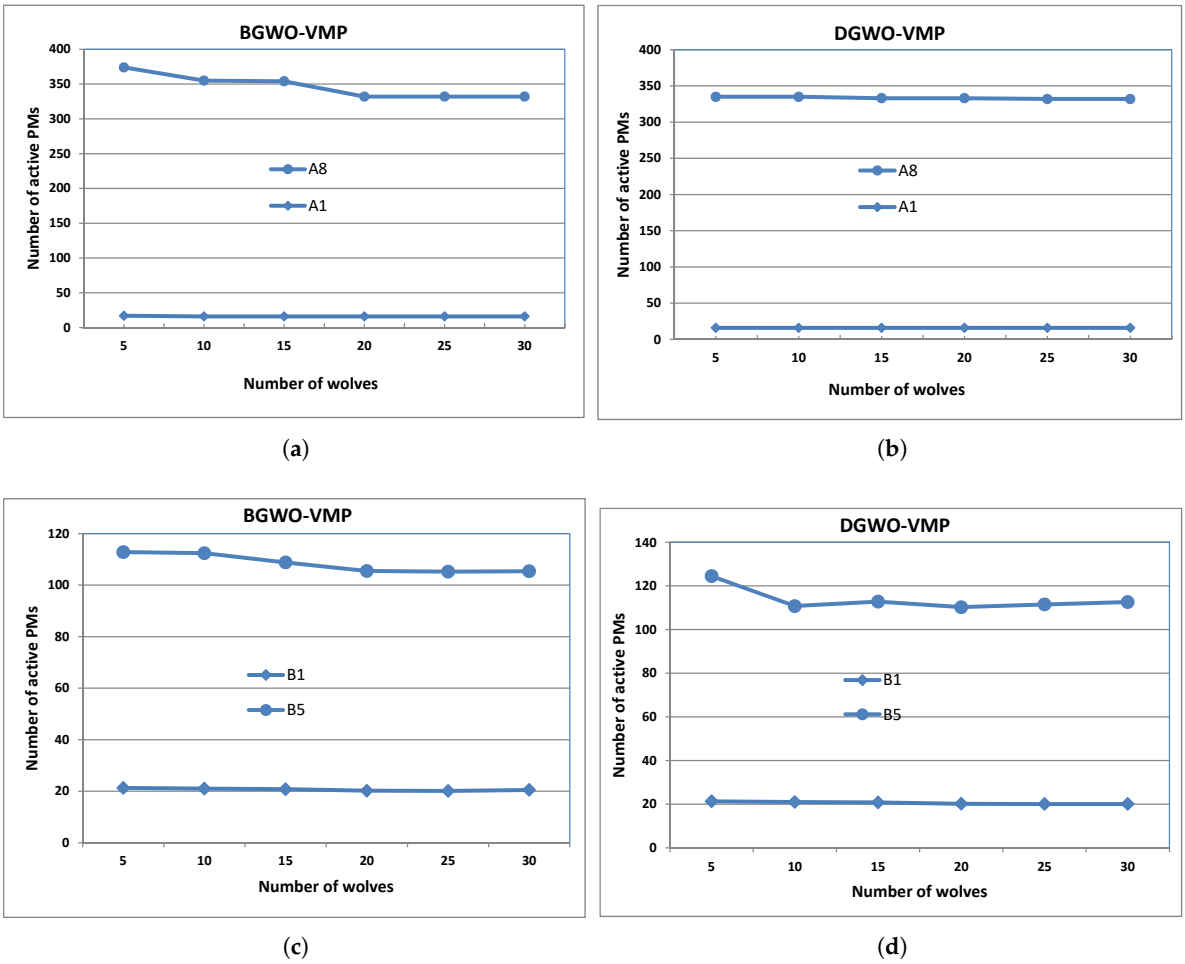
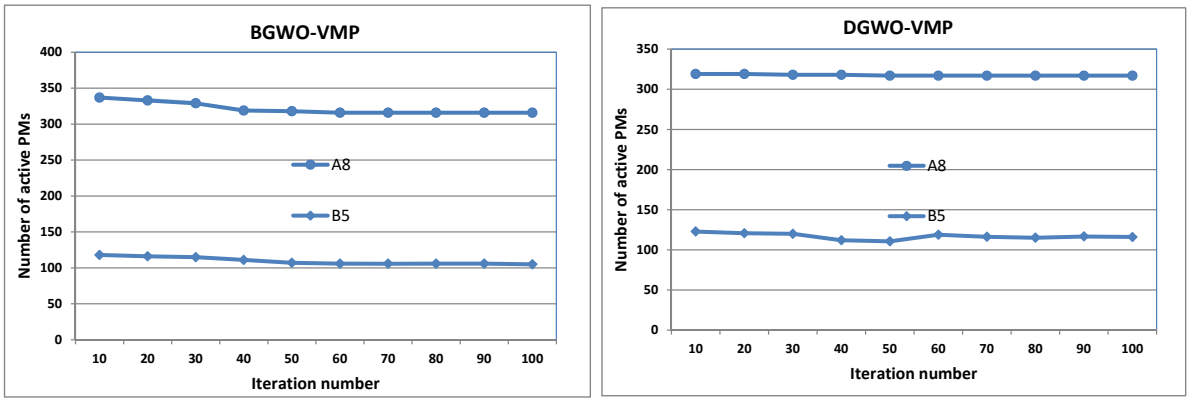


Figure 9. Influence of the number of wolves in BGWO-VMP and DGWO-VMP for both homogeneous and heterogeneous environments. (**a**) Homogeneous BGWO-VMP; (**b**) Homogeneous DGWO-VMP; (**c**) Heterogeneous BGWO-VMP; (**d**) Heterogeneous DGWO-VMP.

The results of the investigation about changing in the number of iterations on the number of active PMs is shown in Figure 10. We consider A8 from the homogeneous environment and B5 from the heterogeneous environment as examples for this study. We vary the number of iterations from 10 to 100 with a step length of 10. For A8, BGWO-VMP starts with many active PMs with

10 iterations, in contrast to DGWO-VMP, which starts with few active PMs and the same number of iterations. For BGWO-VMP, the best result is obtained after 60 iterations and becomes stable, as shown in Figure 10a. For DGWO-VMP, the best result is obtained after 50 iterations and becomes stable, as shown in Figure 10b. Hence, when the number of iterations increases, the number of active PMs decreases for both algorithms. However, BGWO-VMP requires more iterations than DGWO-VMP, which makes DGWO-VMP faster than BGWO-VMP in terms of its design and the number of iterations.

For B5, BGWO-VMP begins with fewer active PMs than DGWO-VMP with 10 iterations. Then, the number of active PMs gradually decreases as the number of iterations increases until the best result of BGWO-VMP is obtained after 70 iterations, as shown in Figure 10a. For DGWO-VMP, the number of active PMs decreases as the number of iterations increases and its best result is obtained after 50 iterations, as shown in Figure 10b. However, BGWO-VMP requires more iterations than DGWO-VMP in both the homogeneous and heterogeneous environments and obtains fewer active PMs.



(**a**) (**b**)

Figure 10. Influence of the number of iterations on BGWO-VMP (**a**) and DGWO-VMP (**b**) for both homogeneous and heterogeneous environments.

From the results of this study, we conclude that BGWO-VMP outperforms DGWO-VMP in solving the VMP optimization problem. BGWO-VMP represents the VMP problem solution as a two-dimensional array, while DGWO-VMP utilizes a one-dimensional array for the same purpose. Thus, the former obtains its solution over a wider search space in comparison to the latter. However, the convergence time of BGWO-VMP is increased. DGWO-VMP is faster than BGWO-VMP. The running time of BGWO-VMP is approximately an order of magnitude longer compared to the OEMACS method on average; BGWO-VMP is approximately twice as slow as OEMACS on average. OEMACS reported results that are based on five ants and use approximately five iterations, while B/D-GWO-VMP reported results that are based on 20 wolves and use 100 iterations. This exemplifies the time difference between the proposed method and the most efficient state-of-the-art method. For instance, the running times for B3 are 1.1, 0.7, and 0.45 s for BGWO-VMP, DGWO-VMP, and OEMACS, respectively.

5. Conclusions

In this paper, the VMP optimization task is formulated as a combinatorial problem. In this context, we propose utilizing GWO to address this problem. The problem is addressed via binary and discrete approaches. Binary approaches yield better overall performance in terms of reducing the number of active physical servers due to a superior ability to represent the problem. However, these advantages come at the cost of the running time; discrete approaches are faster than binary approaches by an order of magnitude. This is because the discrete approach has a quicker convergence in comparison, and the process of RIS, to correct the infeasible solutions, is rarely used to only correct the overload situation.

In comparison to the discrete approach, the binary approach executes the process of RIS more, and it converges slower.

The proposed method is examined via a set of experiments with various VM sizes on heterogeneous and homogeneous physical server environments. The experimental results demonstrate that the proposed method realizes a larger reduction in the number of active physical servers in comparison to the state-of-the-art methods. For future work, we will extend the proposed model to include the objectives of the dynamic VMP class. These objectives include minimizing the consumed bandwidth of VM migration, minimizing the migrated VM shutdown times, and minimizing SLA violation of the migrated VMs, in addition to the considered objective, which is to minimize the power consumption.

Author Contributions: All authors together proposed, implemented, organized and wrote the paper.

Funding: This work was supported in part by the National Natural Science Foundation Of China under Grant 61672220, in part by the Key Scientific and Technological Research and Development Plan of Hunan Province under Grant 2017GK2030, in part by the Program of China under Grant 2016YFB0201402, in part by the National Natural Science Foundation of China under Grant 61702170, Grant 61602350, Grant 61602170, and Grant 61750110531, and in part by National Key R&D Program of China under Grant SQ2018YFB020061.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alford, T.; Morton, G. *The Economics of Cloud Computing: Addressing the Benefits of Infrastructure in the Cloud*; Booz Allen Hamilton: McLean, VG, USA, 2009.
2. Zhang, Q.; Cheng, L.; Boutaba, R. Cloud computing: State-of-the-art and research challenges. *J. Internet Serv. Appl.* **2010**, *1*, 7–18. [CrossRef]
3. Li, H.H.; Fu, Y.W.; Zhan, Z.H.; Li, J.J. Renumber strategy enhanced particle swarm optimization for cloud computing resource scheduling. In Proceedings of the 2015 IEEE Congress on. Evolutionary Computation (CEC), Sendai, Japan, 25–28 May 2015; pp. 870–876.
4. Yang, B.; Li, Z.; Chen, S.; Wang, T.; Li, K. Stackelberg game approach for energy-aware resource allocation in data centers. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 3646–3658. [CrossRef]
5. Chen, Z.G.; Zhan, Z.H.; Li, H.H.; Du, K.J.; Zhong, J.H.; Foo, Y.W.; Li, Y.; Zhang, J. Deadline constrained cloud computing resources scheduling through an ant colony system approach. In Proceedings of the 2015 International Conference on Cloud Computing Research and Innovation (ICCCRI), Singapore, 26–27 October 2015; pp. 112–119.
6. Mondal, S.K.; Muppala, J.K.; Machida, F. Virtual machine replication on achieving energy-efficiency in a cloud. *Electronics* **2016**, *5*, 37. [CrossRef]
7. Mastroianni, C.; Meo, M.; Papuzzo, G. Probabilistic consolidation of virtual machines in self-organizing cloud data centers. *IEEE Trans. Cloud Comput.* **2013**, *1*, 215–228. [CrossRef]
8. Yin, L.; Luo, J.; Luo, H. Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4712–4721. [CrossRef]
9. Mathew, V.; Sitaraman, R.K.; Shenoy, P. Energy-aware load balancing in content delivery networks. In Proceedings of the 2012 IEEE INFOCOM, Orlando, FL, USA, 25–30 March 2012; pp. 954–962.
10. Mei, J.; Li, K.; Ouyang, A.; Li, K. A profit maximization scheme with guaranteed quality of service in cloud computing. *IEEE Trans. Comput.* **2015**, *64*, 3064–3078. [CrossRef]
11. Hu, J.; Luo, J.; Li, K. Opportunistic Energy Cooperation Mechanism for Large Internet of Things. *Mob. Netw. Appl.* **2018**, *23*, 489–502. [CrossRef]
12. Vogels, W. Beyond server consolidation. *Queue* **2008**, *6*, 20–26. [CrossRef]
13. Tang, Z.; Ma, W.; Li, K.; Li, K. A data skew oriented reduce placement algorithm based on sampling. *IEEE Trans. Cloud Comput.* **2016**. [CrossRef]
14. Gupta, R.K.; Pateriya, R. Energy efficient virtual machine placement approach for balanced resource utilization in cloud environment. *Int. J. Cloud-Comput. Super-Comput.* **2015**, *2*, 9–20. [CrossRef]
15. Cao, R.; Tang, Z.; Li, K.; Li, K. HMGOWM: A Hybrid Decision Mechanism for Automating Migration of Virtual Machines. *IEEE Trans. Serv. Comput.* **2018**. [CrossRef]

16. Chaisiri, S.; Lee, B.S.; Niyato, D. Optimal virtual machine placement across multiple cloud providers. In Proceedings of the 2009 IEEE Asia-Pacific Services Computing Conference (APSCC), Singapore, 7–11 December 2009; pp. 103–110.
17. Speitkamp, B.; Bichler, M. A mathematical programming approach for server consolidation problems in virtualized data centers. *IEEE Trans. Serv. Comput.* **2010**, *3*, 266–278. [CrossRef]
18. Wang, S.; Gu, H.; Wu, G. A new approach to multi-objective virtual machine placement in virtualized data center. In Proceedings of the 2013 IEEE Eighth International Conference on Networking, Architecture and Storage (NAS), Xi'an, China, 17–19 July 2013; pp. 331–335.
19. Wilcox, D.; McNabb, A.; Seppi, K. Solving virtual machine packing with a reordering grouping genetic algorithm. In Proceedings of the 2011 IEEE Congress on Evolutionary Computation (CEC), New Orleans, LA, USA, 5–8 June 2011; pp. 362–369.
20. Fatima, A.; Javaid, N.; Sultana, T.; Hussain, W.; Bilal, M.; Shabbir, S.; Asim, Y.; Akbar, M.; Ilahi, M. Virtual Machine Placement via Bin Packing in Cloud Data Centers. *Electronics* **2018**, *7*, 389. [CrossRef]
21. Foo, Y.W.; Goh, C.; Lim, H.C.; Zhan, Z.H.; Li, Y. Evolutionary neural network based energy consumption forecast for cloud computing. In Proceedings of the 2015 International Conference on Cloud Computing Research and Innovation (ICCCRI), Singapore, 26–27 October 2015; pp. 53–64.
22. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
23. Emary, E.; Zawbaa, H.M.; Hassanien, A.E. Binary grey wolf optimization approaches for feature selection. *Neurocomputing* **2016**, *172*, 371–381. [CrossRef]
24. Vaezi, M.; Zhang, Y. Virtualization and cloud computing. In *Cloud Mobile Networks*; Springer International Publishing: Basel, Switzerland, 2017; pp. 11–31.
25. Tang, Z.; Mo, Y.; Li, K.; Li, K. Dynamic forecast scheduling algorithm for virtual machine placement in cloud computing environment. *J. Supercomput.* **2014**, *70*, 1279–1296. [CrossRef]
26. Liu, X.F.; Zhan, Z.H.; Deng, J.D.; Li, Y.; Gu, T.; Zhang, J. An energy efficient ant colony system for virtual machine placement in cloud computing. *IEEE Trans. Evolut. Comput.* **2016**, *22*, 113–128. [CrossRef]
27. Greenberg, A.; Hamilton, J.; Maltz, D.A.; Patel, P. The cost of a cloud: Research problems in data center networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *39*, 68–73. [CrossRef]
28. Kusic, D.; Kephart, J.O.; Hanson, J.E.; Kandasamy, N.; Jiang, G. Power and performance management of virtualized computing environments via lookahead control. *Clust. Comput.* **2009**, *12*, 1–15. [CrossRef]
29. Beloglazov, A.; Buyya, R.; Lee, Y.C.; Zomaya, A. A taxonomy and survey of energy-efficient data centers and cloud computing systems. In *Advances in Computers*; Elsevier: Amsterdam, The Netherlands, 2011; Volume 82, pp. 47–111.
30. Barroso, L.A.; Hölzle, U. The case for energy-proportional computing. *Computer* **2007**, *14*, 33–37. [CrossRef]
31. Fan, X.; Weber, W.D.; Barroso, L.A. Power provisioning for a warehouse-sized computer. In *ACM SIGARCH Computer Architecture News*; ACM: New York, NY, USA, 2007; Volume 35, pp.13–23.
32. Chen, M.; Zhang, H.; Su, Y.Y.; Wang, X.; Jiang, G.; Yoshihira, K. Effective VM sizing in virtualized data centers. In Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management. Citeseer, Dublin, Ireland, 23–27 May 2011; pp. 594–601.
33. Lawey, A.Q.; El-Gorashi, T.E.; Elmirghani, J.M. Distributed energy efficient clouds over core networks. *J. Lightw. Technol.* **2014**, *32*, 1261–1281. [CrossRef]
34. Eusuff, M.; Lansey, K.; Pasha, F. Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization. *Eng. Optim.* **2006**, *38*, 129–154. [CrossRef]
35. Kansal, N.J.; Chana, I. Artificial bee colony based energy-aware resource utilization technique for cloud computing. *Concurr. Comput. Pract. Exp.* **2015**, *27*, 1207–1225. [CrossRef]
36. Wu, Y.; Tang, M.; Fraser, W. A simulated annealing algorithm for energy efficient virtual machine placement. In Proceedings of the 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Seoul, Korea, 14–17 October 2012; pp. 1245–1250.
37. Cho, K.M.; Tsai, P.W.; Tsai, C.W.; Yang, C.S. A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing. *Neural Comput. Appl.* **2015**, *26*, 1297–1309. [CrossRef]
38. Wen, W.T.; Wang, C.D.; Wu, D.S.; Xie, Y.Y. An aco-based scheduling strategy on load balancing in cloud computing environment. In Proceedings of the 2015 Ninth International Conference on Frontier of Computer Science and Technology (FCST), Dalian, China, 26–28 August 2015; pp. 364–369.

39. Feller, E.; Rilling, L.; Morin, C. Energy-aware ant colony based workload placement in clouds. In Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing, Lyon, France, 21–23 September 2011; IEEE Computer Society: Washington, DC, USA, 2011; pp. 26–33.
40. Tawfeek, M.A.; El-Sisi, A.B.; Keshk, A.E.; Torkey, F.A. Virtual machine placement based on ant colony optimization for minimizing resource wastage. In *International Conference on Advanced Machine Learning Technologies and Applications*; Springer: Cham, Switzerland, 2014; pp. 153–164.
41. Suseela, B.B.J.; Jeyakrishnan, V. A multi-objective hybrid ACO–PSO optimization algorithm for virtual machine placement in cloud computing. *Int. J. Res. Eng. Technol.* **2014**, *3*, 474–476.
42. Abdessamia, F.; Tai, Y.; Zhang, W.Z.; Shafiq, M. An Improved Particle Swarm Optimization for Energy-Efficiency Virtual Machine Placement. In Proceedings of the 2017 International Conference on Cloud Computing Research and Innovation (ICCCRI), Singapore, 11–13 April 2017; pp. 7–13.
43. Braiki, K.; Youssef, H. Multi-Objective Virtual Machine Placement Algorithm Based on Particle Swarm Optimization. In Proceedings of the 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), Singapore, 11–12 April 2018; pp. 279–284.
44. Buyya, R.; Ranjan, R.; Calheiros, R.N. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In Proceedings of the 2009 International Conference on High Performance Computing & Simulation, Leipzig, Germany, 21–24 June 2009; pp. 1–11.
45. Ajiro, Y.; Tanaka, A. Improving packing algorithms for server consolidation. In Proceedings of the International CMG Conference; San Diego, CA, USA, 2–7 December 2007; Volume 253, pp. 399–406.
46. Cavdar, M.C. A Utilization Based Genetic Algorithm for Virtual Machine Placement in Cloud Computing Systems. Ph.D. Thesis, Bilkent University, Ankara, Turkey, 2016.