Applying Ant Colony System algorithm in multi-objective resource allocation for virtual services

Nguyen Minh Nhut Pham* and Van Son Le

Vietnam Korea Friendship Information Technology College, Danang, Vietnam; Department of Informations, University of Education, Danang University, Danang, Vietnam

ABSTRACT

Over the past few years, using cloud computing technology has become popular. With the cloud computing service providers, reducing the number of physical machines providing resources for virtual services in cloud computing is one of the efficient ways to reduce the amount of energy consumption which in turn enhance the performance of data centres. However, using a minimum of physical machines to allocate resources for virtual services can result in system overload and break the SLA of service. Consequently, providing resources for virtual services which do not only satisfy the constraint of reducing the energy consumption but also ensure the load balancing of the whole system is necessary. In this study, we present the multi-objective resource allocation problem for virtual services. This problem aims at both reducing the energy consumption and balancing the load of physical machines. The MORA-ACS algorithm is proposed to resolve the problem by the Ant Colony System method. The experimental results show that in the CloudSim environment, the MORA-ACS algorithm could balance the load as well as reduce the energy consumption better than the Round Robin algorithm.

1. Introduction

In cloud computing, the virtualization technology allows partitioning the resource of physical machines into virtual machines to execute the applications on demands. A system that consists of multiple physical machines connecting together for sharing resources is called a shared hosting platform (Stillwell, Vivien, & Casanova, 2012). One of the challenges of this system facing is to optimize the resources of the platform for virtual services while still ensuring the Quality of Service.

Furthermore, the development of virtual technology and the applicability of cloud computing have led to the rapid growth of the need for physical machines in data centres. The amount of energy consumed by data centres was between 1.1% and 1.5% of the total worldwide electric power consumption in 2010 (Koomey, 2011). This is consistent with the typical yearly electricity consumption of 120 million households, producing negative greenhouse effects and CO2 footprints. As a result, energy consumption increases more...
and more rapidly which can threaten the environment and cause serious problems. Moreover, besides the huge energy costs, heat dissipation undoubtedly increases with higher power consumption, doubles the probability of failures (Feng, Feng, & Ge, 2008) and breaks the Service Level Agreement (SLA). Prior studies (Arianyan, Taheri, & Sharifian, 2015; Beloglazov & Buyya, 2012; Calheiros, Ranjan, Beloglazov, De Rose, & Buyya, 2011; Cao & Dong, 2010; Farahnakian et al., 2010; Jansen & Brenner, 2011; Luo et al., 2012; Vigliotti & Batista, 2014) showed that energy consumption is relevant to the resource of physical machines (e.g. CPU, RAM, hard disk, bandwidth, etc.) which are used in providing virtual services.

One of the ways to decrease the energy consumption of the system is decreasing the number of physical machines. However, this will increase the load of physical machines which can cause a system overload and break the SLA. In order to overcome this issue, the load balancing mechanism among all physical machines should be considered during the resource-providing process for virtual services. Therefore, exploring the optimal solution to allocating resources which meet the objectives of minimizing the energy consumption as well as achieving the load balancing among physical machines plays an essential role in virtual services.

However, almost prior studies only focused on single objective: balancing the load or minimizing the energy consumption of the system. This results in a limitation in allocating resources for the virtual services in real clouds which require the resource allocation process to be flexible and relevant to the optimal objectives. To address this problem, Bachlaus, Tiwari, and Chan (2009) suggested a multi-objective optimization model to maximise the broad objectives of profit, time of delivery, quality for a supply chain problem. The results demonstrate the benefits of multi-objective optimization model for solving the type of optimal problems.

Therefore, this study aims to solve the multi-objective resource provision problem (physical resources) for virtual services on the heterogeneous shared platforms which involves in both minimizing the systems energy consumption and balancing the load among physical machines. The main results are as follows:

(a) Building the resource need model and allocating resources for virtual services with the constraint that each virtual service is a separate virtual machine.
(b) Stating the resource allocation problem for virtual services as an optimal problem: minimizing the energy consumption of the system and balancing the load among physical machines.
(c) Proposing the MORA-ACS algorithm based on the Ant Colony System algorithm using CloudSim (Calheiros et al., 2011) to solve the problem. Comparing the energy consumption, standard deviation and the algorithm’s processing time between the MORA-ACS and Round Robin algorithm (Klaithem, Nader, Mariam, & Jameela, 2012) executed on the real data.

The rest of this paper is structured as follows: Section 2 reviews the related works, while Section 3 presents the problem model as a two-objective optimal problem. Section 4 proposes the Pareto-optimal method to solve the multi-objective problems. Section 5
proposes the MORA-ACS algorithm and Section 6 shows the experimental method and the estimated results. The conclusions and solutions are indicated in Section 7.

2. Related works

Several researchers explored the resource allocation issue for virtual services which meets the minimal energy consumption constraint. Feller, Rilling, and Morin (2011) suggested a resource allocation model for virtual services using the Ant colony optimization algorithm to estimate. The simulation results proved that the energy consumption is reduced if the number of used physical machines is diminished. However, they assumed that the platforms of physical machines are homogeneous and experimented on the simulated data. In contrast, we considered the platforms as heterogeneous ones, i.e. the resource configurations of physical machines are different, and then empirically tested on the real data (Beloglazov & Buyya, 2012; Calheiros et al., 2011).

Stillwell et al. (2012) solved the resource allocation problem as a linear programming model and used the FFD algorithm to assess. Conversely, Setzer, and Stage (2011) claimed that this algorithm tends to waste the resources. We thus propose the MORA-ACS algorithm which is based on the Ant Colony System algorithm to solve the problem.

In our previous studies (Nguyen, Dang, Nguyen, Le, & Nguyen, 2015; Pham et al., 2016), we proposed two algorithms to detect a deadlock mechanism for allocating resources in the heterogeneous shared platform. We designed the deadlock detection algorithms in order to resolve the resource optimization problems built on the recovery of allocated resources. Also, we proposed the greedy algorithms to solve a single-objective optimization problem: minimizing used physical machine. In this paper, we propose a multi-objective resource allocation (MORA) problem for virtual services based on the heterogeneous shared platform: minimizing the systems energy consumption and load balancing.

Prior studies have documented mixed evidence for the resource allocation approaches aimed at minimizing the systems energy consumption. Yet, they only focused on examining the energy consumption of CPU of physical machines. Other researches (Quan et al., 2012) have indicated, however, that the energy consumption does not only depend on the CPU, but also on other resources such as hard disk, bandwidth and RAM. In this research paper, we sought to research the MORA problem (physical resources) for virtual services on the heterogeneous shared server platforms with the aim of minimizing energy consumption of all resources of the system.

From another aspect, allocating resources which could balance the load over participated processors in order to improve the entire performance of the system have been paid attention in prior studies such as active clustering (Randles, Lamb, & Taleb-Bendiab, 2010), random sampling (Randles et al., 2010), active monitoring load balancer (Bhathiya, 2009), throttled load balancer (Bhathiya, 2009), Workload and Client Aware Policy (Mehta, Kanungo, & Chandwani, 2011), Join-Idle-Queue (Lu et al., 2011) and Central Load Balancing Policy for Virtual Machines (Bhadani & Chaudhary, 2010). The load balancing mechanism can be divided into two main types: dynamic load balancing and static load balancing. In static load balancing, loads of physical machines are distributed based on their resource-providing capabilities, but the dynamic changes of Load Balancer features are not mentioned during the runtime. Major static load balancing algorithms are Round Robin (Klaithem et al., 2012) and Weighted Round Robin (Klaithem
et al., 2012). By contrast, in dynamic load balancing the information about the resource needs of virtual service and resource capability of the system is collected in order to redistribute the loads among the physical machines. The most common dynamic load balancing algorithms for virtual services are least connection (Klaithem et al., 2012) and weighted least connection (Klaithem et al., 2012).

3. MORA for virtual service model

As stated in our previous work (Pham et al., 2016), a heterogeneous shared platform including a set of $M$ physical machines, $\mathbb{M} = \{j \mid j = 1, \ldots, M\}$, which has different resource configurations, and being interconnected by a high-speed network device which is deployed for sharing resources to set of $N$ virtual services, $\mathbb{N} = \{i \mid i = 1, \ldots, N\}$, is carried out in this paper. Each physical machine provides a set of $D$ types of resource, $\mathbb{D} = \{k \mid k = 1, \ldots, D\}$, where $N$ is the number of virtual services, $M$ is the number of physical machines and $D$ is the dimension of resources.

Each type of ability at an objective engine might have one or more distinct elements such as one or more separate real CPU, one or more single substantial memory and aggregate one. Thus, the capabilities of a corporeal appliance are represented by an ordered pair of multi-dimensional resource vectors. The elementary resource vector expresses the capacity of a single aspect, while the aggregate resource vector executes the sum of multi-dimensional resource vectors. The elementary resource vector expresses the gate one. Thus, the capabilities of a corporeal appliance are represented by an ordered pair such as one or more separate real CPU, one or more single substantial memory and aggregate one. Indeed, the resource needs of each virtual service are categorized into two groups: rigid needs and fluid needs (Stillwell et al., 2012). Rigid needs act as a specific fraction of required resource. The virtual service cannot benefit from a larger fraction and operate with a smaller one as well. Fluid needs specify the additional fraction of a resource that the service could use. The virtual service cannot gain from a bigger portion, but can run with a minor one at a lower cost.

Hence, the rigid needs of a virtual service $i$ for a resource type $k$ are represented by a first ordered vector pair $(R_e^i, R_a^i)$, which determines the resource demands in running the virtual service at the acceptable level. In which, $R_e^i = \{r_{ik}^e \mid r_{ik}^e \in \mathbb{Q}^+; i = 1, \ldots, N; k = 1, \ldots, D\}$ and $R_a^i = \{r_{ik}^a \mid r_{ik}^a \in \mathbb{Q}^+; i = 1, \ldots, N; k = 1, \ldots, D\}$. And, $r_{ik}^e, r_{ik}^a$ are items which denote the elementary and aggregate rigid needs of a resource type $k$ of a virtual service $i$, respectively.

The fluid needs of a virtual service $i$ for a resource type $k$ are represented by a second ordered vector pair $(F_e^i, F_a^i)$, which displays the additional resource demand when running the virtual service at the maximum performance level. In which, $F_e^i = \{f_{ik}^e \mid f_{ik}^e \in \mathbb{Q}^+; i = 1, \ldots, N; k = 1, \ldots, D\}$ and $F_a^i = \{f_{ik}^a \mid f_{ik}^a \in \mathbb{Q}^+; i = 1, \ldots, N; k = 1, \ldots, D\}$. And, $f_{ik}^e, f_{ik}^a$ are items which denote the elementary and aggregate fluid needs of a resource type $k$ of a virtual service $i$, respectively.

The utilization of all resources corresponding to fluid needs is linearly correlated. For example, if the virtual service is only provided by with a half of the CPU resource needs,
the possibility is that it only uses half of the bandwidth I/O compared with the resource needs. This is consistent with the reality because when the CPU resource needs cut down, this leads to the reduction of other resources consumed (in this case is the bandwidth I/O). For simplicity, the additional factor of all fluid needs can show the same value and its value is between 0 and 1, with 0 coinciding with the case the virtual service is not provided the fluid resource, whereas with 1 conforming to the virtual service which executes at the maximum resource allocation. Therefore, resource needs of resource type \( k \) of virtual service \( i \) on physical machine \( j \) with additional factor are given by an ordered vector pair \((R^k + Q \times F^e, R^a + Q \times F^a)\), in which \( Q = \{q_{ij} \mid q_{ij} \in \mathbb{Q}^+; \ i = 1, \ldots, N; \ j = 1, \ldots, M\} \) is a vector of additional factor of virtual services when the user requests. And, \( q_{ij} \) intends the additional factor of virtual service \( i \) on physical machine \( j \).

Similar to the study of Stillwell et al. (2012), we assume that elementary resource needs of a virtual service do not exceed the elementary resource capacity of a physical machine. Providing resource also fails if the aggregate resource needs of virtual services exceed the capacity of physical machines aggregate resources. And, we assume that each virtual service consists of a single virtual machine which has a fixed resource needs.

In contrast to our previous work (Pham et al., 2016) that only considers a single-objective optimization problem: minimizing used physical machine, we propose an MORA problem for virtual services based on the heterogeneous shared platform: minimizing the system's energy consumption and balancing the load among physical machines as follows.

Assume a set of \( \mathbb{N} \) virtual services, \( \mathbb{N} = \{i \mid i = 1, \ldots, N\} \) and \( N \) be the number of virtual services, a set of \( \mathbb{M} \) physical machines, \( \mathbb{M} = \{j \mid j = 1, \ldots, M\} \) and \( M \) be the number of physical machines having the different resource configuration. Each physical machine provides a set of \( \mathbb{D} \) types of resource, \( \mathbb{D} = \{k \mid k = 1, \ldots, D\} \) and \( D \) be the number of resource types. Use \( r^e_{ik}, r^a_{ik} \) to denote the elementary rigid needs and aggregate rigid needs of virtual service \( i \) for resource type \( k \). \( f^e_{ik}, f^a_{ik} \) stand for the elementary fluid needs and aggregate fluid needs of virtual service \( i \) for resource type \( k \), respectively. \( c^e_{jk}, c^a_{jk} \) imply the elementary resource and aggregate resource of physical machine \( j \) for resource type \( k \), and \( q_{ij} \) intends the additional factor of virtual service \( i \) on physical machine \( j \). Binary variable \( x_{ij} \) is equal to 1 if virtual service \( i \) is allocated the resource by physical machine \( j \) and 0 otherwise.

Finally, \( p^\text{max}_j \in \mathbb{Q}^+ \) and \( p^\text{idle}_j \in \mathbb{Q}^+ \) are the power of physical machine \( j \) in the maximum used utilities state and idle state, respectively. Given these notations, a MORA problem is described as follows:

\[
x_{ij} \in \{0, 1\}; q_{ij}, r^e_{ik}, r^a_{ik}, f^e_{ik}, f^a_{ik}, c^e_{jk}, c^a_{jk} \in \mathbb{Q}^+, \quad \forall i, j, k,
\]

\[
\sum_{j=1}^{M} x_{ij} = 1, \quad \forall i,
\]

\[
(r^e_{ik} + q_{ij} \times f^e_{ik}) \times x_{ij} \leq c^e_{jk}, \quad \forall i, j, k,
\]

\[
\sum_{i=1}^{N} (r^a_{ik} + q_{ij} \times f^a_{ik}) \times x_{ij} \leq c^a_{jk}, \quad \forall j, k,
\]
\[
R_j = \sum_{k=1}^{D} \left( c_{jk}^a - \sum_{i=1}^{N} (r_{ik}^a + q_{ij} \times f_{ik}^a) \times x_{ij} \right), \quad \forall j, \tag{5}
\]

\[
\bar{R} = \frac{1}{M} \times \sum_{j=1}^{M} R_j, \tag{6}
\]

\[
\sigma = \sqrt{\frac{1}{M} \times \sum_{j=1}^{M} (R_j - \bar{R})^2} \rightarrow \min, \tag{7}
\]

\[
E = \left( \sum_{j=1}^{M} \left( \sum_{k=1}^{D} \left( (P_{j}^{\text{max}} - P_{j}^{\text{idle}}) \times \sum_{i=1}^{N} \left( \frac{(r_{ik}^a + q_{ij} \times f_{ik}^a) \times x_{ij}}{c_{jk}^a} + P_{j}^{\text{idle}} \right) \right) \right) \right) \times \Delta T \rightarrow \min. \tag{8}
\]

Constraint (1) defines the domain of the variables. Constraint (2) determines the state that resources for virtual service \(i\) are provided by exactly one physical machine \(j\). Constraint (3) states that the elementary resource of physical machine \(j\) does not overcome, and constraint (4) disposes that the aggregate resource of physical machine \(j\) does not overcome. Constraint (5) represents the residual resources of physical machine \(j\) when allocating resource for virtual service. Constraint (6) acts for the average residual resource of all physical machines. Equation (7) states the standard deviation with the objective of finding the optimal solution so as to achieve the system’s load balancing. Equation (8) shows the objective of minimizing the energy consumption of the system in a time period \(\Delta T\).

### 4. Pareto-optimal solution

For a problem having \(T\) objective function with the objective function \(f_j, \quad j = 1, 2, \ldots, T; \quad T > 1\). Let \(s\) and \(s'\) be the feasible solutions of problem. A solution \(s\) is said to dominate the other solution \(s'\) if both the following conditions are true:

- The solution \(s\) is no worse than (represented: \(\preceq\)) \(s'\) in all objectives, or \(f_j(s) \preceq f_j(s'), \quad j = 1, 2, \ldots, T\).
- The solution \(s\) is strictly better than (represented: \(\prec\)) \(s'\) in at least one objective, or \(f_{j^*}(s) < f_{j^*}(s'), \quad j^* = \{1, 2, \ldots, T\}\).

If one of the above conditions is desecrated, the solution \(s\) does not dominate the solution \(s'\), or \(s\) is non-dominated by \(s'\). If a solution is not dominated by other solutions in the solution space, then it is called the Pareto optimization solution. The set of non-dominated feasible solutions in the solution space is called the Pareto optimization set. With a Pareto optimization set, the set of objective function values corresponding to each solution in the solution space is called the Pareto Front set.

Algorithm 1 (Branke, Deb, Miettinen, & Slowinski, 2008; Deb, 2001; Ikeda et al., 2006) can be used to get the non-dominated set of the feasible solutions.
5. Applying Ant Colony System algorithm in solving MORA problem

The Ant Colony optimization method is used to design heuristic for optimal problem algorithm. The ant colony algorithm finds the optimal path simulation based on the food searching behaviour of real ants. If an ant seeks a food source, it crawls back to the colony leaving pheromones (markers) which indicate the path having food. When other ants come across the markers, they settle the path by their own markers as they carry the food back. As more ants seek the path, the markers get stronger and the ants then choose the path having the strongest pheromones, hence finding the shortest path (in other words, optimizing solution).

The Ant Colony optimization method was first introduced in 1991 by Dorigo (Colorni, Dorigo, & Maniezzo, 1991) and then several algorithms based on this principle of algorithm have been developed such as Ant System algorithm, Max–Min Ant System algorithm and Ant Colony System algorithm. In this paper, we aim to build the MORA-ACS algorithm based on the Ant Colony System algorithm (Dorigo & Gambardella, 1997) to solve the MORA problem for virtual services with two objectives: balancing the load among physical machines and minimizing the energy consumption of the whole system.

The characteristic of Ant colony optimization algorithms is using the elements of previous solutions through the use of random steps. The random probability is defined by combining the heuristic information of previous solution and the information of pheromones. The Ant Colony System algorithm (Dorigo & Gambardella, 1997) differs from the other ant colony optimization algorithms owing to the following three aspects: (a) the state transition rule when building the solution, (b) the global updating rule is used only to an ant which having the best solution and (c) the local pheromone updating rule is used while ants construct completely a solution.

5.1. State transition rule

In the resource allocation process, the ant selects virtual service \( i \) as the next virtual service to allocate a resource from current physical machine \( j \) according to the pseudo-random-proportional rule, given by the below equation:

\[
i = \begin{cases} \arg\max_{u \in \mathbb{N}^{nump}} \left[ \tau_{uj} \right]^{\alpha} \times \left[ \eta_{uj} \right]^{\beta}, & q \leq q^0, \\ \text{Equation (12)} & \text{otherwise}, \end{cases}
\]

where \( \alpha, \beta \geq 0 \) (to point out the pheromone and the heuristic information respectively); \( q \) is a random number which distributed uniformly in \([0,1]\) and \( q^0 \) is a parameter...
(1 ≤ q₀ ≤ 0) determined by the relationship between exploitation and exploration. If q > q₀, this process is called exploration, otherwise exploitation. \( N^{\text{temp}} \) is the set of virtual services in which if eligible, \( N^{\text{temp}} \) will be chosen to be allocated resource from current physical machine \( j \), i.e. the set of virtual services not being allocated resource and not breaking the constraint of resource need in all kinds of resource \( k \) of current physical machine \( j \), given by

\[
N^{\text{temp}} = \left\{ i \in \{1, \ldots, N\} \left| \sum_{j=1}^{M} x_{ij} = 0 \land (\text{Load}_j + (r_{ik}^a + q_{ij} \times f_{ik}^a)) \leq c_{jk}^a, \forall j, k \right. \right\},
\]

where \( \text{Load}_j \) is a load of current physical machine \( j \), and given by

\[
\text{Load}_j = \sum_{i \in j} (r_{ik}^a + q_{ij} \times f_{ik}^a), \quad \forall j, k.
\]

The probability which an ant selects virtual service \( i \) as the next virtual service to be allocated a resource from current physical machine \( j \) is calculated as follows:

\[
p_{ij} = \frac{[\tau_{ij}]^\alpha \times [\eta_{ij}]^\beta}{\sum_{u \in N^{\text{temp}}} [\tau_{uj}]^\alpha \times [\eta_{uj}]^\beta}, \quad \forall i \in N^{\text{temp}},
\]

where \( \tau_{ij} \) is the pheromone which the ant based on that marker selects virtual service \( i \) as the next virtual service to be allocated a resource from current physical machine \( j \). With the MORA-ACS algorithm, the time of leaving pheromone \( \tau_{ij} \) is defined as follows:

\[
\tau_{ij} = \tau_{ij}^0 = \frac{1}{N \times (E(s^0) + \alpha(s^0))},
\]

where \( E(s^0) \) and \( \alpha(s^0) \) are energy consumption and standard deviation of the solution \( s^0 \) at the setting up time as indicated in Equations (7) and (8), and \( N \) is the number of virtual services. In this study, we used the FFD algorithm (Stillwell et al., 2012) to build the solution \( s^0 \). With the aim at minimizing the energy consumption and balancing the load of all physical machines, information of heuristic \( \eta_{ij} \) is identified in Equation (16):

\[
\eta_{ij}^e = \frac{1}{\sqrt{\frac{1}{M} \sum_{j=1}^{M} (R_j - \bar{R})^2}},
\]

\[
\eta_{ij}^e = \frac{1}{\sum_{j=1}^{M} \sum_{k=1}^{D} ((P_{\text{busy}}^j + P_{\text{idle}}^j) \times \sum_{i=1}^{N} (r_{ik}^a + q_{ij} \times f_{ik}^a) \times x_{ij} + P_{\text{idle}}^j) \times \Delta T},
\]

where \( \eta_{ij}^e \) is the information of heuristic for the objective of load balancing and \( \eta_{ij}^e \) is the information of heuristic for the objective of minimizing the energy consumption in a time period \( \Delta T \).

### 5.2. Pheromone updating rule

According to the Ant Colony System optimization method, the pheromone updating mechanism is executed through the local pheromone updating rule or global pheromone
updating rule. When an ant builds the solution, the pheromone of that ant is updated as follows:

\[
\tau_{tij}^f = (1 - \rho_{\text{local}}^{\text{global}}) \times \tau_{tij}^{f-1} + \rho_{\text{local}}^{\text{global}} \times \tau_{0ij}^f, \quad \forall i = 1, \ldots, N; \ j = 1, \ldots, M, \quad (17)
\]

where \( \rho_{\text{local}}^{\text{global}} \in [0, 1) \), whereas \((1 - \rho_{\text{local}}^{\text{global}})\) represents the evaporation rate of pheromone over the time, \(\tau_{tij}^f\) is the pheromone at that time \(t\) and \(\tau_{tij}^{f-1}\) is the pheromone at the previous time \((t-1)\). After all ants build the solution, the global updating rule is applied. Besides this is the multi-objective optimization problem, we have used the Pareto optimization method. Therefore, the global pheromone updating rule, after setting up the solution \(s\) in the current Pareto, is defined as follows:

\[
\tau_{tij}^g = (1 - \rho_{\text{global}}^{\text{global}}) \times \tau_{tij}^{g-1} + \rho_{\text{global}}^{\text{global}} \times \frac{\text{numAnt}}{E(s_{\text{best}}) + \sigma(s_{\text{best}})^2}, \quad \forall i = 1, \ldots, N; \ j = 1, \ldots, M. \quad (18)
\]

where \(E(s_{\text{best}})\) and \(\sigma(s_{\text{best}})^2\) are energy solution and standard deviation with respect to the global optimal solution. \(\rho_{\text{global}}^{\text{global}} \in [0, 1)\), whereas \((1 - \rho_{\text{global}}^{\text{global}})\) represents the evaporation rate over the time, \(\text{numAnt}\) is the number of ants, \(\tau_{tij}^g\) is the pheromone at that time \(t\) and \(\tau_{tij}^{g-1}\) is the pheromone at the previous time \((t-1)\).

5.3. Algorithm MORA-ACS

In summary, pseudo-code of the colony system algorithm which serves the MORA problem for virtual services, MORA-ACS, is presented in Algorithm 2.

5.3.1. Operation of algorithm

The algorithm takes as input the set of virtual services \(\mathbb{N}\) and a set of resource types \(\mathbb{D}\) with the elementary rigid needs \(r_{ik}^e\), the aggregate rigid needs \(r_{ik}^a\), the elementary fluid needs \(f_{jk}^e\), aggregate fluid needs \(f_{jk}^a\), a set of physical machines \(\mathbb{M}\) with the elementary resource \(c_{jk}^e\) and the aggregate resource \(c_{jk}^a\) and the additional factor \(q_{ij}\).

First, the parameters \(\rho_{\text{local}}^{\text{local}}, \rho_{\text{global}}^{\text{global}}, \alpha, \beta, q^0, \text{numAnt}, \text{numLoop}\) and a set of empty Pareto-optimal solutions, \(\mathbb{P}\), are initialized. The initial solution is implemented by an FFD algorithm (Stillwell et al., 2012) and the pheromone trial, \(\tau_{0ij}^f\), is calculated by Equation (12) (lines 1–3).

Next, the algorithm enters the \text{numLoop} iterations and builds the resource allocation solutions from a set of physical machines \(\mathbb{M}\) to a set of virtual services \(\mathbb{N}\) (lines 4–33). In each iteration, each ant builds the solutions and updates the best local pheromone according to Equation (17) (lines 5–27).

After all ants have constructed their solutions, the algorithm calculates two best solution values by using Equations (7) and (8) which the best ants have constructed. Next, Algorithm 1 is used to determine the optimal Pareto solution, store the values in \(\mathbb{P}\) and update the global pheromone according to Equation (18) (lines 28–32).

Finally, after running all iterations, the algorithm will stop and return a set of Pareto-optimal values, \(\mathbb{P}\) (line 34).

5.3.2. Algorithm analysis

Let \(N\) be the number of virtual services, \(M\) be the number of physical machines, \(D\) be the dimension of resources, \text{numLoop} be the number of iterations and \text{numAnt} be the number
of ants. The complexity of algorithms is calculated as follows: \(O(n_{\text{numLoop}} \times n_{\text{numAnt}} \times N \times M \times D)\). If \(D\) is considered as a constant, the complexity of the MORA-ACS algorithm is an \(O(n_{\text{numLoop}} \times n_{\text{numAnt}} \times N \times M)\).

Let \(P(n_{\text{numLoop}})\) be the probability of finding the solution of the Ant Colony System algorithm after the iteration \(n_{\text{numLoop}}\). Using the non-homogeneous Markov model, Stutzl and Dorigo (Dorigo, Birattari, & Stutzle, 2006) demonstrated that:

\[ \forall \varepsilon > 0, \text{ if } n_{\text{numLoop}} \text{ is big enough, then } P(n_{\text{numLoop}}) > 1 - \varepsilon. \]

Therefore, \(\lim_{n_{\text{numLoop}} \to \infty} P(n_{\text{numLoop}}) = 1\). So, the MORA-ACS algorithm will converge after the iteration \(n_{\text{numLoop}}\).

6. Experimental method and results

6.1. Experimental method

The Round Robin algorithm (Klaithem et al., 2012) allocates resource for virtual service according to the one–one mechanism, i.e. the first virtual service is provided a resource
from the first physical machine, the second virtual service is provided a resource from the second physical machine, etc. If it turns to the last physical machine, the next virtual service is a circularly allocated resource from the first physical machine. This is the common algorithm and has been widely used in the cloud system. So as to assess the MORA-ACS algorithm, we compared this algorithm to the Round Robin algorithm (Klaithem et al., 2012) by using the CloudSim simulation tool (Calheiros et al., 2011).

As presented in Section 3, each virtual service is one separate virtual machine. To execute one virtual service, we inherited the $Vm$ layers to extend the resource need features of the virtual machine and the $Host$ layer to extend the resource features of the physical machine. Meanwhile, we inherited the $VMAllocationPolicy$ layer to perform the resource allocation policy for virtual machine based on the MORA-ACS algorithm.

The experimental data are extracted from the real data as indicated in Beloglazov and Buyya (2012) and Calheiros et al. (2011). Table 1 presents the resource characteristics and the power of physical machines. The resource characteristics of virtual machines, which are similar to the virtual machines of Amazon EC2 cloud, are adjusted to suit the problem, shown in Tables 2 and 3.

The number of virtual machines (represents for the number of virtual services) in the experimental scripts is $N = \{100; 200; 300; 400; 500\}$. Measure unit of energy consumption is kWh, and processing time is calculated in second(s). The performed time is measured by a PC having an Intel(R) Core(TM) i5-3235M 2.60 GHz, RAM 4Gb microprocessor.

The parameters of algorithm $\rho^{local}$, $\rho^{global}$, $\alpha$, $\beta$, $\rho^{0}$ are the experimental parameters. The selection of these parameters depends on the objective function. The experience shows

---

**Table 1.** Resource characteristics and power consumption of physical machine.

| Type of physical machine | CPU (MHz) | RAM (GB) | BW (GB/s) | Disk (GB) | $P_{idle}$ (Watt) | $P_{max}$ (Watt) |
|--------------------------|-----------|----------|-----------|-----------|------------------|------------------|
| HP proliant G4           | 2 core x 1860 | 4        | 1         | 20        | 86               | 117              |
| HP proliant G5           | 2 core x 2660 | 4        | 1         | 40        | 93.7             | 135              |
| IBM Server x 3250        | 4 core x 2933 | 8        | 1         | 600       | 46.1             | 113              |
| IBM Server x 3550        | 6 core x 3067 | 16       | 1         | 800       | 58.4             | 222              |

**Table 2.** CPU resource and RAM resource characteristics of virtual machine.

| Type of virtual machine | CPU (MHz) | RAM (GB) | Total |
|-------------------------|-----------|----------|-------|
| VM 1                    | 1000      | 1500     | 2500  |
| VM 2                    | 1000      | 1000     | 2000  |
| VM 3                    | 500       | 500      | 1000  |
| VM 4                    | 250       | 250      | 500   |

**Table 3.** BW resource and DISK resource characteristics of Virtual Machine.

| Type of virtual machine | BW (MHz/s) | Disk (GB) | Total |
|-------------------------|------------|-----------|-------|
| VM 1                    | 0.20       | 0.25      | 0.45  |
| VM 2                    | 0.10       | 0.25      | 0.35  |
| VM 3                    | 0.10       | 0.15      | 0.25  |
| VM 4                    | 0.05       | 0.10      | 0.15  |

---
that $\rho_{\text{local}} = \rho_{\text{global}} = 0.45$ and $\alpha = \beta = 0.5$, $q^0 = 0.9$ indicate the best results (best objective function value and earliest convergence algorithm). Each algorithm uses three measures: standard deviation (SD), energy consumption in a time period of $\Delta T = 24$ h and the runtime of algorithm. Moreover, we use Formula (19) to compare the gain energy consumption between two algorithms:

\[
\text{Gain Energy(\%)} = \frac{E_{\text{Round Robin}} - E_{\text{MORA-ACS}}}{E_{\text{MORA-ACS}}} \times 100.
\]

where $E_{\text{Round Robin}}$ is the energy consumption when using the Round Robin algorithm and $E_{\text{MORA-ACS}}$ is the energy consumption when using the MORA-ACS algorithm.

### 6.2. Experimental results

The experimental results, presented in Table 4 and Figure 1, show that the energy consumption and the standard deviation of the MORA-ACS algorithm are better than the Round Robin algorithm. This is explained by the fact that the Round Robin algorithm tends to use a large number of physical machines. Besides, the MORA-ACS algorithm uses a more efficient global search solution. Therefore, it uses fewer resources and ensures better load balancing. However, the runtime of the MORA-ACS algorithm is higher than that of the Round Robin algorithm due to the loop parameter, numLoop, and the number of ants, numAnt. Furthermore, in the experimental scenario, we found the following.

**Table 4.** Consumption energy and standard deviation (SD) of algorithms.

| Num. of VM | Algorithms    | Times (s) | Energy (kWh) | Gain energy(%) | SD    |
|------------|---------------|-----------|--------------|----------------|-------|
| 100        | Round Robin   | 0.031     | 201.284      | 53.24          |       |
|            | MORA-ACS      | 0.037     | 190.726      | 40.75          | 5.84  |
| 200        | Round Robin   | 0.078     | 396.706      | 49.95          |       |
|            | MORA-ACS      | 0.084     | 390.292      | 43.04          | 1.64  |
| 300        | Round Robin   | 0.116     | 597.989      | 52.35          | 4.09  |
|            | MORA-ACS      | 0.121     | 574.440      | 41.33          | 4.09  |
| 400        | Round Robin   | 0.144     | 793.411      | 54.63          | 5.84  |
|            | MORA-ACS      | 0.150     | 770.908      | 42.31          | 2.91  |
| 500        | Round Robin   | 0.200     | 994.694      | 53.03          | 4.09  |
|            | MORA-ACS      | 0.216     | 960.156      | 40.52          | 3.59  |

**Figure 1.** The graph of standard deviation and consumed energy.
The MORA-ACS algorithm could give optimal or non-optimal results depending on the input value of the algorithm, including the number of virtual services $N$, the number of resource types $D$, elementary rigid needs $r^e_{ik}$, aggregate rigid needs $r^a_{ik}$, elementary fluid needs $f^e_{ik}$, aggregate fluid needs $f^a_{ik}$, additional coefficient $q_{ij}$, the number of physical machines $M$, elementary resource $c^e_{jk}$ and the aggregate resource $c^a_{jk}$.

If the resource capacity of the physical machine is strong enough (elementary resource and aggregate resources) and the resource needs of the virtual service are low, the MORA-ACS algorithm gives a good effect. The best case of the algorithm is that when the first Ant instance finds an acceptable solution, then the algorithm stops and offers the acceptable solution.

The worst case is that all Ant instances do not find any solutions and the execution time of the algorithm is very large because of the algorithm complexity. This case occurs when the resource needs of virtual service are too large, the system does not have enough resources to provide.

7. Conclusion

The paper presents the MORA problem for virtual services with the optimal constraints: minimizing the energy consumption of the system and balancing the system load. Each virtual service is one separate virtual machine, and the resource need is unchanged during the process. Based on the Ant Colony System optimization method, we proposed the MORA-ACS algorithm. We also set up, assessed and compared with the Round Robin algorithm through the parameters: standard deviation of the residual resource of physical machines, energy consumption and algorithm’s processing time. The algorithms were executed on the real data by CloudSim. The experimental results indicate that the MORA-ACS algorithm yields a better performance in comparison with the Round Robin algorithm in executing the objective functions of the problem. In order to evolve this research, it should be further conducted with the resource allocation problem for virtual services which aims at achieving the dynamic balancing with respect to the virtual machine migration mechanism.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Arianyan, E., Taheri, H., & Sharifian, S. (2015). Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers. *Computers and Electrical Engineering, 47*, 222–240.

Bachlaus, M., Tiwari, M. K., & Chan, F. T. S. (2009). Multi-objective resource assignment problem in a product-driven supply chain using a Taguchi-based DNA algorithm. *International Journal of Production Research, 47*, 2345–2371.

Beloglazov, A., & Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience, 24*(13), 1397–1420.
Bhadani, A., & Chaudhary, S. (2010). Performance evaluation of web servers using central load balancing policy over virtual machines on cloud. Proceedings of the third annual ACM Bangalore conference (COMPUTE’10), Bangalore, India, No. 16.

Bhathiya, W. (2009). CloudAnalyst a CloudSim-based tool for modelling and analysis of large scale cloud computing environments. MEDC Project, Cloud Computing and Distributed Systems Laboratory, University of Melbourne (pp. 1–44).

Branke, J., Deb, K., Miettinen, K., & Slowinski, R. (2008). Multiobjective optimization: Interactive and evolutionary approaches. Berlin: Springer-Verlag.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., & Buyya, R. (2011). Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 41(1), 23–50.

Cao, Z., & Dong, S. (2010). Dynamic VM consolidation for energy-aware and SLA violation reduction in cloud computing. Parallel and distributed computing, applications and technologies (PDCAT), Beijing, China (pp. 363–369).

Colorni, A., Dorigo, M., & Maniezzo, V. (1991). Distributed optimization by ant-colonies. Proceedings of the first European conference on artificial life (ECAL’91), Paris, France (pp. 134–142).

Deb, K. (2001). Multi-objective optimization using evolutionary algorithms. New York: Wiley & Sons.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE Computational Intelligence Magazine, 1(4), 28–39.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1(1), 53–66.

Farahnakian, F., Ashraf, A., Liljebra, P., Pahlkkala, T., Plosila, J., Porres, I., & Tenhunen, H. (2010). Energy-aware dynamic VM consolidation in cloud data centers using ant colony system. 2014 IEEE seventh international conference on cloud computing, Anchorage, USA (pp. 104–111).

Feller, E., Rilling, L., & Morin, C. (2011). Energy-aware ant colony based workload placement in clouds. Grid computing, 2011 12th IEEE/ACM international conference, Lyon, France (pp. 26–33).

Feng, W. C., Feng, X., & Ge, R. (2008). Green supercomputing comes of age. IT Professional, 10(1), 17–23.

Ikeda, M., Barolli, L., Koyama, A., Durrese, A., De Marco, G., & Iwashiga, J. (2006). Performance evaluation of an intelligent CAC and routing framework for multimedia applications in broadband networks. Journal of Computer and System Sciences, 72(7), 1183–1200.

Jansen, R., & Brenner, P. R. (2011). Energy efficient virtual machine allocation in the cloud. Green computing conference and workshops (IGCC), Orlando, USA (pp. 1–8).

Klaus, A. N., Nader, M., Mariam, A. N., & Jameela, A.-J. (2012). A survey of load balancing in cloud computing: Challenges and algorithms. Second symposium on network cloud computing and applications, London, England (pp. 137–142).

Koomey, J. (2011). Growth in data center electricity use 2005 to 2010. A Report by Analytical Press. Completed at the Request of The New York Times (vol. 9).

Lu, Y., Xie, Q., Kliot, G., Geller, A., Larus, J. R., & Greenberg, A. G. (2011). Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services. Performance Evaluation, 68, 1056–1071.

Luo, L., Wu, W., Di, D., Zhang, F., Yan, Y., & Mao, Y. (2012). A resource scheduling algorithm of cloud computing based on energy efficient optimization methods. Green computing conference (IGCC), San Jose, USA (pp. 1–6).

Mehta, H., Kanungo, P., & Chandwani, M. (2011). Decentralized content aware load balancing algorithm for distributed computing environments. Proceedings of the international conference workshop on emerging trends in technology (ICWET), Mumbai, India (pp. 370–375).

Nguyen, H. H. C., Dang, H. V., Nguyen, M. N. P., Le, V. S., & Nguyen, T. T. (2015). Deadlock detection for resource allocation in heterogeneous distributed platforms. Recent advances in information and communication technology 2015, Bangkok, Thailand (pp. 85–295).

Pham, N. M. N., Nguyen, T. H., Le, V. S. (2016). Resource Allocation for Virtual Service based on Heterogeneous Shared Hosting Platforms. Proceedings of the eighth Asian conference ACIIDS 2016, Danang, Vietnam (pp. 51–60).
Quan, D. M., Basmadjian, R., Meer, H. D., Lent, R., Mahmoodi, T., Sannelli, D., … Dupont, C. (2012). Energy efficient resource allocation strategy for cloud data centres. 26th International symposium on computer and information sciences on computer and information sciences II, London, England (pp. 133–141).

Randles, M., Lamb, David, & Taleb-Bendiab, A. (2010). A comparative study into distributed load balancing algorithms for cloud computing. IEEE 24th international conference on advanced information networking and applications workshops (WAINA), Perth, Australia (pp. 551–556).

Setzer, T., & Stage, A. (2011). Decision support for virtual machine reassignments in enterprise data centers. Network operations and management symposium workshops (NOMS Wksps), Osaka, Japan (pp. 88–94).

Stillwell, M., Vivien, F., & Casanova, H. (2012). Virtual machine resource allocation for service hosting on heterogeneous distributed platforms. Proceedings of the 2012 IEEE 26th international parallel and distributed processing symposium, Shanghai, China (pp. 786–797).

Vigliotti, A., & Batista, D.M. (2014). Energy-efficient virtual machines placement. Proceedings of the computer networks and distributed systems (SBRC), Florianopolis, Brazil (pp. 1–8).