algorithms

MDPI

Article

A Fuzzy Grouping Genetic Algorithm for Solving a Real-World Virtual Machine Placement Problem in a Healthcare-Cloud

Nawaf Alharbe [1,*], Abeer Aljohani [1] and Mohamed Ali Rakrouki [1,2,3]

[1] Applied College, Taibah University, Medina 42353, Saudi Arabia; aahjohani@taibahu.edu.sa (A.A.); mrakrouki@taibahu.edu.sa (M.A.R.)
[2] Ecole Supérieure des Sciences Economiques et Commerciales de Tunis, University of Tunis, Montfleury 1089, Tunisia
[3] Business Analytics and DEcision Making Lab. (BADEM), Tunis Business School, University of Tunis, Bir El Kassaa 2059, Tunisia
* Correspondence: nrharbe@taibahu.edu.sa

Abstract: Due to the large-scale development of cloud computing, data center electricity energy costs have increased rapidly. Energy saving has become a major research direction of virtual machine placement problems. At the same time, the multi-dimensional resources on the cloud should be used in a balanced manner in order to avoid resources waste. In this context, this paper addresses a real-world virtual machine placement problem arising in a Healthcare-Cloud (H-Cloud) of hospitals chain in Saudi Arabia, considering server power consumption and resource utilization. As a part of optimizing both objectives, user service quality has to be taken into account. In fact, user quality of service (QoS) is also considered by measuring the Service-Level Agreement (SLA) violation rate. This problem is modeled as a multi-objective virtual machine placement problem with the objective of minimizing power consumption, resource utilization, and SLA violation rate. To solve this challenging problem, a fuzzy grouping genetic algorithm (FGGA) is proposed. Considering that multiple optimization objectives may have different degrees of influence on the problem, the fitness function of the proposed algorithm is calculated with fuzzy logic-based function. The experimental results show the effectiveness of the proposed algorithm.

Keywords: genetic algorithm; healthcare cloud; virtual machine placement; virtualization

check for updates

Citation: Alharbe, N.; Aljohani, A.; Rakrouki, M.A. A Fuzzy Grouping Genetic Algorithm for Solving a Real-World Virtual Machine Placement Problem in a Healthcare-Cloud. *Algorithms* **2022**, *15*, 128. https://doi.org/10.3390/a15040128

Academic Editors: Angel A. Juan and Frank Werner

Received: 20 February 2022
Accepted: 13 April 2022
Published: 14 April 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In 2020, the world has entered one of the most serious health crises in its history. The outbreak of the COVID-19 pandemic has caused a significant spike demand for hospital and healthcare services. In addition to the health side of this pandemic, healthcare systems need reliable information systems to deal with high volume of users' requests. In this context, a hospital chain in Saudi Arabia needs to optimize its Healthcare-Cloud. This H-Cloud is a centralized cloud computing infrastructure that serves over 3000 healthcare staff across 11 hospital locations and an estimated 15,000 users of various types (see Figure 1). It is designed as a national private cloud that supports all of the applications used by each of the user's hospital branches and health systems. This cloud ensures that patient care is not disrupted or compromised in the event of an infrastructure or component failure. The growing number of linked medical devices is adding to the workload of IT specialists. Data are moved to the cloud or edge networks as a result of this influx for more flexibility, better performance, and cost savings.

The optimization problem of this Healthcare-Cloud was formulated as a multi-objective virtual machine placement problem with the objective of minimizing power consumption, resource utilization, and improving user application performance. Resource utilization is evaluated based on the degree of deviation of CPU and memory utilization on the

physical machine, and user application performance is evaluated by the data center SLA violation rate.

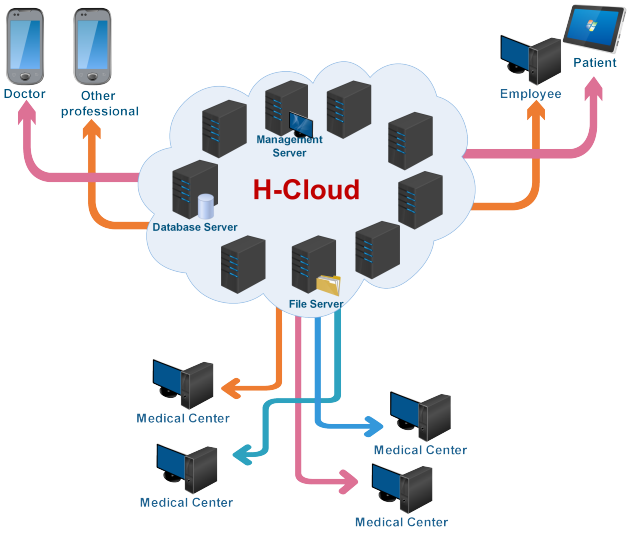

Figure 1. H-Cloud illustration.

The virtual machine placement problem has been widely concerned by researchers and has become a hot topic. The placement of virtual machines directly affects the service performance, resource utilization, and power consumption of cloud data centers. Therefore, the research on virtual machine placement is a scientific problem with practical significance. Most of the literature focuses on improving resource utilization and load balancing. However, as the scale of data centers continues to expand, the power consumption increases rapidly. Data centers in the United States consumed almost 70 billion kWh in 2010, accounting for about 1.8% of total domestic electricity usage, growing from around 30 billion kWh in 2000 [1,2]. So, saving energy has become an urgent problem for cloud data centers. Generally, researchers achieve this by aggregating servers, and they achieve the goal of energy saving by minimizing the number of activated physical machines [3,4]. This method is indeed effective in terms of energy saving. However, if the physical machines are overloaded with too many tasks, the performance of user applications decreases, resulting in a poor user experience. Therefore, while considering energy saving, it is important to remember that the data center's ultimate goal is to serve users, and the quality of service is not something to be ignored. At the same time, improving the balanced utilization of server resources is also an effective way to improve data center efficiency and reduce resource waste. For example, the CPU utilization on a physical machine has reached 85%, while the memory utilization is only 20%. At this time, the physical machine cannot allocate other virtual machines on it because the CPU utilization has reached the threshold. The memory resources are wasted, which increases the probability of starting a new physical server, and it increases the number of migrations, resulting in more waste of resources.

The remainder of this paper is organized as follows. In Section 2, some important related works to the virtual machine placement problem are presented. The description of the problem under consideration is presented in Section 3. In Section 4, we propose a hybrid group genetic algorithm to our problem. The experimental results are presented in Section 5. Finally, Section 6 summarizes this research work.

2. Literature Review

The problem of virtual machine placement has been widely studied in the literature. The following are some important papers related to the problem under consideration.

SLA has been considered for the first time by Pires and Barán [5], who comprehensively optimized the energy consumption, network traffic, and economic benefits of cloud providers. The authors defined SLA as that any virtual machine requested by the user must

meet its physical resource allocation; that is, the placement request of the virtual machine must be enforced to ensure the correct operation of mission-critical applications. In large data centers, users' virtual machine requests can generally be satisfied. Therefore, the SLA defined in this paper is not innovative.

Zhiqiang and Haibo [6] proposed a virtual machine consolidation strategy based on server load prediction to solve the problem of data center load balancing, reduce server energy consumption, and meet SLA. In this paper, a local regression heuristic algorithm is used to predict the resource utilization at the next moment according to the historical value of server resource utilization and migrate the low-load server and the virtual machine on the overloaded server to improve the overall resource utilization of the data center and turn off low power consumption. This method achieves the goal of saving energy and improving resource utilization through virtual machine migration, which is different from the idea in our work. Similarly, Liao et al. [7] achieved dynamic server aggregation using virtual machine migration strategies, which is appropriate for cloud environments with a small number of virtual machines but not for a large-scale virtual machine placement problems.

Gao et al. [8] proposed an ant colony optimization algorithm to minimize the total waste of resources and power consumption. The algorithm has achieved good results in terms of energy saving and improving resource utilization, but the author did not consider the impact of the optimization of these two goals on user applications. In Mosa and Paton [9], a genetic algorithm has been proposed in order to optimize energy consumption and SLA violations. The proposed method adopts utility functions to formulate the virtual machine placement problem. The experimental results showed that the proposed algorithm reduced the average energy consumption by approximately 6% and the overall SLA violations by more than 38% compared to well-known heuristics. Kumar and Raza [10] proposed a modified PSO algorithm to reduce total resource waste and psychical server use. When compared to typical placement strategies such as Best-Fit, First-Fit, and Worst-Fit, the proposed algorithm produces better results. Farzai et al. [11] proposed a hybrid multi-objective genetic-based optimization algorithm by considering three objectives: reduced power consumption, reduced resource wastage, and reduced bandwidth. The experimental results demonstrated a high potential of scalability for large problem instances. Gopu and Venkataraman [12] proposed a multi-objective evolutionary algorithm in order to reduce resource wastage, power consumption, and propagation delay in a distributed cloud environment. The experiments have shown that the proposed approach outperforms existing algorithms. Alhammadi and Vasanthi [13] proposed two multi-objective algorithms for minimizing energy consumption, SLA violation, and the number of migrated virtual machines. The experimental results have shown that the proposed algorithms outperformed some simple well-known heuristics.

Recently, a new hybrid metaheuristic has been proposed by Pushpa and Siddappa [14] in order to minimize resources load, virtual machine migration cost, and power consumption. The proposed method hybridizes artificial bee colony and cat swarm optimization algorithms. In Lu et al. [15], an improved genetic algorithm has been proposed for minimizing the energy consumption of the cloud data center and maintaining its high availability. Mejahed and Elshrkawey [16] proposed a multi-objective hybrid particle swarm optimization for minimizing the placement time of requested virtual machines, power consumption, and resource wastage in a cloud data center. The proposed approach used a flower pollination optimization algorithm when trapped in local optima, and the experiments show that when using this technique, the algorithm has become more efficient and outperformed the bin-packing best-fit strategy in terms of server utilization. Li et al. [17] studied the problem of minimizing energy consumption and computing delay in edge cloud computing, and they proposed a genetic algorithm to solve this problem. This method has been improved by a low-rank representation algorithm and a minimal migration time algorithm.

An extensive literature review of ant colony-based approaches for solving the virtual machine placement problem in a cloud environment has been presented by Asghari and Jafari Navimipour [18].

3. Problem Description

Users apply to cloud service providers for virtual machine resources to deploy their own applications. To ensure good service performance, cloud service providers need to sign SLAs with users. The virtual machine placement problem is described as follows: cloud service providers design virtual machine scheduling strategies to meet the virtual machine resource requests of each user and at the same time reduce power consumption of the cloud data center, reduce resources waste, and reduce the SLA violations rate. The virtual machine placement problem can be formulated as a multi-dimensional bin packing problem, which is proven to be $\mathcal{NP}$-hard [19].

Based on the above description, the following definitions are given:

We are given a cloud data center sharing M physical machines, denoted as $P = \{P_1, P_2, \ldots, P_i, \ldots, P_M\}$. Each physical machine P_i has d available resources (such as CPU, memory, bandwidth, and storage). Each resource j has a capacity $H_{i,j}$, where $j \leq d$. The users applied for a group of N virtual machines $V = \{V_1, V_2, \ldots, V_i, \ldots, V_N\}$. The required resource j of any virtual machine V_i is denoted as $R_{i,j}$.

Definition 1. *A cloud data center shared M physical machines, denoted as* $P = \{P_1, P_2, \ldots, P_i, \ldots, P_M\}$. *The number of types of available resources (such as CPU, memory, bandwidth, and storage) for a physical machine* P_i *is denoted as d, and* $H_{i,j}$ *is the capacity of the resource j on a physical machine* P_i, *where* $j \leq d$.

Definition 2. *A user applies for a group* $V = \{V_1, V_2, \ldots, V_i, \ldots, V_N\}$ *of N virtual machines. The required resources of any* V_i *is denoted as* $R_{i,j}$, *which represents the number of requirements for a resource j on a virtual machine* V_i, *where* $j \leq d$. *In this paper, we only consider CPU and memory as resources, and we do not consider disks. The reason is that cloud computing data centers generally use network-attached storage (NAS), and storage can be used as a separate module, so* $d = 2$.

3.1. Energy Consumption Modeling

Server power consumption is mainly affected by CPU, memory, disk storage, and bandwidth; of these, CPU has the largest contribution to server power consumption and accounts for most of the energy consumption. For simplicity, we consider only the impact of CPU on power consumption. In this model, the power consumption of the server fluctuates depending on the usage of the CPU on this server. According to [20], it can be seen that when the server changes from zero load (0%) to full load (100%), the power consumption of the server has a linear relationship with CPU utilization, and the power consumed by the server at zero load is 67% of the power consumed at full load. The power consumption W_i of a physical machine P_i can be calculated as follows:

$$W_i = (W_{\max} - W_{idle}) U^i_{cpu} + W_{idle} \tag{1}$$

where $W_{\max}$ and W_{idle} indicate the power consumption when the server is fully loaded and idle, respectively; U^i_{cpu} is the CPU utilization rate on the physical machine P_i.

3.2. Resource Wastage Modeling

The virtual machine placement problem is a multi-dimensional bin-packing problem, involving components such as CPU and memory. The resource usage of each physical node in the data center should be kept as balanced as possible in different dimensions to avoid wasting resources due to the barrel effect and reduce resource utilization. As shown in Figure 2, the rectangular box represents the memory capacity of the physical machine. Physical nodes have less and less resources in each dimension. The final remaining resources are shown in the dark shaded rectangle. The CPU resource remaining rate is much larger than the memory resource remaining rate. At this time, due to the lack of memory

resources, no more virtual machines can be allocated to the physical machine, and the remaining CPU resources are wasted.

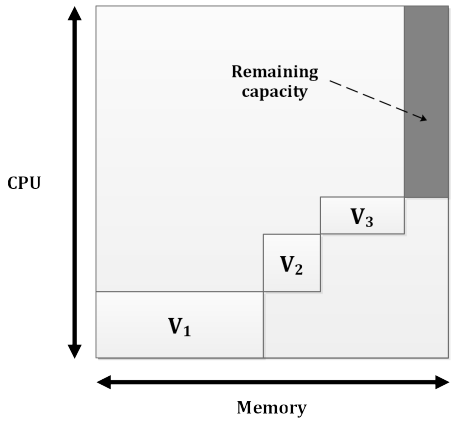

Figure 2. Example of resource allocation for 3 virtual machines deployed on a physical machine.

The physical node resources waste can be calculated as follows:

$$R_i = \frac{\mid L^i_{cpu} - L^i_{mem} \mid + \epsilon}{U^i_{cpu} + U^i_{mem}} \tag{2}$$

$$R = \sum_{i=1}^{n} R_i \tag{3}$$

where L^i_{cpu} is the ratio of the remaining CPU resources to the total CPU resources of a physical node P_i, L^i_{mem} indicates the memory remaining ratio of the node P_i, and ϵ is a small positive real number (we set it to 0.0001 to facilitate subsequent calculations).

3.3. SLA Violation Modeling

In cloud computing services, the cloud provider needs to sign an SLA with the user, which defines the service quality, responsibilities, billing system, etc. The service quality of virtual machine allocation is generally based on a user's application performance. Cloud providers must allocate resources to users that match their needs to guarantee performance. In our work, we defined the SLA violation rate as a function of the CPU utilization rate. The reason is that the CPU utilization of physical machines is closely related to the SLA violation rate. When CPU utilization increases, the rate of SLA violations increases with it; when CPU utilization reaches a peak, the rate of SLA violations increases quickly. Li et al. [21] defined the evaluation function of the SLA violation rate as follows:

$$f_i = \ln(2 + U^i_{cpu} - U^{\max}_{cpu}) \tag{4}$$

where $U^{\max}_{cpu}$ is the maximum CPU utilization rate on the physical machine.

3.4. Constraints

The problem under consideration includes the following constraints:

(a) Any virtual machine V_i can only be assigned to one physical node P_j. Let $X_{i,j} \in \{0, 1\}$ be a binary variable, $X_{i,j} = 1$ if V_i is assigned to P_j and 0 otherwise:

$$\sum_{j=1}^{M} X_{i,j} \leq 1; \forall i \in \{1, 2, \ldots, N\} \tag{5}$$

(b) The virtual machine requested by the user must meet its allocation requirements:

$$\sum_{j=1}^{M} X_{i,j} = 1; \forall i \in \{1, 2, \ldots, N\}; \tag{6}$$

(c) Physical server resource limit: the virtual machine allocated on each physical node does not occupy more resources than its capacity:

$$\sum_{i=1}^{M} X_{i,j} R_{i,cpu} \leq H_{j,cpu}; \forall j \in \{1, 2, \ldots, M\}; \tag{7}$$

$$\sum_{i=1}^{M} X_{i,j} R_{i,mem} \leq H_{j,mem}; \forall j \in \{1, 2, \ldots, M\}; \tag{8}$$

where $R_{i,cpu}$ ($R_{i,mem}$) represents the CPU requirements (memory requirements) of a virtual machine V_i, and $H_{j,cpu}$ ($H_{j,mem}$) represents the remaining available CPU resources (memory resources) on a physical node P_j.

3.5. Optimization Goal

Our goal is to minimize energy consumption, resource waste, and SLA violation rate:

$$\min W = \sum_{i=1}^{M} W_i \tag{9}$$

$$\min R = \sum_{i=1}^{M} R_i \tag{10}$$

$$\min f = \sum_{i=1}^{M} f_i \tag{11}$$

4. Improved Grouping Genetic Algorithm FGGA

At present, there are many evolutionary algorithms for solving multi-objective optimization problems. This type of algorithm abstracts a mathematical model that conforms to certain laws through the simulation of biological phenomena. They have strong self-organization and self-adaptation capabilities. In this section, we will present the grouping genetic algorithm (GGA) to solve our problem. The following will analyze the advantages and disadvantages of the GGA algorithm, and it will clarify the reasons for choosing this algorithm and the basis for subsequent improvements.

(1) Advantages of GGA algorithm:

1. The grouping genetic algorithms have been proved to be suitable for grouping problems, such as virtual machine placement, bin packing, vehicle routing, jobshop scheduling, and clustering [22,23].
2. The GGA algorithm uses a block coding method, which is consistent with the characteristics of the virtual machine placement problem. The physical machine is regarded as a chromosome, and the virtual machine deployed on it is regarded as a gene.
3. The search process is inspired by fitness values, and the process is simple.

(2) GGA algorithm defects:

1. Algorithm performance is more dependent on the selection of parameters, and an improper selection of parameters will seriously affect the pros and cons of the solution.
2. The crossover and mutation operations in the algorithm randomly select genes on the chromosome, which is blind and affects the convergence speed of the algorithm.
3. When evaluating multi-objective optimization problems, the fitness function transforms the linear summation of multiple objectives into a single-objective problem, which is inconsistent with the fact that the degree of influence of multiple objectives on the problem is uncertain in reality.

Based on the above analysis, we proposed a Fuzzy Grouping Genetic Algorithm (FGGA) to solve the multi-objective scheduling problem, which improves the crossover and mutation operations in the basic grouping genetic algorithm to ensure the diversity of the population while ensuring that the population evolves toward the Pareto optimal solution. Then, we introduce fuzzy logic theory. We perform fuzzy processing on multiple optimization goals, determine the fitness function based on the principle of maximizing minimum satisfaction, and evaluate the pros and cons of the solution.

4.1. FGGA Algorithm Process

The GGA algorithm process mainly has the following steps: gene coding, setting fitness function, selection operation, crossover operation, and mutation operation. Based on the GGA algorithm, we use fuzzy logic to determine the fitness function, solve the uncertainty of the impact of multiple objectives in the problem, and improve and optimize crossover and mutation operators to overcome the random selection in the GGA algorithm. Figure 3 shows our proposed FGGA algorithm. In the flow chart, the main improvement points are reflected in the dotted rectangle box. This section will go over the specific process of improving the proposed algorithm.

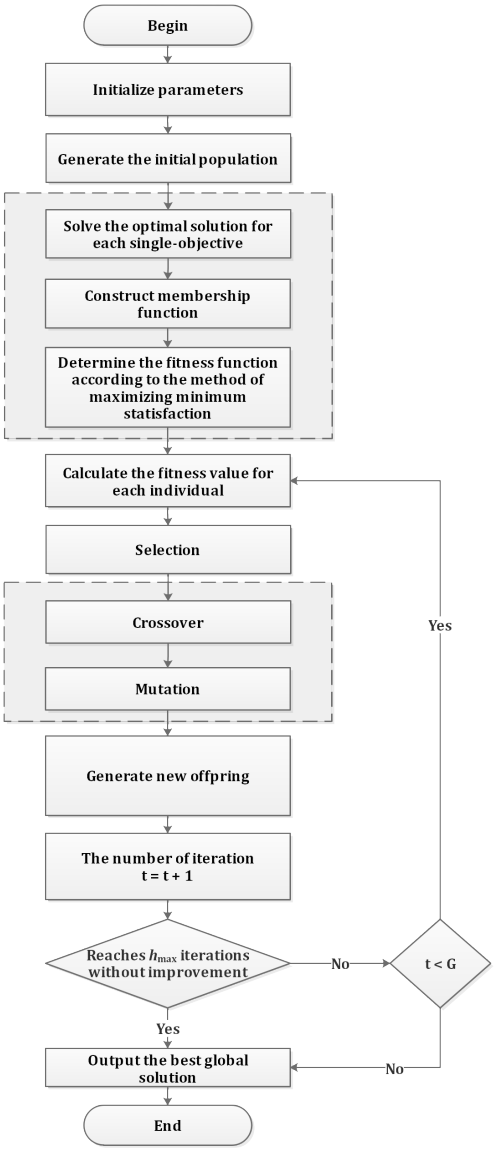

Figure 3. Flow chart of our proposed FGGA algorithm.

4.2. Gene Coding

In the group coding method, coding is performed in groups. In our algorithm, a chromosome (solution) is divided into groups (physical machines). Each group is divided into genes (virtual machines). As shown in Figure 4, seven virtual machines are deployed on three servers. The code of the chromosome is ABC, and the gene value on each chromosome is the number of the virtual machine deployed on the server. The advantage of this is that it can solve the problem of unclear grouping information in the traditional encoding method; that is, during crossover and mutation, the virtual machine can be operated, and the server information can be transparent.

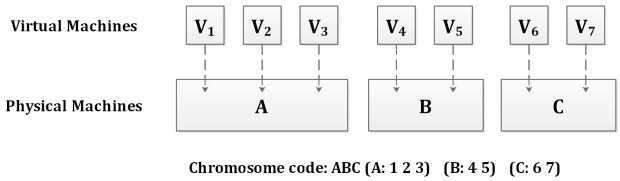

Figure 4. Virtual machine group coding.

4.3. Fitness Function

The fitness function is used in genetic algorithms to assess the quality of a solution. In our problem, the larger the fitness value, the easier the solution is accepted, and the smaller it is, the easier it is to be eliminated. The fitness function is the selection criterion for the survival of the fittest in the genetic algorithm and the driving force for the evolution of the population. Its design and structure should be combined with the objective function of the problem. The optimization goal of the problem under consideration is to minimize the power consumption, resource utilization, and SLA violation rate at the same time. A multi-objective optimization problem can be solved in one of two ways. One approach is to use a linear-weighted sum to reduce a multi-objective problem to a single-objective problem. The other alternative is to optimize each objective function independently. Both of these methods require decision makers to clearly know the importance of each optimization item and give a certain weight. In a virtual machine placement problem, it is often difficult for decision makers to judge the impact of each goal on the problem.

Therefore, in order to calculate the fitness function for a solution, we used fuzzy logic to evaluate our multiple optimization goals. Fuzzy logic was introduced by [24]. It is different from classical logic in that there is no precise boundary between true and false; that is, from true to false is a gradual change process. This process is reflected by the membership function. Many fuzzy logic models have been proposed in the literature. In our work, we have used the maximum and minimum satisfaction fuzzy logic model to fuzzify our three optimization goals. The steps for determining the fitness function are as follows.

4.3.1. Solve Each Single-Objective Problem

In order to clarify whether the final solution performs well in each optimization objective, it is first necessary to solve the each single-objective problem. Since the power consumption, resource utilization, and SLA of the cloud data center are dynamically changing, it is difficult to solve a definite optimal value. Therefore, we give an estimate of the optimal values of the three objective functions.

First, we determine the minimum number $L_{\min}$ of physical machines in the data center that are turned on, as expressed by Equation (12), where R_{sum_cpu} and R_{sum_mem} represent the sum of CPU resources and the sum of memory resources requested by all virtual machines in the data center, respectively, and H_{cpu}, H_{mem} indicate the CPU resources and memory resources owned by a single physical machine, respectively. The resource requests of any virtual machine that have been restricted in our work are all satisfiable, so the maximum number of enabled physical machines is $L_{\max} = N$. According to Equation (12),

it can be known that the physical nodes that are turned on are in a fully loaded state, so the optimal value of the total power consumption W^* of the data center can be estimated by Equation (13). The optimal value of resource waste R^* is expressed by Equation (15), where H_{sum_cpu} and H_{sum_mem} represent the sum of CPU resources and the sum of memory resources in the data center, respectively. The upper and lower limits of the SLA violation rate are determined according to Equation (4).

$$L_{\min} = \max\left(\frac{R_{sum\_cpu}}{H_{cpu}}, \frac{R_{sum\_mem}}{H_{mem}}\right) \tag{12}$$

$$W^* = L_{\min} \cdot P_{\max} \tag{13}$$

$$W_{\max} = L_{\max} \cdot P_{\max} \tag{14}$$

$$R^* = \left|\frac{R_{sum\_cpu}}{H_{sum\_cpu}} - \frac{R_{sum\_mem}}{H_{sum\_mem}}\right| \tag{15}$$

$$R_{\max} = \sum_{i=1}^{M}\left|\frac{R_{i\_cpu}}{H_{i\_cpu}} - \frac{R_{i\_mem}}{H_{i\_mem}}\right| \tag{16}$$

4.3.2. Membership Function

According to the best value obtained in Section 4.3.1, the membership function of each objective function is constructed. The membership function reflects the degree of the optimization objective. The smaller the degree of membership, the more unacceptable the solution. A value of 0 is completely unacceptable, and a value of 1 is an optimal value. In the following, $\mu(f_i(x))$ represents the membership value of the ith optimization objective.

$$\mu(f_1(x)) = \begin{cases} 1; & \text{if } f_1(x) \le W^* \\ 1 - \dfrac{f_1(x) - W^*}{\delta_1}; & \text{if } W^* < f_1(x) \le W^* + \delta_1 \\ 0; & \text{if } f_1(x) > W^* + \delta_1 \end{cases} \tag{17}$$

$$\mu(f_2(x)) = \begin{cases} 1; & \text{if } f_2(x) \le R^* \\ 1 - \dfrac{f_2(x) - R^*}{\delta_2}; & \text{if } R^* < f_2(x) \le R^* + \delta_2 \\ 0; & \text{if } f_2(x) > R^* + \delta_2 \end{cases} \tag{18}$$

$$\mu(f_3(x)) = \begin{cases} 1; & \text{if } f_2(x) \le F^* \\ 1 - \dfrac{f_3(x) - F^*}{\delta_3}; & \text{if } F^* < f_3(x) \le F^* + \delta_3 \\ 0; & \text{if } f_3(x) > F^* + \delta_3 \end{cases} \tag{19}$$

where $W^* + \delta_1$, $R^* + \delta_2$, and $F^* + \delta_3$ represent the maximum power consumption, the maximum value of resource waste, and the maximum value of SLA violation, respectively, and δ_i represents the difference between the optimal value and the maximum value.

4.3.3. Fitness Function Construction

We have used the maximization and minimum satisfaction method to determine the fitness function. This method makes all objective functions have the highest possible membership degree. From Section 4.3.2, we can see that the higher the membership degree of the objective function, the closer the goal is to the optimal solution. The fitness function is stated as follows.

$$\mu(x) = \min\{\mu_1(x), \mu_2(x), \mu_3(x)\} \tag{20}$$

Then, the multi-objective virtual machine placement problem is transformed into a maximized single-objective problem, such as in Equation (21).

$$\max \mu(x)$$
$$s.t. \begin{cases} \mu_1(x) \geq \mu(x) \\ \mu_2(x) \geq \mu(x) \\ \mu_3(x) \geq \mu(x) \\ \sum\limits_{i,j=1} x_{s,t}^{i,j} - x_{t,s}^{i,j} = 0 \\ \sum\limits_{i,j=1} x_{s,t}^{i,j} \leq C_{s,t} \\ x_{s,t}^{i,j} \geq 0 \end{cases} \tag{21}$$

4.4. Selection Operator

The genetic algorithm imitates the principle of biological evolution and uses selection operations to ensure the survival of the fittest. Individuals with high fitness have a greater probability of being inherited to their offspring, and individuals with low fitness have a smaller probability of being inherited to their offspring. If only individuals with high fitness are selected for inheritance, they will fall into a local optimum, so the roulette method is used to select inherited individuals.

First, we calculate the fitness value of all individuals in the population $\mu(x_i), \forall i \in \{1, 2, \ldots, m\}$, where m is the population size, and we calculate the total fitness of the population. Then, the probability of selection $p(x_i)$ of the individual x_i is calculated as in Equation (22). The cumulative probability of the individual is calculated as in Equation (23), and we use the simulating roulette to randomly generate a random number $r \in [0, 1]$ where $q_{i-1} < r \leq q_i$. Then, the ith individual will be inherited to the next generation. This selection operator can avoid the defect of falling into a local optimum caused by only selecting individuals with high fitness for inheritance.

$$p(x_i) = \frac{\mu(x_i)}{\sum\limits_{j=1}^{m} \mu(x_i)} \tag{22}$$

$$q_i = \sum_{j=1}^{i} p(x_j) \tag{23}$$

4.5. Crossover Operator

Crossover operation is that in which two chromosomes exchange some genes to form new individuals (offspring). It is the main step of a genetic algorithm to generate new individuals. The purpose is to hope that excellent genes can be inherited into offspring. The crossover process of the grouping genetic algorithm can be stated as follows. We select two parent individuals X and Y using a selection operator, and we randomly select a gene from one of the chromosomes and inject it into the other. This operation may generate duplicate virtual machines in multiple physical machines. If this happens, the physical machines having the duplicate virtual machines will be deleted. The deletion operation may cause some virtual machines to be unassigned to a physical machine. These unallocated virtual machines need to be re-encoded into a physical machine. The entire crossover process is shown in Figures 5–8.

The above method randomly selects genes in individuals to perform crossover operations. This method is blind, causing the genes of excellent individuals to still be difficult to inherit after multiple iterations, reducing the convergence speed. Based on this defect, in our proposed algorithm, we will determine which gene segment will be crossed based on the degree of resources waste of the physical machine and the SLA violation

rate. The power consumption is not considered because the SLA violation rate and power consumption are closely related to the CPU utilization. Therefore, the crossover operator is modified as follows:

1. According to Equations (2) and (4), we calculate the resource waste rate and SLA violation rate of physical nodes.

2. The weighted sum of the two rates is the criterion for selecting the physical machine in the individual. The physical node with the smaller value is selected, and we perform the crossover operation as shown in Figures 6 and 7.

3. If an isolated virtual machine appears in the new generated individual (offspring), it will be reallocated according to the well-known Best-Fit algorithm (BFA). This algorithm searches the entire offspring for free genes on a physical machine and assigns the isolated virtual machine to the smallest gene that fits.

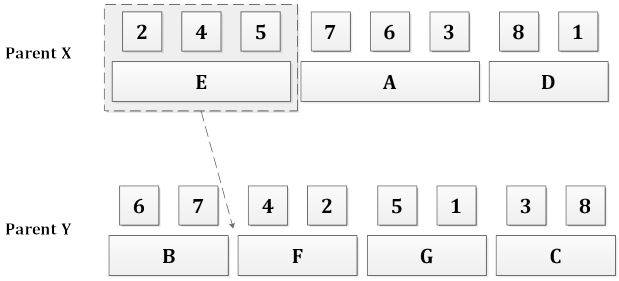

Figure 5. Select randomly a group from X and an intersection on Y.

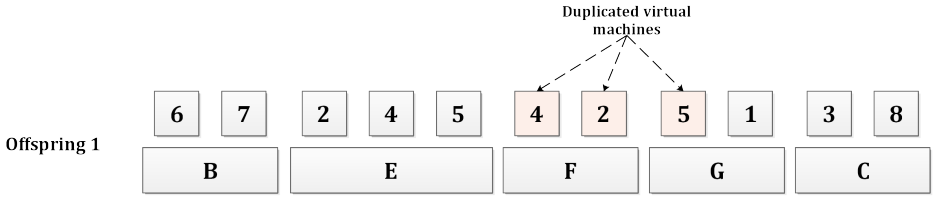

Figure 6. Insert the group E on the selected intersection on Y.

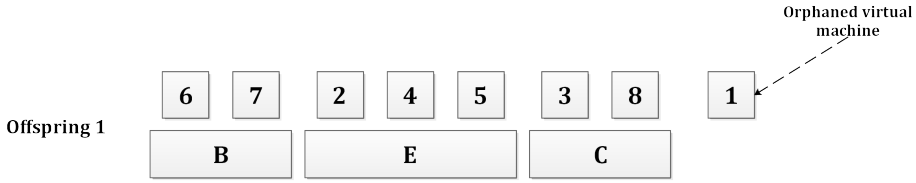

Figure 7. Delete the physical machines containing duplicate virtual machines.

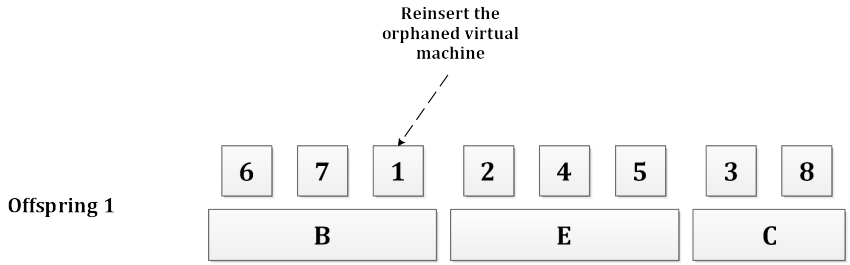

Figure 8. Reinsert the orphaned virtual machine.

4.6. Mutation Operator

The mutation operator in the basic genetic algorithm consists of randomly selecting one or more genes in the individual to mutate in order to avoid falling into a local optimum and to maintain a certain diversity of the population. As mentioned above, this kind of random selection is blind, which makes it impossible to obtain a better solution after multiple inheritances. Therefore, we will select genes with a small evaluation value for

mutation, as in Section 4.5. Unallocated virtual machines are deployed according to the BFA algorithm, which has the advantage of ensuring that the population develops in a good direction.

4.7. FGGA Algorithm Implementation

Our proposed FGGA algorithm implementation is as Algorithm 1.

Algorithm 1 FGGA Algorithm.

Initialize: Virtual machine required resources R; Physical machine resources H; Population size m; Crossover probability p_C; Mutation probability p_M; Number of iterations G; Counter $h_{\max}$; The current best fitness f;
Output: The global best fitness f^*; the global best solution π^*

$h = 0; t = 0; f = 0;$
Pop(0) = Initialize(m); //Randomly generate the initial population (m individuals)
while $t < G$ && $h \neq h_{\max}$ **do**
  for $i = 1$ to m **do**
    Calculate fitness[i] according to Equation (20);
  end for
  All solutions are sorted in descending order of fitness value;
  for $i = 1$ to m **do**
    Apply Selection to Pop(t) //According to Equations (22) and (23);
  end for
  for $i = 1$ to m **do**
    Apply Crossover to Pop(t);
  end for
  for $i = 1$ to m **do**
    Apply Mutation to Pop(t);
  end for
  Get the current best fitness f;
  if $f = f^*$ **then**
    $h++;$
  else
    $f^* = f;$
    $h = 0;$
  end if
  if $h = h_{\max}$ **then**
    return f^*;
    return π^*;
  end if
  $t++;$
end while

5. Experimental Results

In order to verify the effectiveness of our proposed algorithm, we conduct simulation experiments on the CloudSim platform [25]. The data center's electricity consumption, resource waste, and SLA violation rate are evaluated under different algorithms. The algorithm and simulation experiments are coded in JAVA, and the hardware configuration is: CPU 3.2GHz, RAM 8GB, HDD 1TB.

In order to simulate the real data on our considered H-Cloud, the CloudSim simulation platform has been equipped with 100 servers, and the number of virtual machines has been divided into three groups: 80, 200, and 400. The computing power of the server is set to four levels {1500, 2000, 2600, 3000}, the unit is MIPS, and the memory size is {2048 MB, 4096 MB, 8192 MB}.

The computing resource request of the virtual machine is {300, 500, 750, 1500}, the unit is MIPS, and the memory resource request is {256 MB, 512 MB, 2048 MB}. The peak power

of the server is set to 260 W, and the power when idle is 170 W. In the experiment, tasks are set as full-load tasks; that is, task demand is constant, and the data of the data center are simulated for 3 h.

Different values of the parameters of the FGGA algorithm have different effects on its performance. In Section 4, several important parameters that affect our proposed algorithm are introduced: namely, the population size m, the number of iterations G, crossover probability p_C, and mutation probability p_M. After extensive experiments, using the ParamILS framework [26], and under different parameters combinations, we have selected the parameters with the best performance, as shown in Table 1.

Table 1. FGGA algorithm parameters.

Parameter	Value
Population size m	60
Crossover probability p_C	0.8
Mutation probability p_M	0.15

5.1. Single-Objective-Based Experiments

In the experiment, the performance of the FGGA algorithm is compared through two groups of algorithms. First, the single-objective optimization for the three optimization objectives are compared with the FGGA algorithm. For the comparison of power consumption, the single-objective optimization algorithm SO_MEC based on minimum energy consumption and the single-objective optimization algorithm SO_BUCPU based on CPU balanced utilization are selected. The experimental comparison is shown in Figure 9. Since the SLA violation rate and power consumption are closely related to the CPU utilization, and the change trend is close, the comparison of SLA violation rate is the same as the single-objective optimization power consumption comparison algorithm, SO_MEC–SO_BUCPU. The experimental comparison is shown in Figure 10. For the comparison of resource waste, the virtual machine placement algorithm VMPACS [8] based on multi-objective optimization is selected. The experimental comparison is shown in Figure 11.

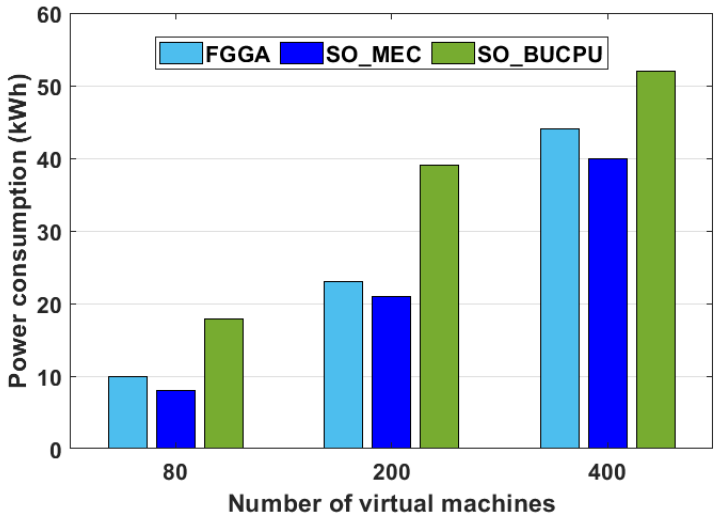

Figure 9. Comparison of power consumption.

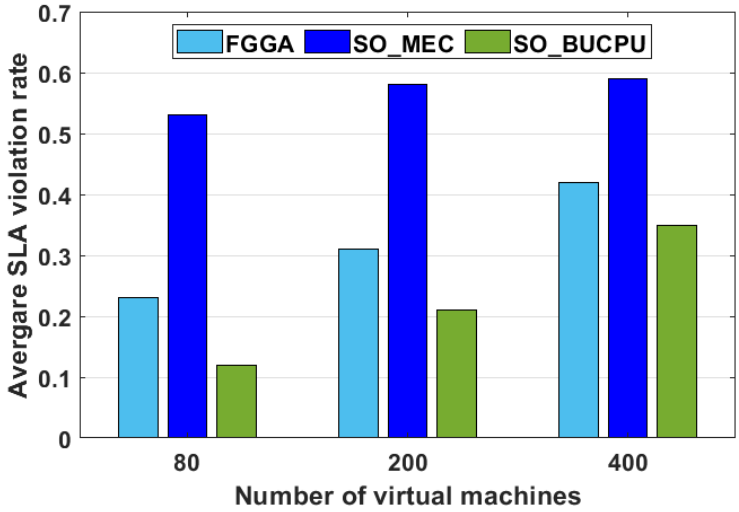

Figure 10. Comparison of average SLA violation rates.

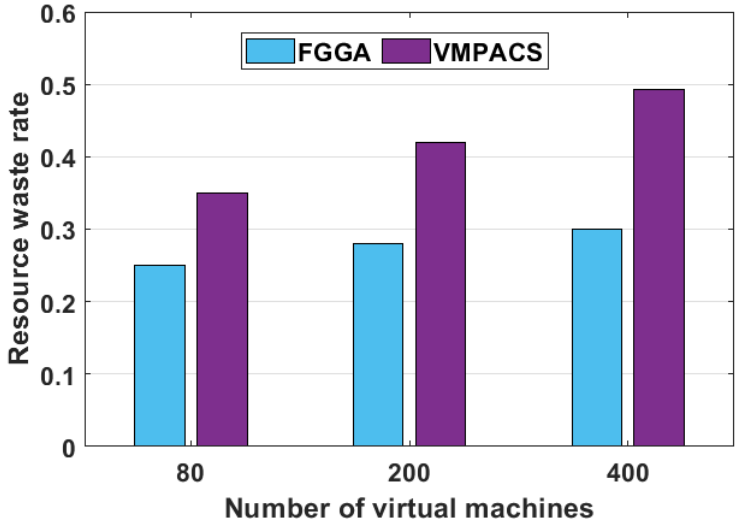

Figure 11. Comparison of average resource waste.

It can be seen from Figure 9 that the power consumption of the SO_MEC algorithm is the smallest, and the power consumption of the SO_BUCPU algorithm is the largest. The FGGA algorithm lies between the two. SO_MEC is optimized with the single goal of electricity consumption costs, the placement of virtual machines is extremely concentrated, and the number of physical machines turned on is small. In the SO_BUCPU algorithm, the CPU is used in a balanced manner. Too many physical nodes are turned on, but the utilization rate is very low, so the performance is the worst. The electricity consumption of the algorithm is not very different from that of SO_MEC.

In evaluating the SLA violation rate, we adopt the average SLA violation rate. The average SLA violation rate is defined as the average of the SLA violation rates of all physical nodes in the running state. The SLA violation rate is calculated by Equation (4). $U_{cpu}^{\max}$ is fixed at 0.9 and f_i fluctuates between (0.1, 0.7). As can be seen from Figure 10, our proposed FGGA algorithm has the best performance, and the SO_MEC algorithm has the worst performance. The reason is that the CPU utilization of each physical node started under this algorithm is very high, and the SO_BUCPU algorithm has the lowest SLA violation rate. The resource utilization rate of physical nodes under the algorithm is low.

It can be seen from Figure 11 that the FGGA algorithm is significantly better than the VMPACS algorithm in terms of reducing resource waste. The reason is that the resource waste of physical nodes has a great relationship with the CPU demand and

memory demand of the virtual machine. When reducing resource waste, the resource requests of virtual machines and the resource utilization of physical nodes should be comprehensively considered.

5.2. Multi-Objective-Based Experiments

In this section, we compare the performance of our proposed algorithm FGGA to the basic grouping genetic algorithm (GGA) and the multi-objective memetic algorithm (MMA) of [5]. Without loss of generality, the GGA algorithm uses linear weighted sums for the three optimization objectives, and the weights of the three optimization objectives are respectively taken as $1/3$. The comprehensive comparison experiment still uses the data of the above experiments for testing. The three sets of algorithms were run 20 times, and the best deployment plan in each set was selected for comparison. The best deployment plans obtained by the three sets of algorithms were recorded as π_{FGGA}, π_{GGA}, and π_{MMA}. In order to comprehensively compare the three optimization goals, we calculate the fitness value of the solution, which is obtained according to Equations (17)–(20). The fitness value is used to weigh the benefits and drawbacks of the solution. The closer the fitness value is to one, the closer the solution is to the ideal state, and the closer it is to zero, the worse the solution.

It can be seen from Figure 12 that the best virtual machine deployment plan obtained by the FGGA algorithm has the largest fitness value. The overall performance of the FGGA algorithm in the multi-objective optimization of power consumption, resource waste, and SLA violation rate is better than the group genetic algorithm GGA and the multi-objective scheduling algorithm MMA. A comprehensive analysis of the single-objective comparison experiment and the multi-objective comparison experiment shows that the FGGA algorithm has no outstanding effect in each single-objective optimization. Its performance is better than other algorithms in the multi-objective simultaneous optimization performance. It shows that our proposed FGGA algorithm performs well in multi-objective optimization.

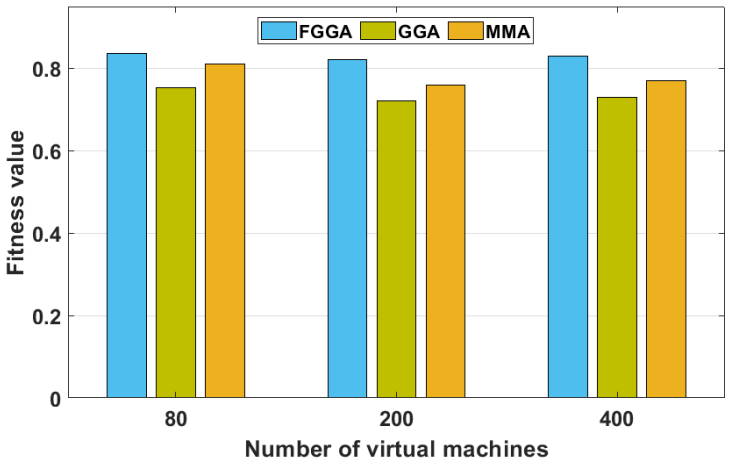

Figure 12. Comparison of fitness value.

6. Conclusions

In this paper, we address a real-world virtual machine placement problem arising in an H-Cloud. Efficient virtual machine placement should consider multiple factors synthetically, including quality of service, energy consumption, and resource utilization. To solve this multi-objective optimization problem, we propose an improved group genetic algorithm. The crossover and mutation operators are improved, and a fuzzy logic-based function is implemented, reducing the decision maker's subjective weight setting to the optimization objectives. The simulation experiments have been conducted on single-objective and

multi-objective optimization experiments. The experimental results demonstrate that our proposed algorithm performs well when optimizing multiple objectives at the same time.

The research on virtual machine placement in this paper is mainly focused on the data center. With the continuous development and growth of cloud computing, the service model of a cross-data center has gradually appeared. Therefore, the next work can consider the virtual machine placement problem of cross-data centers, and we need to consider the network bandwidth of different data centers, their geographic location, and the price of electricity.

Author Contributions: Conceptualization, N.A. and A.A.; methodology, M.A.R. and N.A.; validation, A.A., M.A.R. and N.A.; formal analysis, N.A., A.A. and M.A.R.; investigation, N.A., A.A. and M.A.R.; resources, N.A., A.A. and M.A.R.; experiments, N.A., A.A. and M.A.R.; writing—original draft preparation, N.A., A.A. and M.A.R.; writing—review and editing, N.A., A.A. and M.A.R.; visualization, N.A., A.A. and M.A.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used and analyzed during the current study are available from the corresponding author on reasonable request.

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

Abbreviations

The following abbreviations are used in this manuscript:

BFA	Best-Fit Algorithm
CPU	Central Processing Unit
FGGA	Fuzzy Grouping Genetic Algorithm
GGA	Grouping Genetic Algorithm
H-Cloud	Healthcare Cloud
MIPS	Million Instructions Per Second
MMA	Multi-objective Memtic Algorithm
NAS	Network-Attached Storage
SLA	Service-Level Agreement
SO_MEC	Single-Objective based on Minimum Energy Consumption
SO_BUCPU	Single-Objective based on CPU
VMPACS	Virtual Machine Placement problem based on Ant Colony System

References

1. Whitney, J.; Delforge, P. *Data Center Efficiency Assessment, Scaling Up Energy Efficiency Across the Data Center Industry: Evaluating Key Drivers and Barriers*; Technical Report; Natural Resources Defense Council: New York, NY, USA, 2014.
2. Shehabi, A.; Smith, S.; Sartor, D.; Brown, R.; Herrlin, M.; Koomey, J.; Masanet, E.; Horner, N.; Azevedo, I.; Lintner, W. *United States Data Center Energy Usage Report*; Technical Report; Berkeley Lab.: Berkeley, CA, USA, 2014.
3. Liu, X.F.; Zhan, Z.H.; Du, K.J.; Chen, W.N. Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization approach. In Proceedings of the 2014 Genetic and Evolutionary Computation Conference, Vancouver, BC, Canada, 12–16 July 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 41–47. [CrossRef]
4. Le, K.; Zhang, J.; Meng, J.; Bianchini, R.; Jaluria, Y.; Nguyen, T.D. Reducing electricity cost through virtual machine placement in high performance computing clouds. In Proceedings of the 2011 SC—International Conference for High Performance Computing, Networking, Storage and Analysis, Seatle, WA, USA, 12–18 November 2011. [CrossRef]
5. Pires, F.L.; Barán, B. Multi-objective virtual machine placement with service level agreement: A memetic algorithm approach. In Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, UCC 2013, Dresden, Germany, 9–12 December 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 203–210. [CrossRef]
6. Zhiqiang, R.; Haibo, L. Dynamic virtual machine integration algorithm based on load prediction. *J. Yangtze Univ. Nat. Ed.* **2015**, *12*, 29–33.

7. Liao, X.; Jin, H.; Liu, H. Towards a green cluster through dynamic remapping of virtual machines. *Future Gener. Comput. Syst.* **2012**, *28*, 469–477. [CrossRef]
8. Gao, Y.; Guan, H.; Qi, Z.; Hou, Y.; Liu, L. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *J. Comput. Syst. Sci.* **2013**, *79*, 1230–1242. [CrossRef]
9. Mosa, A.; Paton, N.W. Optimizing virtual machine placement for energy and SLA in clouds using utility functions. *J. Cloud Comput.* **2016**, *5*, 17. [CrossRef]
10. Kumar, D.; Raza, Z. A PSO based VM resource scheduling model for cloud computing. In Proceedings of the 2015 IEEE International Conference on Computational Intelligence and Communication Technology, CICT 2015, Ghaziabad, India, 13–14 February 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 213–219. [CrossRef]
11. Farzai, S.; Shirvani, M.H.; Rabbani, M. Multi-objective communication-aware optimization for virtual machine placement in cloud datacenters. *Sustain. Comput. Inform. Syst.* **2020**, *28*, 100374. [CrossRef]
12. Gopu, A.; Venkataraman, N. Optimal VM placement in distributed cloud environment using MOEA/D. *Soft Comput.* **2019**, *23*, 11277–11296. [CrossRef]
13. Alhammadi, A.S.A.; Vasanthi, V. Multi-Objective Algorithms for Virtual Machine Selection and Placement in Cloud Data Center. In Proceedings of the 2021 International Congress of Advanced Technology and Engineering, ICOTEN 2021, Online, 4–5 July 2021; IEEE: Piscataway, NJ, USA, 2021. [CrossRef]
14. Pushpa, R.; Siddappa, M. Adaptive Hybrid Optimization Based Virtual Machine Placement in Cloud Computing. In Proceedings of the 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), Namakkal, India, 12–13 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–9. [CrossRef]
15. Lu, J.; Zhao, W.; Zhu, H.; Li, J.; Cheng, Z.; Xiao, G. Optimal machine placement based on improved genetic algorithm in cloud computing. *J. Supercomput.* **2022**, *78*, 3448–3476. [CrossRef]
16. Mejahed, S.; Elshrkawey, M. A multi-objective algorithm for virtual machine placement in cloud environments using a hybrid of particle swarm optimization and flower pollination optimization. *PeerJ. Comput. Sci.* **2022**, *8*, e834. [CrossRef] [PubMed]
17. Li, J.; Yang, S.; Wang, J.; Moorthy, C.R.S.; Fareentaj, U.; Divya, T.; Shi, L.; Lu, S.; Feng, T. Research on Multi-Objective Optimization Method of Edge Cloud Computing Virtual Machine Placement. *J. Phys. Conf. Ser.* **2022**, *2195*, 012012. [CrossRef]
18. Asghari, S.; Navimipour, N.J. The role of an ant colony optimisation algorithm in solving the major issues of the cloud computing. *J. Exp. Theor. Artif. Intell.* **2021** , *2021*, 1–36. [CrossRef]
19. Garey, M.R.; Johnson, D.S. *Approximation Algorithms for Bin Packing Problems: A Survey*; Springer: Vienna, Austria, 1981; pp. 147–172._8. [CrossRef]
20. Barroso, L.A.; Hölzle, U. The case for energy-proportional computing. *Computer* **2007**, *40*, 33–37. [CrossRef]
21. Li, H.; Li, T.; Shuhua, Z. Energy-performance optimisation for the dynamic consolidation of virtual machines in cloud computing. *Int. J. Serv. Oper. Inform.* **2018**, *9*, 62–82. [CrossRef]
22. Mutingi, M.; Mbohwa, C.; Musiyarira, H. Grouping genetic algorithms: An exploratory study. In Proceedings of the World Congress on Engineering and Computer Science (WCECS 2017), San Francisco, CA, USA, 25–27 October 2017; pp. 1–9.
23. Ramos-Figueroa, O.; Quiroz-Castellanos, M.; Mezura-Montes, E.; Kharel, R. Variation Operators for Grouping Genetic Algorithms: A Review. *Swarm Evol. Comput.* **2021**, *60*, 100796. [CrossRef]
24. Klir, G.J.; Yuan, B. (Eds.) *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi A. Zadeh*; World Scientific Publishing Co., Inc.: River Edge, NJ, USA, 1996.
25. 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 and Simulation, HPCS 2009, Leipzig, Germany, 21–24 June 2009; pp. 1–11. [CrossRef]
26. Hutter, F.; Hoos, H.H.; Leyton-Brown, K.; Stützle, T. ParamILS: An automatic algorithm configuration framework. *J. Artif. Intell. Res.* **2009**, *36*, 267–306. [CrossRef]