energies

Article

An Energy Aware Unified Ant Colony System for Dynamic Virtual Machine Placement in Cloud Computing

Xiao-Fang Liu [1,2], Zhi-Hui Zhan [2,*] and Jun Zhang [2,*]

1 Department of Computer Science, Sun Yat-sen University, Guangzhou 510006, China; xfliu@163.com
2 School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
* Correspondence: zhanapollo@163.com (Z.-H.Z.); junzhang@ieee.org (J.Z.)

Academic Editor: Yun Li
Received: 31 December 2016; Accepted: 17 February 2017; Published: 1 May 2017

Abstract: Energy efficiency is a significant topic in cloud computing. Dynamic consolidation of virtual machines (VMs) with live migration is an important method to reduce energy consumption. However, frequent VM live migration may cause a downtime of service. Therefore, the energy save and VM migration are two conflict objectives. In order to efficiently solve the dynamic VM consolidation, the dynamic VM placement (DVMP) problem is formed as a multiobjective problem in this paper. The goal of DVMP is to find a placement solution that uses the fewest servers to host the VMs, including two typical dynamic conditions of the assignment of new coming VMs and the re-allocation of existing VMs. Therefore, we propose a unified algorithm based on an ant colony system (ACS), termed the unified ACS (UACS), that works on both conditions. The UACS firstly uses sufficient servers to host the VMs and then gradually reduces the number of servers. With each especial number of servers, the UACS tries to find feasible solutions with the fewest VM migrations. Herein, a dynamic pheromone deposition method and a special heuristic information strategy are also designed to reduce the number of VM migrations. Therefore, the feasible solutions under different numbers of servers cover the Pareto front of the multiobjective space. Experiments with large-scale random workloads and real workload traces are conducted to evaluate the performance of the UACS. Compared with traditional heuristic, probabilistic, and other ACS based algorithms, the proposed UACS presents competitive performance in terms of energy consumption, the number of VM migrations, and maintaining quality of services (QoS) requirements.

Keywords: dynamic virtual machine placement (DVMP); ant colony system (ACS); energy saving; cloud computing

1. Introduction

Cloud computing is a large-scale distributed computing paradigm in which customers are able to access elastic resources on demand over the Internet by a pay-as-you-go principle [1–3]. Cloud computing facilitates services at three different levels, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), and four deployment models, including private cloud, community cloud, public cloud, and hybrid cloud [4]. With the increasing size of storage requirements, the inefficient use of resources causes high energy consumption [5,6]. Reports show that energy consumption has occupied a significant proportion of the total cost of data centers [7]. Energy efficiency is becoming a challenge in data center management [8–11]. In cloud computing, virtualization is adopted for abstraction and encapsulation such that the underlying infrastructure can be unified as a pool of resources and multiple applications can be executed within isolated virtual machines (VMs) on the same server simultaneously [12]. This allows the consolidation of VMs on servers and provides an opportunity for energy saving, since an active but idle server often costs much more in terms of power consumption, using between 50% and 70% of the power of a fully utilized server [13]. However, the servers in data center utilize 10% to 50% most of the time and rarely reach 100% [14]. The low utilization is often caused by the over-provisioning of resources since the data center is designed to support the peak load [15].

When a request comes, the data center needs to create a VM and decides which server to place. However, the VM workload often experiences variability during the running time [16]. There are two commonly used strategies; static allocation and dynamic allocation. Static resource allocation adopts the planned assignment unless an exception occurs. Provisioning for the peak workload is simple but inefficient and leads to a low resource utilization. To reduce resource waste, Speikamp et al. [17] optimized the assignment based on daily workload cycles. Setzer et al. [18] also modeled the problem as a capacity constraint optimization and adopted a mathematical method to consolidate the VMs on a minimum number of servers. To increase the resource utilization, some researchers propose resource reservation methods. The resource demand pattern of each VM is modeled, and then a consolidation algorithm based on the stationary distribution of the VM patterns is adopted for assignment. In [19], a Markov chain is used to capture the burstiness of workload firstly, and then the VMs are allocated based on their workload patterns. All these static resource allocation methods assume that the workload patterns of servers are known and the VM set is stable. However, the VMs usually present quite different resource demands and execution times [20]. The static allocation methods may not work well in these cases.

In recent years, the dynamic allocation of VMs with live migration has received significant attention. The VMs are firstly assigned to servers according to a normal workload, and then migration procedure is performed when servers are overloaded or underutilized. The idle servers can respond to the incremental workload and can also be switched to sleep mode. Thus, the VMs are assigned to the minimum active servers to reduce energy consumption and meanwhile meet QoS requirements. Different from static resource allocation, dynamic resource allocation methods do not need planning. The number of active servers is adjusted dynamically. The threshold-based method is simple and commonly used. In commercial and open-source approaches such as VMware's Distributed Resource Management [21], the management performs VM migrations when the resource utilization violates the predefined threshold. However, setting static thresholds is not efficient for non-stationary resource usage patterns. Beloglazov et al. [22] proposed an adaptive threshold method based on the historical data and adopts a modified best-fit decreasing (BFD) method to dynamically allocate VMs. In order to reduce the number of unnecessary migrations, server loads, or VM resource requirements, a prediction method is introduced. The workloads of different applications are classified using k-means in [23,24]. Zhang et al. [25] considered the heterogeneity of the workload and implemented resource prediction according to application characteristics. The VMs are allocated according to a standard convex optimization method and a container-based heuristic method. Recently, Verma et al. [26] proposed a dynamic resource prediction method by feature extraction from users' data and allocated the VMs by a best-fit decreasing algorithm. In addition to the above deterministic algorithms for VM assignment, probabilistic methods are also proposed to solve large-scale problem with thousands of servers. In [15], Mastroianni et al. proposed a probabilistic consolidation method, ecoCloud, to implement VM assignment and migration according to the local information of each server.

In addition to traditional heuristic and probabilistic methods, evolutionary algorithms have also been widely used in DVMP due to their good performance on complex optimization problems [27–29]. In [30], Mi et al. proposed a genetic algorithm based approach to adaptively adjust the VMs according to time-varying requirements. In [31], Ashraf et al. employed ant colony optimization (ACO) to find a VM migration plan to consolidate VMs on underutilized servers. In [32], Farahnakian assigned the VMs by a best-fit algorithm first and then performed an ant colony system for VM migration when observing workload variation. A VM migration is allowed only if it is able to reduce the energy consumption and the number of VM migrations. Thus, it may not work well on the situations where many VMs present short-term bursts and a large number of servers are overloaded.

The approaches mentioned above all deal with the VM assignment and VM re-allocation separately. However, these methods are not efficient in cases when new application requests and migrations occur simultaneously. Moreover, the VM re-allocation procedure is to find a new server in which to place the corresponding VM. Thus, we can use a unified algorithm for the assignment and re-allocation of VMs. In this paper, we formulate the dynamic virtual machine placement (DVMP) as a multiobjective combinatorial optimization problem and design an online algorithm and an ant colony system (ACS) based unified algorithm, termed UACS, to consolidate VMs for saving energy and reducing the number of migrations while meeting QoS requirements simultaneously. This is a further extension of our early work on VM assignments at a certain time in [33]. In UACS, multiple ants construct solutions concurrently with the guidance of a pheromone and heuristic information. A dynamic pheromone deposition method is designed to support the dynamic VM assignment and migration. In order to reduce the number of VM migrations, heuristic information is also introduced to help the solution construction. The QoS requirements are formalized via Service Level Agreements (SLA). Experiments with large scale random workload or real workload traces are carried out to evaluate the performance. Not only the traditional heuristic approach, BFD [22], and the probabilistic method, ecoCloud [15], but also a recently proposed ACS-based algorithm, ACS-VM [32], are implemented for comparison. Compared with the traditional heuristic, probabilistic, and other evolutionary algorithms, the proposed UACS achieves competitive performance in terms of energy consumption, the number of VM migrations, and the number of SLA violations.

The remainder of this paper is organized as follows. Section 2 describes the DVMP problem, performance metrics, and the ant colony system in detail. Section 3 develops the proposed algorithm, UACS. Experiments are undertaken to evaluate the performance of the UACS in Section 4. Finally, the conclusions are drawn in Section 5.

2. Model

2.1. DVMP Problem

Elasticity is one of the most important characteristics of cloud computing. Once the cloud data center receives an application request from a customer, a VM is created to host the application. Then the VM is assigned to one available server according to the placement strategy. With the time-varying resource requirement, the original assignment may violate the QoS requirements. Some servers are overloaded and some servers are underutilized. This triggers VM migration. However, live migration may cause the downtime of the service. Thus, minimizing the number of migrations is another objective to improve the QoS requirments. How to assign the VMs to reduce energy consumption while using as few as migrations as possible and meeting QoS becomes a challenge. Since the two objectives in terms of energy consumption and the number of migrations conflict with each other, the DVMP is modeled as a multiobjective optimization, minimizing energy consumption and the number of migrations while meeting QoS requirements. The time is split into intervals t = [1, 2, ..., T]. In interval t, we assume that there is an available server set $P_t = \{1, 2, \ldots, M_t\}$ with a size of M_t, the VM set that has already been assigned to servers $A_t = \{1, 2, \ldots, H_t\}$ with a size of H_t, and a new incoming VM set $V_t = \{1, 2, ..., N_t\}$ with size of N_t. The assigned VMs in A_t are allowed to be migrated to other servers and the new incoming VMs in V_t need to be allocated to servers. Two resources, CPU and RAM, are considered in this paper. In interval t, the CPU and RAM requirements of each VM j are denoted as vc_{jt} and vm_{jt}. The corresponding resource capacities of each server i for CPU and RAM in interval t are denoted as PC_{it} and PM_{it}, respectively. To ensure the performance of user applications, the maximum resource utilization is defined as R. The peak workload of each VM is assumed to be less than the maximum amount of resources provided by each server. For a placement solution S_t in interval t, a zero-one adjacency matrix X_t is used to represent the assignment relationship between the

VMs and the servers. In X_t, the element x_{ijt} is used to describe whether VM j is assigned to server i. The DVMP problem for minimizing energy E and the number of VM migrations L is formulated as:

$$\text{Minimize } F(S_t) = (E(S_t), L(S_t)) \tag{1}$$

subject to:

$$x_{ijt} = \begin{cases} 1, & \text{if server } i \text{ hosts the VM } j \text{ at interval } t \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in P_t \text{ and } \forall j \in (V_t \cup A_t) \tag{2}$$

$$y_{it} = \begin{cases} 1, & \text{if } \sum_{j \in V_t} x_{ijt} + \sum_{k \in A_t} x_{ikt} \geq 1 \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in P_t \tag{3}$$

$$\sum_{i \in P_t} x_{ijt} = 1 \quad \forall j \in (V_t \cup A_t) \tag{4}$$

$$\sum_{j \in V_t} vc_{jt} \cdot x_{ijt} + \sum_{k \in A_t} vc_{kt} \cdot x_{ikt} \leq R \cdot PC_{it} \cdot y_{it}, \quad \forall i \in P_t \tag{5}$$

$$\sum_{j \in V_t} vm_{jt} \cdot x_{ijt} + \sum_{k \in A_t} vm_{kt} \cdot x_{ikt} \leq R \cdot PM_{it} \cdot y_{it}, \quad \forall i \in P_t \tag{6}$$

The value of x_{ijt} is set as 1 if VM j is assigned to server i, as in Equation (2). In Equation (3), y_{it} presents whether there are any VMs assigned to server i in interval t. A VM is allowed to be assigned to one and only one server, as shown in Equation (4). Each server must satisfy the resource requirements of VMs on it, as with constraints (5) and (6). The energy consumption E and number of migrations L will be described in detail in the following section.

2.2. Performance Metric

2.2.1. SLA Violation

In data center, it is important to meet QoS requirements. QoS requirements are often defined by SLAs, such as minimum latency or maximum response time [5]. In this paper, we adopt the workload independent metric defined in [5] to evaluate the SLA violation level. The metric calculates the percentage of time that the CPU utilization of the active servers appeared larger or equal to 100% and is defined as SLAO (where O represents server overload) as follows:

$$SLAO = \frac{1}{N} \sum_{i=1}^{N} \frac{T_{u_i}}{T_{ai}} \tag{7}$$

where N is the number of active servers during the runtime, T_{ui} is the total time that the CPU utilization of server i reaches 100%, and T_{ai} is the total active time of server i.

2.2.2. Number of Migrations

Migration may cause the performance degradation of VMs and a downtime of services. The time used for one migration is determined by the memory of the VM and the network bandwidth between the source and destination server [32]. Thus, we calculate the number of migrations during the runtime for evaluation.

2.2.3. Energy Consumption

A server in an idle state consumes a large amount of power, which is 50–70% of its maximum power consumption [8]. As the CPU utilization increases, the power consumption of a server grows approximately linearly [34]. Therefore, similar to our previous study in static environments [33], we still define the power model as:

$$P(u) = r_{idle} \cdot P_{\max} + (1 - r_{idle}) \cdot P_{\max} \cdot u \tag{8}$$

where r_{idle} is the ratio of power consumption in an idle server, $P_{\max}$ is the maximum power consumption of the server, and u $(0 \leq u \leq 1)$ is the CPU utilization of the server. According to (8), we can see that, by reducing the number of active servers, it is possible to reduce the energy consumption.

2.3. ACS

Inspired by the foraging behavior of ants, Dorigo and Gambardella proposed ACS firstly in 1997 for solving the traveling salesman problem (TSP) [35]. In ACS, multiple ants construct solutions with the guidance of a pheromone and heuristic information in parallel. The pheromone is deposited between city pairs to record the historical search experience. The heuristic information embedded with a greedy selection based on the current state helps to find better solutions. In each generation, each ant constructs a route by visiting the cities step by step. During the solution construction, pheromone local updating is performed to evaporate the pheromone on the visited routes and diversify the solutions. After all the ants have finished construction, the best solution is selected and used for pheromone global updating to record the good solution. The algorithm terminates when the maximum generation is reached.

3. Method

Due to the good performance of ACS on the static VMP problem presented in our previous work [33], we developed an ACS based unified algorithm, UACS, for the DVMP. Different from the VMP with fixed resource requirements considered in [33], the VMs with different arrival times, durations, and time-varying resource requirements are tracked in the DVMP in this paper. The UACS deals with the VM migration as a VM assignment problem. That is to say, the UACS can deal with not only the new incoming VM assignment but also determine the VM migration plan. UACS is applied periodically to assign new VMs and adaptively optimize the VM placement according to the workload. The new unassigned VMs and the VMs on overloaded and underutilized servers are collected together firstly. Then, the UACS is performed to obtain nondominated assignments stored in an archive. Finally, we select one according to the preference of decision makers from the archive to execute the assignment. The detail of the algorithm is described as follows. We take time t as an example.

3.1. VMs Set for Assignment

At time t, assume that new incoming VMs arrive and are collected in set NV_t. Define the size of NV_t as $|NV_t|$. The VMs on overloaded servers are collected together into a set OV_t. To save energy. the VMs on underutilized servers can be migrated. In order to reduce the number of unnecessary migrations, some underutilized servers are reserved for the assignment of new VMs and VM migrations from overloaded servers. The sum of total resource requirements of new VMs and total excess resource requirements of overloaded servers are estimated and defined as TR. Then the remaining resources of underutilized servers are accumulated gradually in order until the value reaches $2TR$. Then the VMs on the remaining underutilized servers are collected into a set UV_t. The VMs in these three sets, NV_t, OV_t, and UV_t, are combined together into a set V_t. The size of V_t is denoted as N_t. The V_t set is the VM input of the UACS algorithm. The other running VMs remain on their original assigned servers. The pseudo code is presented in Algorithm 1.

Algorithm 1. Construct VMs set for assignment

```
1.  $TR_{\text{cpu}} \leftarrow 0$, $TR_{\text{memo}} \leftarrow 0$; $UR_{\text{cpu}} \leftarrow 0$; $UR_{\text{memo}} \leftarrow 0$; $NV_t \leftarrow \phi$; $OV_t \leftarrow \phi$; $UV_t \leftarrow \phi$
2.  for each new coming VM $j$ do
3.      $TR_{\text{cpu}} \leftarrow TR_{\text{cpu}} + vc_j$
4.      $TR_{\text{memo}} \leftarrow TR_{\text{memo}} + vm_j$
5.      add VM into $NV_t$
6.  end for
7.  for each overloaded server $i$ do
8.      $TR_{\text{cpu}} \leftarrow TR_{\text{cpu}} + |PC_i - UC_i|$
9.      $TR_{\text{memo}} \leftarrow TR_{\text{memo}} + |PM_i - UM_i|$
10.     add VMs on server $i$ into $OV_t$
11. end for
12. for each underutilized server $i$ do
13.     $UR_{\text{cpu}} \leftarrow UR_{\text{cpu}} + |PC_i - UC_i|$
14.     $UR_{\text{memo}} \leftarrow UR_{\text{memo}} + |PM_i - UM_i|$
15.     if $UR_{\text{cpu}} \geq 2 \times TR_{\text{cpu}}$ && $UR_{\text{memo}} \geq 2 \times TR_{\text{memo}}$ then
16.         sleepindex $\leftarrow i$;
17.         go to line 19
18.     end if
19. end for
20. for each underutilized server $k$ behind server *sleepindex* do
21.     add the VMs on server $k$ into $UV_t$
22. end for
```

3.2. Dynamic Pheromone Deposition

The pheromone in UACS is deposited on the VM pairs. The pheromone between the new incoming VM and any other one is deposited with a value of the initial pheromone $\tau_0 = 1/M_t$, where M_t is the number of available servers in the data center at time t. Since the VMs in OV_t and UV_t have an original assigned server in the last time $t-1$, these VM pairs inherit their pheromone in last time $t-1$ to make them have a higher opportunity to be grouped together. Thus, the number of migrations may be reduced. In a special case, some VMs in OV_t and UV_t were not included for assignment at time $t-1$; the pheromone related to them is set as the initial pheromone τ_0. In the following, the pheromone between VM j and k is denoted by $\tau(k, j)$.

3.3. UACS

Pareto-based [36] and decomposition via aggregation [37] methods are often used in multiobjective problems. In Pareto-based approaches, the Pareto dominance pushes the population toward the Pareto front (PF) as shown in Figure 1. As illustrated in Figure 1b, decomposition via aggregation decomposes a multiobjective problem into a set of scalar optimization subproblems and optimizes them simultaneously. However, the setting of weights is a difficult problem especially when the two objectives are not comparable [38]. Thus, in this paper, we adopt a different method with an external archive to solve the problem, as shown in Figure 1c. The curve represents the PF, where the solutions are nondominated by any others. The horizontal line with an arrow represents the optimization direction of the number of migrations. The vertical line represents the optimization direction of energy consumption. The circles with different fill patterns are solutions in different generations. In each generation, certain number of servers are given, and we find feasible solutions within these servers. Based on the power model defined in Section 2, the optimization of a number of active servers coincides with the energy consumption. By the nondominated sorting of the obtained solutions, only the solutions with minimum energy consumption and migrations are preserved. For example, Figure 1c shows 12 solutions; four are hollow circles, four are circles with points, and four are circles with grids.

At first, only the four solutions denoted as hollow circles are found, but only the two solutions closed to the PF are stored in the archive, as grouped by the ellipse. Then the minimum servers among these two nondominated solutions are denoted as $M^{\min}$. In the next generation, $M^{\min} - 1$ servers are given, and UACS tries to find feasible solutions. Assume that the four solutions denoted as circles with points are obtained in this generation. However, only two of them are nondominated solutions and are added into the archive, also grouped by the ellipse. Similar search schema are performed in the following generations. For example, assume that the four solutions denoted as circles with grids are found, and three of them are nondominated solutions and are stored in the archive, as grouped by the ellipse. Therefore, as the number of generations increases, the number of provided servers decreases, and the computation concentrates on the unexplored areas. The optimization of the number of migrations is implemented by the design of pheromone, heuristic information, and nondominated sortation. Through multiple iterations, we can find solutions along the PF. Since $M^{\min}$ is our optimization objective, which is unknown in advance, we begin with $M^{\min} = M_t + 1$ in the initialization state, where M_t is the number of available servers in the data center at time t. In order to simplify the symbol representation, we use the vc_j and vm_j to represent the CPU and RAM requirements at time t.

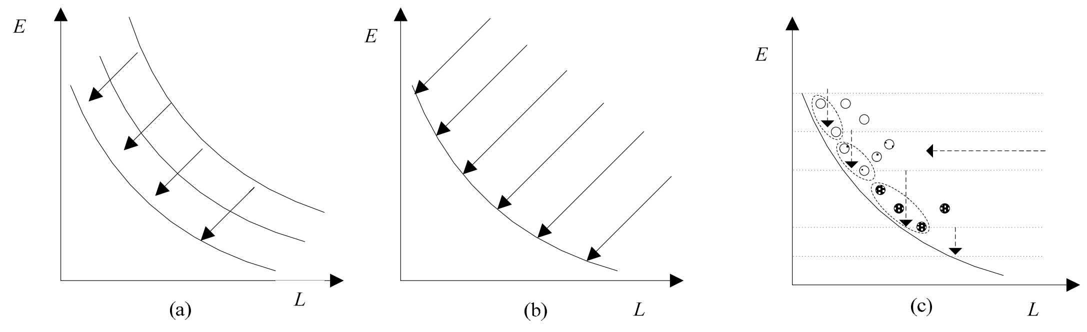

Figure 1. The search schemas in methods (**a**) Pareto-based; (**b**) Decomposition via aggregation; (**c**) UACS.

3.3.1. Solution Construction

After the initialization, UACS goes to construct solutions iteration by iteration so as to find better feasible solutions with fewer servers. In each iteration g ($g \geq 1$), UACS aims to find a feasible solution with one server less than $M^{\min}$. Therefore, m ants try to place the N_t VMs to the $M_g = M^{\min} - 1$ servers. The ant constructs a solution with N_t steps by selecting a server to assign to each VM j ($j \in V_t$), according to the pheromone and heuristic information. The servers with enough remaining resources for placing VM j are added into a set C_j, while the other available servers are added into a set D_j. To get a feasible solution, we consider the servers in C_j firstly and then in D_j only when there are no available servers with enough remaining resources. Since the pheromone is deposited between VM pairs, the preference $Pre(i, j)$ of VM j for server i is calculated as:

$$Pre(i,j) = \begin{cases} \frac{1}{|s_i|} \sum\limits_{k \in s_i} \tau(k,j), & \text{if } |s_i| \neq 0 \\ \tau_0, & \text{otherwise} \end{cases} \tag{9}$$

where s_i is the existing VM set on server i and $|s_i|$ is the number of VMs deployed on server i.

The heuristic information for each server is calculated as

$$\eta(i,j) = \begin{cases} \dfrac{1.0}{\left|\frac{PC_i - UC_i - vc_j}{PC_i}\right| + \left|\frac{PM_i - UM_i - vm_j}{PM_i}\right| + 1.0}, & \text{if } i \in C_j \\ 2 - \left|\dfrac{PC_i - UC_i - vc_j}{PC_i}\right| - \left|\dfrac{PM_i - UM_i - vm_j}{PM_i}\right|, & \text{if } i \in D_j \end{cases} \tag{10}$$

where UC_i and UM_i represent the usage of CPU and the memory of server i before joining VM j. In order to reduce the number of migrations, the reassigned VMs tend to select the original servers for placement. In order to enhance the probability of selecting the original server, the heuristic information $\eta(i, j)$ of the VM j and its corresponding original server i are set as five times that of (10).

With the design of the pheromone and heuristic information, the probability for assigning an unassigned VM j to server i from a set I_j ($I_j = C_j$ if the $C_j \neq \phi$; otherwise $I_j = D_j$) is calculated by:

$$p(i,j) = \frac{Pre(i,j)\eta(i,j)^{\beta}}{\sum\limits_{k\in I_j} Pre(k,j)\eta(k,j)^{\beta}}, \quad \forall i \in I_j \tag{11}$$

where β ($\beta > 0$) is a predefined parameter that controls the relative importance of heuristic information. For VM j, it chooses server i from the servers set I_j by applying the state transition rule given by:

$$i = \begin{cases} \underset{k\in I_j}{\text{argmax}}\, Pre(k,j)\eta(k,j)^{\beta}, & \text{if } q \leq q_0 \\ I, & \text{otherwise} \end{cases} \tag{12}$$

where q is a random number in range of [0, 1], I is a random integer selected from I_j by a roulette wheel selection according to the probability distribution in (11), and q_0 is a predefined parameter ($0 \leq q_0 \leq 1$) controlling the exploitation and exploration behaviors of the ant.

After all the ants have finished solution construction, it is firstly checked whether the solutions are feasible. For the infeasible solutions, a local search by exchanging and moving the VMs, proposed in [33], is performed to adjust the VM placement. Note that the local search herein is performed on the obtained solutions not the current placement in data center, and it does not lead to real cost except for time. If the solution is still infeasible after a local search, we provide more servers and adjust the VMs on the overloaded servers to the new servers by a first-fit algorithm [39] until it becomes feasible. The first-fit algorithm assigns the VMs to the first server with enough remaining resources in order.

3.3.2. Solution Preservation

The solutions are evaluated on two objectives, power consumption E and the number of migrations L. We use an archive A with fixed size NA to store the nondominated solutions obtained until the current generation. The current size of A is denoted as na. If na exceeds NA, solutions from A ($na - NA$) are randomly discarded except for the solutions with minimum power consumption or with minimum migrations.

3.3.3. Pheromone Update

During the solution construction process, a local pheromone update is performed to evaporate the pheromone on visited VM pairs and avoid similarity between solutions. On each VM-pair (k, j) on the same server, the pheromone is updated by:

$$\tau(k,j) = (1-\rho)\cdot\tau(k,j) + \rho\cdot\tau_0 \tag{13}$$

where $0 < \rho < 1$ is the pheromone decay parameter.

At the end of each generation, global pheromone updating is performed to reinforce the good assignment. Since UACS aims to find a feasible solution with a given server number in each generation, we select the one with minimum power consumption S^p from archive A to perform global pheromone updating. For each VM-pair on the same server in S^p, the pheromone is updated as:

$$\tau(k,j) = (1-\varepsilon)\cdot\tau(k,j) + \varepsilon\cdot\Delta\tau_i, \text{ if } (k,j) \in s_i, \forall s_i \in S^p \tag{14}$$

$$\Delta\tau_i = \frac{1}{M^{\min}} + \frac{1}{LC_i + LM_i + 1} \tag{15}$$

where ε ($0 < \varepsilon < 1$) is the parameter to control the degree of pheromone enhancement, s_i is the VM set on server i, and LC_i and LM_i are the corresponding normalized remaining CPU and RAM on server i (the ratio of remaining resource to the resource capacity).

UACS terminates when the maximum predefined generation is reached. The decision makers can select a solution from archive according to their preference for VM assignment and migration. In this paper, we focus more on energy saving, thus the solution with the minimum energy consumption firstly and minimum migrations secondly is selected from the archive to implement assignment and migration. The process of UACS contains the 'Construct VMs set for assignment' procedure as in Algorithm 1 and the 'Initialize Pheromone Deposition' procedure, as in Algorithm 2. The pseudo-code of UACS is given in Algorithm 3.

Algorithm 2. Initialize Pheromone Deposition

```
1.  $\tau_0 \leftarrow 1/M_t$
2.  for each VM pair $(k, j)$
3.    if VM $k$ and VM $j$ are included in the assignment in the last time then
4.      $\tau(k, j) \leftarrow \tau_{old}(k, j)$
5.    else
6.      $\tau(k, j) \leftarrow \tau_0$
7.    end if
8.  end for
```

Algorithm 3. UACS

```
1.  Construct VM sets for assignment following Algorithm 1
2.  Initialize pheromone $\tau$ as Algorithm 2
3.  $g \leftarrow 1, M^{\min} \leftarrow M_t + 1, A \leftarrow \phi$
4.  while $g \leq G_{\max}$ do
5.    $M_g \leftarrow M^{\min} - 1$
6.    for each ant $m$ do
7.      $S_m \leftarrow \phi$
8.      for each VM $j$ do
9.        $C_j \leftarrow \phi; D_j \leftarrow \phi$
10.       for each server $i$ do
11.         if there is enough remaining resources to place VM $j$ then
12.           add $i$ into $C_j$
13.         else
14.           add $i$ into $D_j$
15.           end if
16.         calculate $Pre(i, j)$ by using (9)
17.         calculate $\eta(i, j)$ by using (10)
18.         if VM $j$ is assigned to server $i$ in the last assignment then
19.           $\eta(i, j) \leftarrow 5 \times \eta(i, j)$;
20.         end if
21.       end for
22.       if $C_j \neq \phi$
23.         $I_j \leftarrow C_j$
24.       else
25.         $I_j \leftarrow D_j$
26.       end if
27.       apply rule in (12) to select a server for VM $j$ and add the assignment into $S_m$
28.       apply pheromone local update rule in (13)
29.     end for
30.     if solution $S_m$ is infeasible then
31.       perform local search on $S_m$
32.       if $S_m$ is infeasible then
33.         Given more servers, and the VMs on overloaded servers in $S_m$ are adjusted by First-Fit algorithm
34.       end if
35.     end if
36.   end for
37.   add the solutions into $A$ and update $A$ by storing the nondominated solutions
38.   update $M^{\min}$
39.   apply global update rule in (14)
40.   $g \leftarrow g + 1$
41. end while
42. $\tau_{old} \leftarrow \tau$
```

4. Results

To evaluate the performance of the proposed algorithm, a series of simulation experiments with random workload and real workload are conducted. In each test instance, we assume that the allowed maximum utilization of each server is $U_{\max} = 0.9$. The server with utilization exceeding 100% is considered overloaded while under $U_{\text{under}} = 0.5$ is seen as underutilized. We compare UACS with three algorithms, the traditional heuristic approach, BFD; probabilistic method, ecoCloud [15]; and ACS-VM [32]. BFD is widely adopted for DVMP with a consolidation ratio of 11/9 and is used as a baseline. It allocates the VM to the server that fits best. In ecoCloud, each server determines whether to receive the VM for placement with a probability according to its own local information. ACS-VM is a very recently proposed algorithm for VM migration and has shown good performance. These representative algorithms make the comparisons more comprehensive and convincing. The parameters of the compared algorithms are set follow the original paper. For UACS, the related parameters are $m = 5$, $q_0 = 0.7$, $\rho = 0.1$, $\varepsilon = 0.1$, $\beta = 2.0$, and $NA = 10$ and maximal generation $G_{\max} = 20$. The SLAO violated level, the total number of VM migrations, and the energy consumption during the runtime of all algorithms on each test case are reported.

4.1. Random Workload

In the random workload, each VM runs an application with a variable CPU and memory utilization generated by a uniform distribution. The simulation is based on a Poisson arrival process with different arrival rate λ. The duration of the VM is generated by a Gaussian distribution with a mean value of 150 and a standard deviation of 10. Five test instances, named RW1 to RW5, are performed. RW1 and RW2 are to evaluate the performance of algorithms with different configurations of VMs and servers. RW3 and RW4 are to test the scalability of the proposed algorithm. In the real cloud, the requests often present a periodicity variation with a large number of requests in the daytime and fewer requests at night. Thus, we simulate an instance with an λ of "9.6 + 30 + 9.6" across one day in RW5. The configurations of the servers and VMs in the five instances are listed as Table 1.

Table 1. Configurations of virtual machines (VMs) and servers in random workloads, RW1 to RW5.

Test	λ	CPU Range	RAM Range	VM Number	Server Number (CPU, RAM)
RW1	9.6	[1, 4]	[1, 8]	10,000	8000 (24, 48)
RW2	9.6	[1, 1]	[0, 3]	10,000	8000 (16, 32)
RW3	13.3	[1, 4]	[1, 8]	19,200	8000 (16, 32)
RW4	30	[1, 4]	[1, 8]	43,200	8000 (16, 32)
RW5	9.6 + 30 + 9.6	[1, 4]	[1, 8]	27,288	8000 (16, 32)

The SLAO violated level, the number of VM migrations, and the energy consumption of all algorithms in RW1 to RW5 are reported in Table 2. Refering to the number of migrations, no migrations occur in ecoCloud, and ACS-VM gets the second smallest values. BFD gets the maximum number of migrations in each case. For the energy consumption, UACS gets the minimum value in RW1 and the second highest values in RW2–RW5, following ACS-VM. From Table 2, we can see that the SLAO value of UACS, ecoCloud, and BFD is 0, while that of ACS-VM becomes larger with the growth of the VM number. ACS-VM cannot deal with large-scale scenarios and many servers are in an overloaded state for half of the active time. In RW1, UACS adopts more VM migrations to avoid server overload and increase resource utilization. UACS uses less energy consumption than ACS-VM. This shows the stronger ability of UACS to consolidate VMs and increase resource utilization. Although ACS-VM can obtain a small number of VM migrations, it cannot meet the SLA requirements. In ACS-VM, a migration is allowed only if it can reduce the active server number and the number of migrations. This selection mechanism causes a situation in which the VMs on overloaded servers cannot be migrated and leads to a smaller value for energy consumption but a higher SLAO violated level in RW2 to RW5. This means that ACS-VM often finds solutions that violate the SLA requirements, which are not efficient in real cloud systems. The ecoCloud needs no migrations but displays large energy consumption in all cases. Due to the probabilistic assignment method based on local information, ecoCloud consolidates the VMs on more servers and causes resource waste. Meanwhile, the performance of the algorithm greatly depends on the parameters in the probability function. For BFD, it gets the largest number of VM migrations and the second last rank for energy consumption. Therefore, when the SLAO violated levels, the number of VM migrations, and the energy consumption are combined, UACS performs the best and needs a smaller number of VM migrations and consumes less energy while maintaining QoS requirements.

4.2. Real Workload

To further evaluate the performance of UACS on real workload traces in the cloud system, three test cases, RW6 to RW8, are conducted. We use the workload patterns from the three distinct sets MIX0, MIX1, and MIX2 provided in [21]. In each set of MIX0, MIX1, and MIX2, there are 20 traces selected from the 481 raw workload traces from a large European data center [21]. The demand patterns in MIX0, MIX1, and MIX2 are illustrated in Figure 2. Most of the workload traces in MIX0 present small variance during the execution time, while the workload traces in MIX1 show many short-term bursts [21]. MIX3 mixes the samples in MIX0 and MIX1 simultaneously. The measured workload traces describe the VM utilization in values of range [0, 150] on each logical CPU. In the three test cases, RW6, RW7, and RW8, each VM selects a random demand pattern from the corresponding workload traces set, MIX0, MIX1, and MIX2, respectively, as its time-varying resource requirement. Each server is equipped with a single CPU with four cores with a total of 200 capacity units and a 16 GByte memory. A VM is configured with two virtual CPU cores and a two GByte memory. In each case, the arrival rate λ is set as 3.4 and the total VM number is 10,000. RW6, RW7, and RW8 evaluate the ability of UACS to deal with workload traces with characteristics of normal bursts, short-term bursts, and hybrid normal and short-term bursts, respectively.

Table 3 reports the SLAO violated levels, the number of VM migrations, and the energy consumption caused by the UACS, BFS, ecoCloud, and ACS-VM methods on RW6, RW7, and RW8. Similar to its performance in RW1 to RW5, ACS-VM gets fewer VM migrations and minimum power consumption but large values for the SLAO violated level in RW6 to RW8. ACS-VM cannot deal with the server overload cases and violates QoS requirements. This limits ACS-VM's application to the real cloud. ecoCloud needs the least number of VM migrations but displays the largest energy consumption, while meeting QoS. This is because ecoCloud assigns the VMs to more servers, and many servers operate at low utilization. This over-provisioning reduces the probability of server overload and leads to fewer migrations. BFD needs the largest number of migrations and displays larger energy consumption than UACS in RW7 and RW8. Compared with ACS-VM, ecoCloud, and BFD, our proposed UACS is able to use relatively small migrations and displays low energy consumption, while maintaining QoS requirements. UACS can deal with the large scale DVMP with different resource demand patterns, such as normal bursts, short-term bursts, and hybrid normal and short-term bursts.

Table 2. Results of UACS, ACS-VM, ecoCloud, and best-fit decreasing (BFD) on random workloads (RW), RW1 to RW5.

Metrics	**Number of VM Migrations**				**Energy Consumption (kWh)**				**SLAO**			
	UACS	**ACS-VM**	**ecoCloud**	**BFD**	**UACS**	**ACS-VM**	**ecoCloud**	**BFD**	**UACS**	**ACS-VM**	**ecoCloud**	**BFD**
RW1	3.69×10^3	6.78×10^2	**0**	1.45×10^6	**263.00**	270.00	4433.33	286.66	**0**	0.0065	**0**	**0**
RW2	5.97×10^3	1.15×10^3	**0**	1.47×10^6	3120.00	**2433.33**	5283.33	5266.66	**0**	0.3541	**0**	**0**
RW3	1.45×10^6	1.61×10^3	**0**	2.84×10^6	1900.00	**1816.66**	8933.33	2266.66	**0**	0.5624	**0**	**0**
RW4	3.35×10^6	1.67×10^3	**0**	6.40×10^6	4266.66	**3983.33**	2,0166.67	5100.00	**0**	0.6157	**0**	**0**
RW5	1.29×10^6	1.63×10^3	**0**	4.04×10^6	2916.66	**2550.00**	12683.33	3216.66	**0**	0.5895	**0**	**0**

The results with boldface are the best results among the compared algorithms.

Table 3. The number of VM migrations, energy consumption, and the SLAO violated levels of UACS, ACS-VM, ecoCloud, and BFD on real workloads, RW6, RW7, and RW8.

Metrics	**Number of VM Migrations**				**Energy Consumption (kWh)**				**SLAO**			
	UACS	**ACS-VM**	**ecoCloud**	**BFD**	**UACS**	**ACS-VM**	**ecoCloud**	**BFD**	**UACS**	**ACS-VM**	**ecoCloud**	**BFD**
RW6	65,606	3211	**470**	2,622,035	12,083.33	**9833.33**	39,416.66	12,000.00	**0**	0.6693	**0**	**0**
RW7	190,615	3144	**674**	2,233,933	5350.00	**4125.00**	34,916.66	5400.00	**0**	0.6517	**0**	**0**
RW8	129,095	3145	**1159**	2,347,086	7791.66	**6250.00**	33,333.33	7808.33	**0**	0.6409	**0**	**0**

The results with boldface are the best results among the compared algorithms.

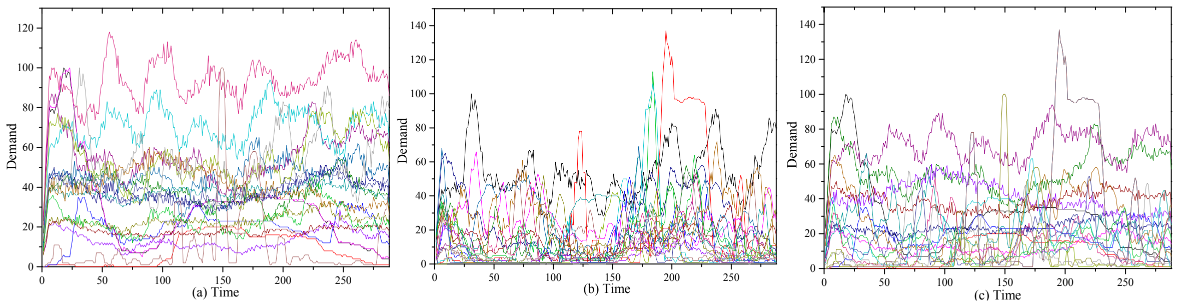

Figure 2. The resource demand patterns in (**a**) MIX0; (**b**) MIX1; and (**c**) MIX2.

In order to obverse the inner behavior of UACS, we take RW8 with hybrid patterns as an example and plot the active server number and the number of migrations used during the runtime. The results before 3000 intervals are illustrated in Figure 3. From Figure 3, we see that the number of active servers increases at the beginning time and remains approximately stable in the middle time, while it tends to decrease in the late stage. In the early time, new VMs arrive and there are more and more VMs running on servers. As time passes by, the data center reaches an approximate balance point between the resource used by the new incoming VMs and the finishing VMs, since the arrival rate is fixed. Then the active server number changes by infinitesimal increments or decrements. Due to the time-varying resource demand and short-term bursts, the VMs on overloaded servers are migrated, as shown in Figure 3b. In the late stage, no new VMs come, and meanwhile more and more VMs have finished their tasks. Then migrations occur to consolidate the VMs on underutilized severs to increase resource utilization and reduce energy consumption. From Figure 3, UACS is able to consolidate the VMs on fewer servers by VM migration.

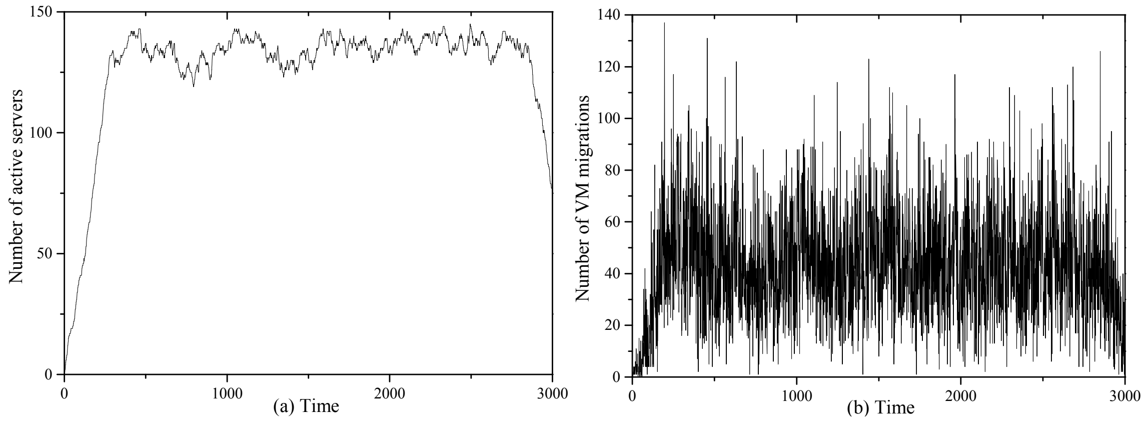

Figure 3. (**a**) Number of active servers; (**b**) Number of VM migrations used by UACS in RW8 during runtime.

5. Conclusions

In this paper, we model the DVMP as a multiobjective optimization problem and present an ACS based unified algorithm, termed UACS, for DVMP to deal with both the conditions of new incoming VM assignments and VM migration. UACS aims to find solutions with minimum energy consumption and VM migrations, while ensuring QoS requirements from a global perspective. In UACS, the number of provided servers decreases as the number of generations increases. In this way, the UACS concentrates more computation on the unexplored PF area of the multiobjective space. A dynamic pheromone deposition method is adopted to record the historical experience. Meanwhile, heuristic information is designed further to enhance the probability of selecting the original assigned server to reduce the number of migrations. Experiments on large scale VM requests with random workload and real workload traces are conducted. The resource demand patterns in different test cases present different characteristics such as normal and short-term bursts. Compared with heuristic, probabilistic, and other ACS based algorithms, the UACS is able to get better placements with a smaller number of VM migrations and lower energy consumption, while maintaining QoS requirements.

Acknowledgments: This work was partially supported by the National Natural Science Foundations of China (NSFC) with No. 61402545, the Natural Science Foundations of Guangdong Province for Distinguished Young Scholars with No. 2014A030306038, the Project for Pearl River New Star in Science and Technology with No. 201506010047, the GDUPS (2016), and the NSFC Key Program with No. 61332002.

Author Contributions: Xiao-Fang Liu conducted the experiments, performed the experiments, and wrote the draft of this paper. The idea of this paper was proposed by Zhi-Hui Zhan, who also made contributions to helping organize and write the paper, together with Jun Zhang.

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

References

1. Foster, I.; Zhao, Y.; Raicu, I.; Lu, S.Y. Cloud computing and grid computing 360-Degree compared. In Proceedings of the IEEE Grid Computing Environments Workshop, Austin, TX, USA, 12–16 November 2008.
2. Lanza, J.; Sánchez, L.; Gutiérrez, V.; Galache, J.A.; Santana, J.R.; Sotres, P.; Muñoz, L. Smart city services over a future internet platform based on internet of things and cloud: The smart parking case. *Energies* **2016**, *9*, 719. [CrossRef]
3. Zhan, Z.H.; Liu, X.F.; Zhang, H.; Yu, Z.; Weng, J.; Li, Y.; Gu, T.; Zhang, J. Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version. *IEEE Trans. Parallel Distrib. Syst.* **2016**. [CrossRef]
4. Motta, G.; Sfondrini, N.; Sacco, D. Cloud computing: An architectural and technological overview. In Proceedings of the International Joint Conference Service Science, Shanghai, China, 24–26 May 2012.
5. Beloglazov, A.; Buyya, R. Optimal Online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurr. Computat. Pract. Exp.* **2012**, *24*, 1397–1420. [CrossRef]
6. Callou, G.; Ferreira, J.; Maciel, P.; Tutsch, D.; Souza, R. An integrated modeling approach to evaluate and optimize data center sustainability, dependability and cost. *Energies* **2014**, *7*, 238–277. [CrossRef]
7. Perspectives, I. Using a Total Cost of Ownership (TCO) Model for Your Data Center. Available online: http://www.datacenterknowledge.com/archives/2013/10/01/using-a-total-cost-of-ownership-tco-model-for-your-data-center/ (accessed on 20 February 2017).
8. Filani, D.; He, J.; Gao, S.; Rajappa, M.; Kumar, A.; Shah, P.; Nagappan, R. Dynamic data center power management: Trends, issues, and solutions. *Int. Technol. J.* **2008**, *12*, 59–67. [CrossRef]
9. Cheng, C.C.; Lee, D.; Wang, C.H.; Lin, S.F.; Chang, H.P.; Fang, S.T. The development of cloud energy management. *Energies* **2015**, *8*, 4357–4377. [CrossRef]
10. Chiaraviglio, L.; Cianfrani, A.; Listanti, M.; Liu, W.; Polverini, M. Lifetime-aware cloud data centers: Models and performance evaluation. *Energies* **2016**, *9*, 470. [CrossRef]
11. Cui, X.; Mills, B.; Znati, T.; Melhem, R. Shadow replication: An energy-aware, fault-tolerant computational model for green cloud computing. *Energies* **2014**, *7*, 5151–5176. [CrossRef]
12. Bourguiba, M.; Haddadou, K.; Korbi, I.E.; Pujolle, G. Improving network i/o virtualization for cloud computing. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 673–681. [CrossRef]
13. 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]
14. Barroso, L.A.; Holzle, U. The case for energy-proportional computing. *Computer* **2007**, *40*, 33–37. [CrossRef]
15. 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]
16. Kandula, S.; Sengupta, S.; Greenberg, A.; Patel, P.; Chaiken, R. The nature of data center traffic: Measurements & analysis. In Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, Chicago, IL, USA, 4–6 November 2009.
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. Setzer, T.; Bichler, M. Using matrix approximation for highdimensional discrete optimization problems: Server consolidation based on cyclic time-series data. *Eur. J. Oper. Res.* **2013**, *227*, 62–75. [CrossRef]
19. Zhang, S.; Qian, Z.Z.; Luo, Z.Y.; Wu, J.; Lu, S.L. Burstiness-aware resource reservation for server consolidation in computing clouds. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 964–977. [CrossRef]
20. Boutaba, R.; Cheng, L.; Zhang, Q. On cloud computational models and the heterogeneity challenge. *J. Internet Serv. Appl.* **2012**, *3*, 77–86. [CrossRef]
21. Wolke, A.; Bichler, M.; Setzer, T. Planning vs. dynamic control: Resource allocation in corporate clouds. *IEEE Trans. Cloud Comput.* **2016**, *4*, 322–335. [CrossRef]
22. Beloglazov, A.; Abawajy, J.; Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Gener. Comput. Syst.* **2012**, *28*, 755–768. [CrossRef]
23. Mishra, A.K.; Hellerstein, J.L.; Cirne, W.; Das, C.R. Towards characterizing cloud backend workloads: Insights from Google compute clusters. *ACM SIGMETRICS Perform. Eval. Rev.* **2010**, *37*, 34–41. [CrossRef]
24. Islam, S.; Keung, J.; Lee, K.; Liu, A. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Gener. Comput. Syst.* **2012**, *28*, 155–162. [CrossRef]
25. Zhang, Q.; Zhani, M.F.; Boutaba, R.; Hellerstein, J.L. Dynamic heterogeneity-aware resource provisioning in the cloud. *IEEE Trans. Cloud Comput.* **2014**, *2*, 14–28. [CrossRef]
26. Verma, M.; Gangadharan, G.R.; Narendra, N.C.; Vadlamani, R.; Inamdar, V.; Ramachandran, L.; Calheiros, R.N.; Buyya, R. Dynamic resource demand prediction and allocation in multi-tenant service clouds. *Concurr. Computat. Pract. Exp.* **2016**. [CrossRef]
27. Zhan, Z.H.; Liu, X.F.; Gong, Y.J.; Zhang, J.; Chung, H.; Li, Y. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Comput. Surv.* **2015**, *47*, 1–33. [CrossRef]
28. Wu, Q.; Peng, C. A least squares support vector machine optimized by cloud-based evolutionary algorithm for wind power generation prediction. *Energies* **2016**, *9*, 585. [CrossRef]
29. Huang, M. Hybridization of chaotic quantum particle swarm optimization with SVR in electric demand forecasting. *Energies* **2016**, *9*, 426. [CrossRef]
30. Mi, H.; Wang, H.; Yin, G.; Zhou, Y.; Shi, D.; Yuan, L. Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In Proceedings of the IEEE International Conference on Services Computing, Miami, FL, USA, 11–15 July 2010.
31. Ashraf, A.; Porres, I. Using ant colony system to consolidate multiple web applications in a cloud environment. In Proceedings of the 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, Turin, Italy, 12–14 February 2014.
32. Farahnakian, F.; Ashraf, A.; Pahikkala, T.; Liljeberg, P.; Plosila, J.; Porres, I.; Tenhunen, H. Using ant colony system to consolidate VMs for green cloud computing. *IEEE Trans. Serv. Comput.* **2015**, *8*, 187–198. [CrossRef]
33. Liu, X.F.; Zhan, Z.H.; Deng, J.D.; Li, Y.; Gu, T.L.; Zhang, J. An energy efficient ant colony system for virtual machine placement in cloud computing. *IEEE Trans. Evolut. Computat.* **2016**. [CrossRef]
34. Fan, X.B.; Weber, W.-D.; Barroso, L.A. Power provisioning for a warehouse-sized computer. *ACM SIGARCH Comput. Archit. News* **2007**, *35*, 13–23. [CrossRef]
35. Dorigo, M.; Gambardella, L.M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evolut. Comput.* **1997**, *1*, 53–66. [CrossRef]
36. Deb, K.; Mohan, M.; Mishra, S. Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions. *Evolut. Comput.* **2005**, *13*, 501–525. [CrossRef] [PubMed]
37. Zhang, Q.; Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evolut. Computat.* **2007**, *11*, 712–731. [CrossRef]
38. Zhan, Z.H.; Li, J.; Cao, J.; Zhang, J.; Chung, H.; Shi, Y.H. Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems. *IEEE Trans. Cybern.* **2013**, *43*, 445–463. [CrossRef] [PubMed]
39. Yue, M. A simple proof of the inequality FFD (L) $\leq$ 11/9 OPT (L) + 1, for all L for the FFD bin-packing algorithm. *Acta Math. Appl. Sin.* **1991**, *7*, 321–331. [CrossRef]