Multiagent-Based Resource Allocation for Energy Minimization in Cloud Computing Systems

Wanyuan Wang, Yichuan Jiang*, *Senior Member*, *IEEE*, Weiwei Wu

Abstract—Cloud computing has emerged as a very flexible service paradigm by allowing users to require virtual machine (VM) resources on-demand and allowing cloud service providers (CSPs) to provide VM resources via a pay-as-you-go model. This paper considers the CSP's problem of efficiently allocating VM resources to physical machines (PMs) with the aim of minimizing the energy consumption. Traditional energy-aware VM allocations either allocate VMs to PMs in a centralized manner or implement VM migrations for energy reduction without considering the migration cost in cloud computing systems. We address these two issues by introducing a decentralized *m*ulti*a*gent(MA)-based VM allocation approach. The proposed MA works by first dispatching a cooperative agent to each PM to assist the PM in managing VM resources. Then, an auction-based VM allocation mechanism is designed for these agents to decide the allocations of VMs to PMs. The theoretical analyses suggest that this auction-based mechanism has a high performance on reducing energy cost. Moreover, to tackle system dynamics and avoid incurring prohibitive VM migration overhead, a local negotiation-based VM consolidation mechanism is devised for the agents to exchange their assigned VMs for energy savings. We evaluate the efficiency of the MA by using both static and dynamic simulations. The static experimental results demonstrate that the MA can incur acceptable computation time to reduce system energy cost compared with traditional bin packing-based and genetic algorithm-based centralized approaches. In the dynamic setting, the energy cost of the MA is similar to that of benchmark centralized resource consolidation approaches, but the MA largely reduces the migration cost.

Index Terms—Cloud computing systems, resource allocation, energy cost, migration cost, multiagent, negotiation.

1 INTRODUCTION

Cloud computing provides flexible and cost-effective services for enterprises, organizations and individuals running computational and data-intensive applications [1]. Through cloud computing platforms (e.g., Amazon EC2, Google AppEngine, and Microsoft Azure), users can submit their resource (e.g., CPU, memory, storage and network, etc.) request to cloud service providers (CSPs). The CSPs then provide the users the required resource in the form of a virtual machine (VM, acting like a real computer) in exchange for financial remuneration [2]. Generally, an effective VM resource allocation should not only deliver scalable services to satisfy various user requirements with the aim of increasing the CSP's profit [3][4], but also conserve the energy consumption of the physical machines (PMs) used for running users' applications with the aim of decreasing the CSP's cost [5-7]. In this paper, we are mainly concerned with developing energy-aware resource allocation approach of allocating VMs to PMs with the aim of minimizing system energy cost, which is a fundamental problem in cloud computing systems [8-27].

A straightforward idea to make a cloud system energy efficient is to develop energy-proportional PMs, i.e., each PM consumes energy only in proportion to the VM loads it undertakes [8]. For this purpose, many technologies such as using high-quality power supplies and voltage regulation modules, have been introduced to achieve PM energy proportionality [9][10]. However, even though equipped with energy-proportional PMs, the cloud system's energy consumption is far from optimal due to inefficient allocation of VMs to PMs [11-15]. In cloud computing systems, PMs are heterogeneous with various resources and operation costs and VMs are heterogeneous with different resource requirements [16]. An undesirable allocation of allocating the large-size VMs to costly PMs might consume tremendous energy [17-20].

Due to its significance to build green cloud systems, the energy-aware VM resource allocation problem has been studied widely, and a number of approaches have been proposed [11-27]. However, from these approaches, we find there are two aspects that need to improve. First, most of existing approaches assume that there is a central resource manager that can monitor and maintain information about all PMs and VMs and thus can allocate VMs to PMs in a centralized manner [11-14][16-26]. Although centralization can guarantee high system performance, its low robustness with a single point of failure creates a

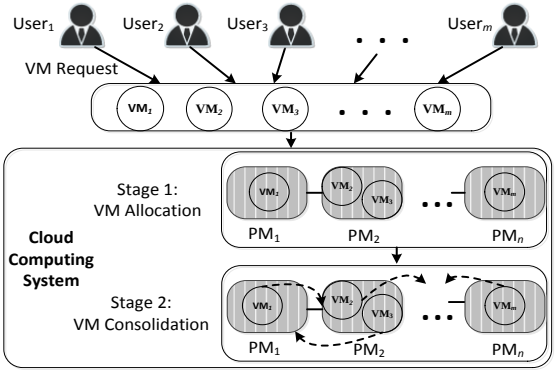


Fig. 1. The multiagent-based resource allocation framework.

*The authors are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China, and also with the Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, China. (*Corresponding author: Yichuan Jiang, email: yjiang@seu.edu.cn).*

Manuscript received 28 March 2015, revised 3 September 2015, accepted 30 November.

vulnerable cloud system [28][29]. Second, because cloud systems are dynamic with dynamic VMs arrival and departure, VM live migration is necessary for resource consolidation. The migration cost (e.g., network traffic cost), occurs when a VM is migrated from one PM to another PM, which is also crucial to the performance of cloud computing systems [30][31]. However, many approaches [15][19][23][24] transfer VMs among PMs without considering the VM migration cost.

To address the above two issues, in this paper we introduce a decentralized *m*ulti*a*gent(MA)-based resource allocation approach by dispatching a cooperative agent to each PM to assist the PM in managing resources. For a set of VM requests newly submitted to the cloud systems, the proposed MA approach allocates these VMs to suitable PMs by the following two sequential stages (Fig. 1 depicts the framework of the MA resource allocation approach):

- ***Auction-Based VM Allocation.*** In the first stage, an auction-based VM allocation mechanism is devised for agents to decide which PM hosts which newly submitted VMs. Theoretical analyses suggest that the auction-based VM allocation mechanism has a high performance guarantee on reducing energy cost compared with the optimal solution.
- ***Negotiation-Based VM Consolidation.*** To tackle system dynamics and avoid incurring prohibitive VM migration overhead, a local negotiation-based VM consolidation mechanism is devised for agents to exchange their assigned VMs for energy cost savings.

We conduct both static and dynamic simulations to evaluate the effectiveness of the MA resource allocation approach. In the static setting with hundreds of VMs, within several seconds, the MA approach can reduce system energy cost significantly compared with benchmark bin packing-based and genetic algorithm-based centralized approaches [13][23][24]. The dynamic experimental results demonstrate that the MA approach can adapt to system dynamics well by consuming as little energy as the centralized and distributed resource consolidation approaches [11][13][15][23][24], but largely reducing the migration cost, showing its great potential for real-world applications.

The remainder of this paper is organized as follows. In Section 2, we provide a thorough review of related work on resource allocation in cloud computing and multiagent systems. In Section 3, we formulate the VM allocation problem with the objective of energy cost minimization. In Section 4, we propose MA-based resource allocation and consolidation mechanisms. In Section 5 we conduct two series of experiments to validate the MA approach's effectiveness in reducing system energy cost. Finally, we conclude our paper and discuss future work in Section 6.

2 Related Work

Generally, from the CSP's perspective, effective (VM)resource allocation should satisfy the following two properties: *i*) allocating the VMs to the users optimally such that the social welfare achieved from the user maximal; and *ii*) allocating the VMs to PMs optimally such that the energy cost produced by the PMs minimal. Therefore, in this section, we first discuss the social-aware and energy-aware resource allocation researches in Section 2.1 and Section 2.2, respectively. Since the main contribution of this paper is to utilize multiagent technology to address the resource allocation problem in cloud computing systems, then finally we briefly review multiagent-based resource allocation in traditional applications in Section 2.3. Fig. 2 depicts the classifications of resource allocation researches in cloud computing systems.

2.1 Social Welfare-Aware Resource Allocation in Cloud Computing Systems

Auction-based resource allocation model has been used as an economic paradigm for the CSPs providing the VM resources to the valuable users [32-38]. In the auction model, the users first submit their request on how many VM resources they require and how much they value the required VMs and then the CSP determines to allocate which VM resources to which users such that social welfare maximal. To maximize social welfare while inducing the users to declare their true private information, Nejad et al. [32] propose a VCG-based truthful mechanism to achieve the optimal social welfare. However, since the social-welfare maximization problem is a NP-hard combinatorial optimization problem [33], VCG mechanism is computation intractable in the large-scale cloud systems. Therefore, the approximation truthful mechanisms with tolerable computation time are more desirable for CSP [34]. Moreover, to deal with the real world dynamic environment, an online truthful mechanism is introduced by [35], which is invoked as soon as a user summits a request or some of the allocated VMs are released and become available. Zhang et al. [36] improve the online truthful mechanism by designing a randomized mechanism that can provide a constant approximation ratio on social welfare and Zhang et al. [37] improve the online mechanism by considering the more flexible bidding language such that a user can also be satisfied if his required resources are accessible during a time period. Although these mechanisms are efficient in achieving desirable social welfare, all of them do not consider the operation cost such as energy cost for running the users' application. Reducing the energy cost not only increases the CSP's net revenue, but also helps build green cloud systems [38].

Huu and Tham [38] first integrate the energy cost factor into the auction model, where they propose a truthful and competitive truthful mechanism to optimize the CSP's net revenue (i.e., social welfare minus energy cost). On the other hand, Xu and Li [6] propose a pricing mechanism to adjust the price optimally to make a tradeoff between the revenue achieved from the users and the energy cost produced by the PMs. All these market-driven mechanisms only focus on allocating how many resources to users and which VMs to which users to maximize CSP's revenue, they do not focus on how to allocate the VM applications to the PMs in the back-end cloud data centers to minimize energy cost. Our study mainly focuses on developing an effective resource allocation approach to minimize system energy cost, which is also a fundamental problem in cloud computing systems [8-27].

2.2 Energy-Aware Resource Allocation in Cloud Computing Systems

The energy-aware resource allocation researches can be further classified into three groups: bin-packing based static resource allocation (i.e., given a set of VMs and PMs, how to allocate the VMs to PMs to minimize PMs' energy cost), energy-aware dynamic resource consolidation (i.e., systems are dynamic with new VM request submitted and old VMs released, how to consolidate system VMs for energy cost savings) and energy and Service

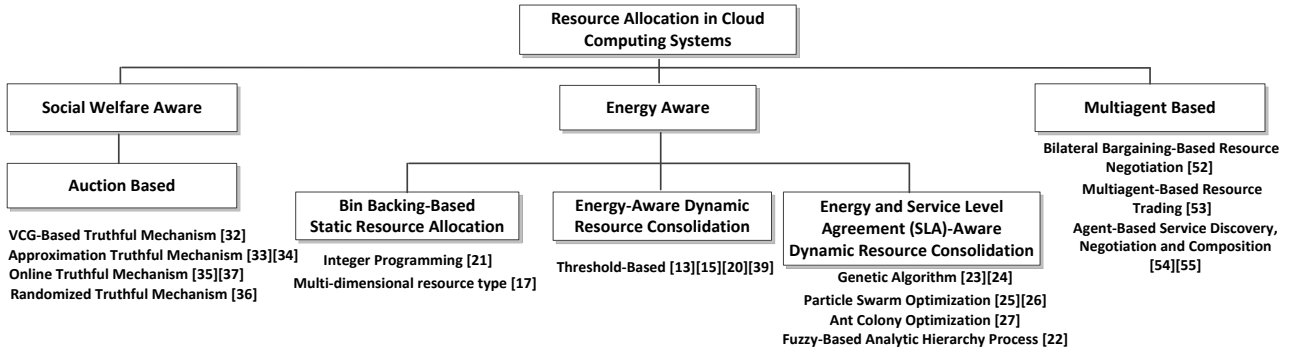


Fig. 2. Classification of resource allocation researches in cloud computing systems.

Level Agreement(SLA)-aware dynamic resource consolidation (i.e., how to consolidate system VMs with the bi-objectives of reducing energy cost and SLA violation).

2.2.1 Bin Packing-Based Static Resource Allocation

Recent studies have shown that PMs, which are used to run VMs, consume a high percentage of the power in cloud computing systems [10]. One natural objective of efficiently allocation of VMs to PMs is to reduce the number of active PMs, which can be called static resource allocation. Intuitively, the static energy-aware resource allocation problem can be modeled as the bin-packing problem, where VMs and PMs are the items and bins, respectively in the bin packing problem. For this transformed bin packing problem with liner usage costs, Cambazard et al. [21] first compute the lower bound of the optimal solution and then design a polynomial cost-based propagation allocation algorithm. By considering the multi-dimensional resource types of PMs, Li et al. [17] present a multi-dimensional space partition model for the multi-dimensional bin packing problem. Based on this model, they then propose a balance VM allocation approach to alleviate the imbalanced utilization of the multi-dimensional resources and thus lower the energy consumption. These static resource allocation approaches are all restricted to the one-shot or offline setting, not targeting on an online setting where VMs arrive and depart the cloud system dynamically. In this kind of dynamic cloud systems, VM consolidation is very necessary to reduce system energy cost [11][15][27][39].

2.2.2 Energy-Aware Dynamic Resource Consolidation

Live migration technology [40], allowing a VM to be migrated from one PM to another PM, has proved to be effective in addressing resource consolidation [13][27]. Motived by the classical online bin packing approach, Song et al. [19] propose an adaptive resource consolidation approach to minimize the number of active PMs. During VM migration, on the one hand, the system should move VMs on *source* PM with a low resource utilization to another *target* PM, thus allowing the *source* PM to switch off without consuming any power. On the other hand, the system should also avoid the *target* PM over-utilized. To achieve these goals, a threshold-based resource consolidation approach has been investigated [13][15][20][39]. This approach works by first predetermining two thresholds, the *high threshold* t_h and the *low threshold* t_l. When the resource utilization of a PM p_i, exceeds t_h, the system will transfer some VMs on p_i to another PM for hotspot avoidance. When the resource utilization of p_i falls below t_l, the system will migrate all of the VMs on p_i to another *target* PM for energy saving. In dynamic cloud systems, to predict the two thresholds t_h and t_l precisely, these researches all assume that there exists a central manager that monitors and maintains information about all PMs and VMs. Our approach does not need such a central manager, instead allowing the PMs to manage resources in a distributed manner, thereby improving system robustness. Moreover, another deficiency of these dynamic energy-aware resource consolidation approaches is that they only focus on the advantage of live migration on reducing energy cost, do not consider its negative effect on violating Service Level Agreement (SLA), such as reducing system throughput and increasing system response time [41][42].

2.2.3 Energy and SLA-Aware Dynamic Resource Consolidation

To reduce system energy consuming while reducing SLA violation, Verma et al. [11] and Ardagna et al. [12] first transform this bi-objective (i.e., energy cost minimization and SLA violation minimization) problem to a single objective problem by setting a tradeoff weight between these two objectives. And then they model this single objective problem as a mixed integer nonlinear programming problem, which can be solved by the competitive approximate algorithms [11][12]. One challenge of transforming the bi-objective problem to a single objective problem is how to set the tradeoff weight between the two objectives. Kord and Haghighi [22] exploit the fuzzy-based Analytic Hierarchy Process (AHP) method to determine the tradeoff weight between the multiple objectives. The main idea behind the fuzzy-AHP model is that the system first determines the relative importance of each objective by pairwise comparison. And then determine the related intermediate priorities of these candidate destination PMs with respect to each objective. Finally, the global priority of each candidate PM is determined by summing all priorities with respect to each objective [43].

Another efficient way to tackle the bi-objective resource allocation problem is to utilize the evolutionary computation (EC) algorithms, including the Genetic Algorithm (GA) [23][24], swarm intelligence algorithms such as Particle Swam Optimization (PSO)[25][26][1] and Ant Colony Optimization (ACO) [27]. For example, in the GA algorithm [23], a "chromosome" represents an allocation solution of VMs to PMs. To begin, GA randomly generates a population of potential chromosomes and evaluates the fitness values (i.e., objective value) of these candidate solutions. And in the second step, the desirable chromo-

[1] For more details on evolutionary computation-based resource allocation approaches in cloud systems, we refer interested readers to the recent survey paper [44] and the references therein.

somes with higher fitness values (e.g., produce little energy cost and violate SLA little) are selected as the parent chromosomes and to crossover the next generation chromosomes. The PSO has a similar optimization process with GA, where in the first step, a set of particles that represent VM allocation solution are randomly initialized. In the second step, based on the local best position and the global best position, each particle improves its fitness value by updating its current position in the population [25]. In the ACO algorithm, a VM migration plan $s=(p_s,v,p_d)$, which means the VM v is migrated from the source PM p_s to the destination PM p_d, is modeled as the edge connected the cities in traveling salesman problem. Then the ants will deposit some pheromone on the migration plan s if s not only reduces energy cost but also guarantees SLA. Iteratively, the migration plan associated with higher pheromone concentration will constitute the global VM migration solution [27]. Although these nature-inspired EC algorithms and fuzzy control-based heuristics are efficient in improving cloud system performance such as reducing energy cost and guaranteeing SLA performance, their efficiency depends much on system parameters such as the mutation probability in GA, the acceleration coefficients of the local and global best position in PSO, the pheromone evaporation rate in ACO, the evolution termination condition in GA, ACO, and PSO, and the importance intensity between any pair of objectives in fuzzy-based AHP approach. In contrast, our multiagent approach depends less on system parameters, making it more practical for the real-world applications.

2.3 Multiagent-Based Resource Allocation

Multiagent technology, which is derived from distributed artificial intelligence (DAI), has shown its effectiveness in addressing distributed system problems [45]. Example applications include coalition formation in the business-to-business (B2B) domain [46], routing in robotics [47], mobile agent-based load balancing in grids [48], negotiation-based task allocation in grids and social networks [49][50], and agent-based modeling for social networks [51]. Recently multiagent technology to tackle cloud resource allocation has received increasing attention. For example, An et al. [52] introduce a bilateral bargaining-based resource negotiation mechanism for users accessing necessary resources. Zhao et al. [53] propose a multiagent-based resource trading protocol to trade efficient and fair resource among selfish users in community cloud systems. Sim [54] presents a systematic agent-based cloud computing model, where agents are developed to support service discovery, service negotiation and service composition. The work of Sim [54] is further investigated by Chen et al. [55] by extending contract net technology to maximize a cloud system's throughput.

Although these approaches are efficient in addressing traditional resource allocation problems, they are inadequate for VM resource allocation in network cloud systems. In network cloud systems, PMs are always interconnected by a communication network. Because of arbitrary negotiation and task migration among PMs, the above approaches [45-55] will consume prohibitive network bandwidths, thereby violating SLA largely. This paper proposes an efficient local resource negotiation mechanism that limits agents' coordination domain locally. Under this local coordination domain constraint, an efficient negotiation-based VM consolidation mechanism is proposed to reduce system energy cost while incurring tolerable migration overhead.

3 Problem Description

We consider a cloud system $CS=<P, E>$ consisting of a set of PMs $P=\{p_1,p_2,\ldots,p_m\}$ interconnected by a communication network and $\forall(p_i,p_j)\in E$ indicates that p_i and p_j can communicate with each other through only one switch. Denoted by $d(p_i,p_j)$ the communication distance between PMs p_i and p_j and $d(p_i,p_j)$ is computed as the number of switches along the shortest path between p_i and p_j. Let $\Theta=\{\theta_1,\theta_2,\ldots,\theta_n\}$ be the set of VM resource (e.g., CPU, memory, storage, bandwidth, etc.) required by users. For simplicity, in this study we consider only one type of resource requirement (e.g., CPU) and denoted by r_i the amount of resources required by VM $\theta_i\in\Theta$. Each PM owns a number of resources that are capable of running multiple VMs and denoted by c_i the amount of resources at PM p_i. To satisfy the submitted VMs' resource requirements, some suitable PMs should be selected to host them, which can be called the VM allocation problem. A feasible VM allocation $\{\Theta(p_1),\Theta(p_2),\ldots,\Theta(p_m)\}$ is defined as a mapping of PM $\forall p_i\in P$ to a set of VMs $\Theta(p_i)$, which must satisfy the following two conditions:

1) Each VM is allocated to at least one PM and no VM is allocated to more than one PM, i.e.,
$$\bigcup_{p_i\in P}\Theta(p_i)=\Theta,\quad \Theta(p_i)\bigcap\Theta(p_j)=\varnothing,\forall 1\le i,j\le m,i\ne j \quad (1)$$
2) For each PM, the total resource requirements of its hosted VMs do not exceed its available resources, i.e.,
$$\sum_{\theta_j\in\Theta(p_i)} r_j \le c_i,\quad \forall 1\le i\le m \quad (2)$$

In addition to satisfying the above two conditions, the ultimate objective of VM allocation is to minimize the total system PMs' energy cost. Although many technologies have been used to develop energy-proportional PMs [9][10], each PM is far from energy-proportional [5][13]. Therefore, to simulate a more practical application, we can model the energy cost function of each PM p_i as

$$e_i(u_i)=\begin{cases}\alpha_i\cdot\lambda_i+(1-\alpha_i)\cdot\lambda_i\cdot u_i, & u_i>0;\\ 0, & u_i=0.\end{cases} \quad (3)$$

where λ_i is the maximum energy consumed when p_i is fully utilized, α_i is the fraction of the maximum energy consumed when p_i is idle and u_i is the resource utilization of p_i, which is computed as $u_i=\sum_{\theta j\in\Theta(pi)} r_j/c_i$. When p_i is active for running VMs (i.e., $u_i>0$), its energy cost $e_i(\cdot)$ then is an affine function of its resource utilization u_i. Otherwise, when p_i is idle (i.e., $u_i=0$), p_i should be turned off, avoiding consuming any energy cost. Now, we will give the formal definition of the VM allocation problem.

Definition 1. VM Allocation Problem. Given a set of VMs $\Theta=\{\theta_1,\theta_2,\ldots,\theta_n\}$ and a set of PMs $P=\{p_1,p_2,\ldots,p_m\}$, the VM allocation problem is to determine the optimal allocation of VMs Θ to PMs P with the minimum energy cost E, i.e.,

$$\text{Minimize } E=\sum_{i=1}^{m} e_i(u_i)$$

$$\text{Subject to:}$$

$$\sum_{j=1}^{n} r_j x_{ij} \le c_i,\quad \forall i\in 1,\ldots,m$$

$$u_i=\sum_{j=1}^{n} r_j x_{ij}/c_i,\quad \forall i\in 1,\ldots,m$$

$$x_{ij}\in\{0,1\},\sum_{i=1}^{m} x_{ij}=1,\quad \forall i=1,\ldots,m,\ j=1,\ldots,n$$

The variable $x_{ij} \in \{0,1\}$ is the decision variable, where $x_{ij}=1$ indicates VM θ_j is allocated to PM p_i; otherwise $x_{ij}=0$.

It is not hard to determine that solve this VM allocation problem is NP-hard because the traditional NP-hard Bin-Packing problem [56] is a special case of this problem by setting $\alpha_1=\alpha_2=\ldots=\alpha_m=\alpha$, $\lambda_1=\lambda_2=\ldots=\lambda_m=\lambda$ and $c_1=c_2=\ldots=c_m=c$. Therefore, it is very essential to devise efficient polynomial approximation algorithms.

4 MULTIAGENT-BASED RESOURCE ALLOCATION

We formulate the distributed multiagent-based resource allocation approach as follows. First, we dispatch a cooperative agent a_i to each PM p_i. These agents $A=\{a_1, a_2,\ldots,a_m\}$ are deployed to assist the PMs in managing resources (hereafter, the terms "agent" and "PM" are used interchangeably). And then we devise the coordination mechanism for these agents to make decisions on which PMs should host which VMs in pursuit of the energy cost minimization. For a set of newly submitted VMs, this multiagent-based VM allocation approach of allocating these VMs to PMs mainly consists of the following two complementary stages:

- **Auction-Based VM Allocation.** An auction-based mechanism is devised for the agents to decide the allocation of the submitted VMs to PMs (Section 4.1).
- **Negotiation-Based VM Consolidation.** A local negotiation-based VM consolidation mechanism is devised for the agents to exchange their assigned VMs for energy cost saving (Section 4.2).

4.1 Auction-Based VM Allocation

In the market-oriented auction architecture [46], the bidders represent the commodity demanders that have a pressing need for the commodities. They express their needs by submitting bids on the price they would like to spend on the commodities. The proposed auction-based VM allocation mechanism works as described in Algorithm 1, where agents are modeled as bidders and VMs are modeled as commodities. In Algorithm 1, initially, all newly submitted VMs Θ are unallocated. At each bidding round (Steps 2~8), each agent a_i only bids for a single VM and it always bids for the largest unallocated VM that it is capable of hosting (Step 3). After bidding for the target VM θ_{ai}^*, each agent a_i broadcasts its bid B_i to all other agents for winner determination. A bid $B_i=<p_i,\lambda_i/c_i>$ consists of the PM identity p_i and its cost-capacity ratio λ_i/c_i (Step 5). After all bids are broadcasted, all of the agents send a winner acknowledgment message <*Ack*> to the winner agent that has the minimum cost-capacity ratio (Step 7). In the event of a tie, the agent that has the smallest index is selected as the winner. In step 8, the agent a_i that receives acknowledgment from all other agents wins the current round bidding. The winner agent a_i then is responsible for running its target VM θ_{ai}^*, informing all other agents that θ_{ai}^* has been allocated. The above bidding process (Steps 2~8) proceeds round by round until all VMs are allocated (Step 1).

Besides simplification, another important property of Algorithm 1 is its efficiency on reducing energy cost, which can be measured by the *approximation ratio* [57].

Definition 2. Approximation Ratio. For a cloud system with a list of VMs Θ to and a set of PMs P, let $A(\Theta,P)$ and $OPT(\Theta,P)$ be the system energy cost generated by the approximation algorithm A and the optimal solution OPT, respectively. The approximation ratio $S(A)$ of algorithm A then can be defined as:

$$S(A) = \sup\{A(\Theta,P)/OPT(\Theta,P)\} \quad (4)$$

Before presenting the approximation ratio of Algorithm 1 in Theorem 1, we first present a simple proposition that is helpful to prove Theorem 1.

Proposition 1. Assume two positive integer sets $X=\{x_1,x_2,\ldots,x_m\}$ and $Y=\{y_1, y_2,\ldots,y_n\}$ satisfy $x_i<y_j$, $\forall 1\le i\le m$ and $1\le j\le n$. If we randomly pick k_1 elements $X_{k1}=\{x_1, x_2,\ldots,x_{k1}\}$ from X, k_2 elements $Y_{k2}=\{y_1,y_2,\ldots,y_{k2}\}$ from Y, and X_{k1} and Y_{k2} satisfy $\sum_{x_i\in X_{k_1}} x_i \ge \sum_{y_i\in Y_{k_2}} y_i$, then we have $k_1>k_2$.

Theorem 1. The approximation ratio of Algorithm 1 $S(A1)=1+\lambda_{max}/\lambda_{min}$, where $\lambda_{max}=\max\{\lambda_i, 1\le i\le m\}$ and $\lambda_{min}=\min\{\lambda_i, 1\le i\le m\}$.

Proof. We first summarize Algorithm 1 in a centralized manner: first, the PMs $P=\{p_1,p_2,\ldots,p_m\}$ are ranked in increasing order of their cost-capacity ratio λ_i/c_i, i.e., $\lambda_1/c_1\le\lambda_2/c_2\le\ldots\le\lambda_m/c_m$ and the VMs $\Theta=\{\theta_1,\theta_2,\ldots,\theta_n\}$ are ranked in decreasing order of their size, i.e., $r_1\ge r_2\ge\ldots\ge r_n$. Then, we allocate the VMs to PMs by a greedy method. This centralized greedy method works as follows: select the first PM p_1 that has the minimum cost-capacity ratio and fill p_1 with VMs one by one in order of these VMs' rank. If at a certain step, a VM θ_x cannot be allocated to p_1 due to its capacity constraint, the successive VM θ_y ($y>x$) that has a smaller size is considered. This process of filling p_1 with VMs continues until either p_1 has been fully utilized or no unallocated VMs that can be allocated to p_1. After the allocation of VMs to p_1 finishes, a similar process of allocating VMs to p_2 repeats. This greedy procedure proceeds until all VMs have been allocated.

Now, we will present the approximation ratio of Algorithm 1. On the one hand, from the perspective of the optimal solution, in the best case, the optimal solution will only use the first k_1 PMs with the minimal cost-capacity ratios, i.e.,

$$k_1=\min\{j\in\{1,\ldots,m\}:\Sigma_{1\le i\le j}c_i\ge R\} \quad (5)$$

where R is the sum of resources required by all VMs, i.e., $R=\Sigma_{\theta j\in\Theta}r_j$.

On the other hand, from the perspective of Algorithm 1, assume that Algorithm 1 uses the first k_2 PMs to host

Algorithm 1. Auction-Based VM Allocation

/$\Theta=\{\theta_1,\theta_2,\ldots,\theta_n\}$ the set of VMs that need to be allocated.*/*

1. **while** ($\Theta\neq\emptyset$) **do**
2. **for** $a_i\in A$ **do**
3. Bid the VM $\theta_{a_i}^* = \arg\max_{\theta_j\in\Theta}\{r_j \mid c_i - \sum_{\theta_k\in\Theta(p_i)} r_k - r_j \ge 0\}$.
4. **if**($\theta_{ai}^*\neq\emptyset$), **then**
5. Broadcast the bid $B_i=<p_i,\lambda_i/c_i>$ to all other agents.
6. **end for**
7. Each agent receives and stores all bids, and then sends an acknowledge message <*Ack*> to the agent a^* that has the minimal cost-capacity ratio, i.e., $\lambda^*/c^*=\min\{\lambda_k/c_k, 1\le k\le m\}$.
8. The agent a_i that receives acknowledgement from all other agents becomes the winner agent and is responsible for running its target VM θ_{ai}^*. After θ_{ai}^* has been allocated, the winner a_i informs all other agents that θ_{ai}^* has been allocated and removes θ_{ai}^* from the VM list Θ.
9. **end while**

system VMs. For these k_2 PMs, denoted by l_i the VM loads on p_i (i.e., $l_i=\sum_{\theta_j\in\Theta(p_i)} r_j$, $1\le i\le k_2$) and w_i the residual capacity of p_i (i.e., $w_i=c_i-l_i$, $1\le i\le k_2$). In Algorithm 1, the next PM is selected to host VMs if and only if the previous PM cannot host any VM. Then, we have

$$w_i<l_j, \forall 1\le i\le k_1,\ k_1+1\le j\le k_2 \quad (6)$$

Next, we divide the proof into two cases according to whether the k_1th PM in the optimal solution is fully utilized or not.

Case 1: the k_1th PM is fully utilized. In this case, for Algorithm 1, the VMs that have not been allocated to the first k_1 PMs should be allocated to the successive (k_2-k_1) PMs (i.e., $\{p_{k1+1},\ldots,p_{k2}\}$), indicating that

$$\sum_{1\le i\le k_1} w_i = \sum_{k_1+1\le j\le k_2} l_j \quad (7)$$

Case 2: the k_1th PM is not fully utilized. In this case, Let l_{k1}(Opt) and l_{k1}(A1) denote the VM loads on the k_1th PM in the optimal solution and Algorithm 1, respectively. Then, we can derive that

$$\sum_{1\le i\le k_1} w_i > \sum_{1\le i\le k_1-1} w_i + l_{k1}(\mathrm{Opt}) - l_{k1}(A1) = \sum_{k_1+1\le j\le k_2} l_j \quad (8)$$

Combing inequalities (7) and (8), we can derive that

$$\sum_{1\le i\le k_1} w_i \ge \sum_{k_1+1\le j\le k_2} l_j \quad (9)$$

Combing inequalities (6) and (9) and Proposition 1, we can determine that the additional number of used PMs in Algorithm 1 (i.e., k_2-k_1) is less than the number of PMs used in the optimal solution (i.e., k_1), that is k_2-k_1<k_1.

Up to this point, we can determine that the approximation ratio of Algorithm 1 is

$$S(A1)=\sup\frac{A1(\Theta,P)}{OPT(\Theta,P)}\le\frac{OPT(\Theta,P)+\sum_{k_1<i\le k_2}e_i(u_i)}{OPT(\Theta,P)}$$
$$\le 1+\frac{\sum_{k_1<i\le k_2}\alpha_i\lambda_i+(1-\alpha_i)\lambda_i u_i}{\lambda_{\min}k_1}\le 1+\frac{\sum_{k_1<i\le k_2}\lambda_i}{\lambda_{\min}k_1}\le 1+\frac{\lambda_{\max}}{\lambda_{\min}}$$

Therefore, we have Theorem 1. □

In addition to analyzing the approximation ratio of Algorithm 1, in cloud computing systems, the efficiency of the distributed algorithm should also be evaluated in terms of computation and communication complexities.

Computation and communication complexities of the distributed auction-based VM allocation algorithm (i.e., Algorithm 1). Recall that in Algorithm 1, exactly one VM is allocated at each auction round. Therefore, n auction rounds are needed to allocate all n VMs, where n is the number of VMs. At each round, each agent takes $O(n)$ operations to compute the best bid (Step 3). In step 5, for each agent, m-1 bid messages need to be sent to all other agents, where m is the number of PMs. Next, in step 7, each agent takes $O(n)$ computations to select the optimal bid with the minimal cost-capacity ratio. For each of the m agents, it needs to send a winner acknowledgement to the winner agent. Finally, in Step 8, the winner agent needs to send m-1 messages to all other agents to inform them that the target VM it bids for has been allocated. Therefore, the total computation of Algorithm 1 is $O(n(n+n))=O(n^2)$ and the total communication messages are $O(n(m(m\text{-}1)+m\text{-}1+m\text{-}1))=O(nm^2)$. Notice that both of the bid and acknowledgement messages contain at most two real numbers, which can be coded by several bytes. Hence, Algorithm 1 consumes little network bandwidth.

4.2 Negotiation-Based VM Consolidation

4.2.1 The Reason for Negotiation-Based VM Consolidation

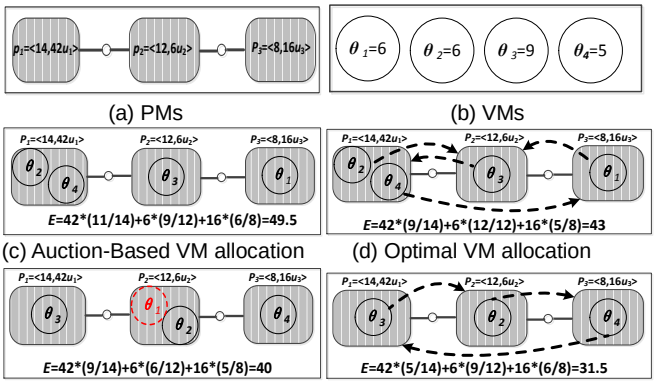


Fig. 3. Advantages of the negotiation-based VM consolidation.

Although the auction-based VM allocation mechanism has several desirable properties, there are two deficiencies that can be further improved. In the following, we use two illustrative examples to highlight the advantages of the negotiation-based VM consolidation mechanism in addressing these two deficiencies.

The first deficiency is that the auction-based VM allocation does not always achieve the optimal solution. Consider, for example (**Example 1**), the simple VM allocation problem presented in Fig. 3. In Fig. 3(a), there is a cloud system consisting of three interconnected PMs p_1, p_2 and p_3, where p_1 and p_2 can communicate directly and p_2 and p_3 can communicate directly. These PMs are denoted by p_1=<14,42u_1>, p_2=<12,48u_2> and p_3=<8,16u_3>, where the first value denotes the capacity and the second function denotes the energy cost model (for convenience, here we assume α_i=0, $\forall 1\le i\le 3$) and u_i is p_i's resource utilization. Now assume that there are four newly submitted VMs $\{\theta_1,\theta_2,\theta_3,\theta_4\}$ and their required resources are r_1=6, r_2=6, r_3=9 and r_4=5, respectively (Fig. 3(b)). By the auction-based VM allocation mechanism, VM θ_3 is allocated to PM p_2, θ_1 is allocated to p_3 and θ_2 and θ_4 are allocated to p_1. This allocation $\{\{\theta_2,\theta_4\},\{\theta_3\},\{\theta_1\}\}$ produces 42×(11/14)+6×(9/12)+16×(6/8)=49.5 units energy cost (Fig. 3(c)). The optimal allocation $\{\{\theta_3\},\{\theta_1,\theta_2\},\{\theta_4\}\}$ can be achieved from the auction-based allocation $\{\{\theta_2,\theta_4\},\{\theta_3\},\{\theta_1\}\}$ by migrating θ_3 from p_2 to p_1, θ_2 from p_1 to p_2, θ_4 from p_1 to p_3 and θ_1 from p_3 to p_2 simultaneously, which only produces 42×(9/14)+6×(12/12)+16×(5/8)=43 units energy cost (Fig. 3(d)). This migration process can be achieved easily by two sequential VM exchange: the exchange of θ_3 and $\{\theta_2, \theta_4\}$ between PMs p_1 to p_2 and the exchange of θ_4 and θ_1 between p_2 and p_3.

The second deficiency is that this static auction-based VM allocation cannot adapt to system dynamics where VMs arrive and depart the system dynamically. For example (**Example 1 continued**), at a certain time-slot of the optimal allocation in Example 1, the application of VM θ_1 at PM p_2 has been satisfied and departs the system, where the system energy cost becomes 42×(9/14)+6×(6/12)+16×(5/8)=40 (Fig. 3(e)). In this case, to re-optimize the allocation of VMs for energy reduction, it is natural to invoke Algorithm 1 again by reallocating all system VMs. This method, however, might generate a tremendous VM migration cost because it does not consider the current VM allocation. Now, consider three simple VM transfers among PMs p_1, p_2 and p_3 by migrating θ_3 from p_1 to p_2, θ_2 from p_2 to p_3 and θ_4 from p_3 to p_1. These VM consolidations reduce the energy cost to

42×(5/14)+6×(9/12)+16×(6/8)=31.5 (Fig. 3(f)).

Therefore, it is very necessary to devise a negotiation-based VM consolidation mechanism by allowing agents to exchange their assigned VMs to save energy cost and address system dynamics.

4.2.2 A Local Negotiation-Based VM Consolidation Mechanism

With the increasing development of virtualization technology, on one hand, VM live migration has been verified as effective in reducing energy consumption in cloud systems [20]. On the other hand, VM live migration might have a negative effect on system performance such as increasing network delay [41]. To reduce system energy cost and avoid incurring prohibitive migration overhead, we propose a local VM consolidation mechanism by allowing a VM to migrate from one agent to another agent that within the same negation domain.

Definition 3. Negotiation Domain. The negotiation domain of each agent a_i is defined as $D_i=\{a_j|d(a_i,a_j)\le\rho\}$, where $\rho\in\mathbb{N}$ is the predetermined negotiation radius parameter, meaning that a_i can only negotiate with agents within the limited communication distance ρ.

Throughout this paper, we refer to any pair of agents are negotiable if these two agents are within the same negotiation domain. Now we are ready to formalize the local negotiation-based VM consolidation mechanism. With respect to negotiation, we mean that the agents make contracts on exchanging their assigned VMs. According to the number of agents involved in the contract, we can classify the contracts into two main families [58]:

- ***Swap Contract***, where only two agents are involved by allowing the first agent to transfer a set of VMs to the second agent and the second agent to transfer another set of VMs to the first agent in return (Section 4.2.2.1).
- ***Cluster Contract***, where a cluster of agents (more than two) are involved by allowing the VMs to be transferred among multiple agents within the negotiation domain (Section 4.2.2.2).

4.2.2.1 Swap Contract

By referring to the related definitions presented in [47], we first define single VM *out* and *in* contract.

Definition 4. Out and In Contract. Given an agent $a_i\in A$, an *out* contract $out(a_i,\theta_x,a_j)$ is defined as a single VM migration that migrates VM $\theta_x\in\Theta(a_i)$ from a_i to another agent a_j and an *in* contract $in(a_i,\theta_x,a_j)$ is defined as a single VM migration that migrates $\theta_x\in\Theta(a_j)$ from a_j to a_i.

Definition 5. Swap Contract. A swap contract $sc(a_i,\Theta_{i,j},a_j,\Theta_{j,i})$ is defined as a union of multiple *out* and *in* contracts between a_i and a_j, i.e.,

$$sc(a_i,\Theta_{i,j},a_j,\Theta_{j,i})=\bigcup_{\theta_k\in\Theta_{i,j}} out(a_i,\theta_k,a_j)\bigcup_{\theta_k\in\Theta_{j,i}} in(a_i,\theta_k,a_j) \quad (10)$$

where $\Theta_{i,j}$ (resp. $\Theta_{j,i}$) represents the set of VMs that agent a_i (resp. a_j) migrates to agent a_j (resp. a_i).

A swap contract is ***feasible*** if and only if it satisfies the capacity constraint, i.e., after executing the VM swap contract $sc(a_i,\Theta_{i,j},a_j,\Theta_{j,i})$, the VM loads of agents a_i and a_j do not exceed their own capacities. Throughout the paper, all contracts are assumed feasible unless noted specifically.

Definition 6. Swap Path. A swap path *SP* is a set of feasible swap contracts that changes a VM allocation $\{\Theta(a_i)\}_{1\le i\le m}$ to another VM allocation $\{\Theta'(a_i)\}_{1\le i\le m}$.

Fig. 4 shows the VM swap graph of Example 1, where

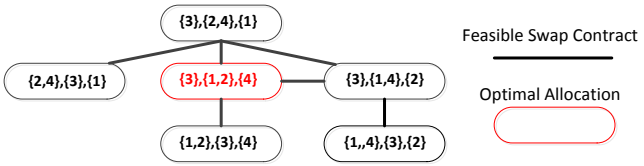


Fig. 4. Feasible swap graph, where negotiation radius ρ=1. A feasible swap path from {{2,4},{3},{1}} to the optimal allocation {{3},{1,2},{4}} is {{2,4},{3},{1}}→{{3},{2,4},{1}}→{{3},{1,2},{4}}.

each agent's negotiation domain is constrained within its direct neighbors, i.e., ρ=1. In Example 1, we can observe that for any allocation, a swap path exists that can lead this allocation to the optimal allocation {{3},{1,2},{4}}. Therefore, we conjecture that the local swap contract has a very inspiring advantage in reducing energy cost, shown in Theorem 2. Before presenting Theorem 2, we first show an interesting property of the local swap contract, which is helpful in proving Theorem 2.

Proposition 2. Assume that there are m PMs $\{p_i|1\le i\le m\}$ and m VMs $\{\theta_i|1\le i\le m\}$. Initially, the VM θ_i is allocated to PM p_i, i.e., the initial allocation $\psi=\{\{\theta_1\},\{\theta_2\},\ldots,\{\theta_m\}\}$ is feasible. If there exists a feasible cyclic swap path $SP=\{sc(a_1,\theta_1,a_2,\emptyset), sc(a_2,\theta_2,a_3,\emptyset),\ldots, sc(a_m,\theta_m,a_1,\emptyset)\}$ (i.e., agent a_i migrates its assigned VM θ_i to its logically next agent $a_{(i+1)\%m}$, $1\le i\le m$, $X\%Y$ denotes the remainder of X/Y) that changes this initial allocation ψ to another allocation $\psi'=\{\theta_m,\theta_1,\ldots,\theta_{m-1}\}$, then we can derive that at allocation ψ, there exists at least one swap contract $sc(a_i,\theta_i,a_{(i+1)\%m},\theta_{(i+1)\%m})$ between a_i and $a_{(i+1)\%m}$ ($1\le i\le m$) for exchanging their assigned VMs θ_i and $\theta_{(i+1)\%m}$.

Proof. Because the initial allocation ψ is feasible, we have that each PM p_i's capacity c_i is greater than their assigned VM θ_i, i.e., $c_i\ge r_i$, $1\le i\le m$. Furthermore, because the cyclic swap path *SP* is feasible, we also have that $c_{(i+1)\%m}\ge r_i$, $1\le i\le m$. Using reductio ad absurdum, we assume that for any pair of agents a_i and $a_{(i+1)\%m}$, it is not feasible for them to change their assigned VMs, i.e.,

$$\begin{cases}(c_1-r_2)(c_2-r_1)<0\\(c_2-r_3)(c_3-r_2)<0\\\quad\vdots\\(c_1-r_m)(c_m-r_1)<0\end{cases} \quad (11)$$

On one hand, from $(c_i-r_{i+1})(c_{i+1}-r_i)<0$ and $c_{i+1}\ge r_i$, $\forall 1\le i\le m-1$, we can derive that $c_i<r_{i+1}$, $\forall 1\le i\le m-1$. Consider the fact that $c_{i+1}\ge r_{i+1}$, $1\le i\le m-1$, we have that $c_1<c_2\ldots<c_m$. On the other hand, from $(c_1-r_m)(c_m-r_1)<0$, $c_1\ge r_m$ and $c_1\ge r_1$, we can derive that $c_1>c_m$, which is contradictory to the conclusion of $c_1<c_m$. Therefore, assumption (11) does not hold and we have this proposition. □

Proposition 3. Assume a VM allocation problem where global communication is permitted (i.e., any pair of PMs can communicate with each other). Given a non-optimal allocation $\psi=\{\Theta(a_i)\}_{1\le i\le m}$ that does not have any feasible swap contract among any pair of agents, then there does not exist a swap path *SP* that can change ψ to another feasible allocation.

Proof. According to the proposition suppose, we have that at this allocation ψ, for any pair of agents a_i and a_j and for any VM set $S_i\subseteq\Theta(a_i)$ on a_i and VM set $S_j\subseteq\Theta(a_j)$ on a_j, the exchange of S_i and S_j is not feasible, i.e., $\forall S_i\subseteq\Theta(a_i)$, $S_j\subseteq\Theta(a_j)$:

$$(c_i-\sum_{\theta_k\in\Theta(a_i)\setminus S_i} r_k-\sum_{\theta_k\in S_j} r_k)\cdot(c_j-\sum_{\theta_k\in\Theta(a_j)\setminus S_j} r_k-\sum_{\theta_k\in S_i} r_k)<0 \quad (12)$$

Using reductio ad absurdum, suppose that there exists a swap path SP=$\{\{a_i,\theta_k,a_j,\emptyset\},\ldots\{a_z,\theta_z,a_y,\emptyset\}\}$ that can change ψ to another feasible allocation ψ'. For the swap contracts along the swap path SP, we conjecture that any agent $a_i \in A_{sp}$ (A_{sp} indicates the agents involved in the swap path SP), involves at least a pair of *out* and *in* swaps (**Conclusion 1**). We prove this conclusion as follows: on the one hand, for the agent $a_i \in A_{sp}$ that only has *out* swaps, we can delete a_i as well as its *out* swaps from the swap path SP. The remaining swap path $SP \setminus \{a_i\}$ is still feasible because deleting a_i's *out* swaps can alleviate other agents' VM loads. On the other hand, for the agent $a_i \in A_{sp}$ that only has *in* swaps, which indicates that there is a feasible *in* swap that other involved agent $a_j \in A_{sp}$ can transfer some VMs to a_i. However, assumption (12) indicates this kind of *in* swap does not exist. Based on Conclusion 1, we can further derive that there exists at least one cyclic swap path CSP within SP, i.e., $CSP \subseteq SP$ (**Conclusion 2**). We can construct such CSP as follows: start from any involved agent $a_i \in A_{sp}$ and add its out swap $out(a_i,\theta_x,a_j)$ to CSP. Then, we proceed to the *out* swap's (i.e., $out(a_i,\theta_x,a_j)$) destination agent a_j and add a_j's out swap $out^*(a_j,\theta_x,a_k)$ to CSP. If the destination agent a_k of the swap out^* has emerged in CSP, then this cyclic swap path CSP is identified. Otherwise, proceed to deal with the destination agent a_k of the swap out^* with the same procedure. Combing Conclusion 2 and Proposition 2, we can finally determine that there exist at least one swap contract between any involved agents $a_i \in A_{sp}$ and $a_i \in A_{sp}$, which contradicts assumption (12). □

Theorem 2. Given a VM allocation problem where global communication is permitted, then for any non-optimal allocation $\{\Theta(a_i)\}_{1\le i\le m}$, there always exists a swap path that changes $\{\Theta(a_i)\}_{1\le i\le m}$ to the optimal allocation $\{\Theta^*(a_i)\}_{1\le i\le m}$ with the minimum energy cost.

Proof. To derive this theorem, we only need to prove that the case that for any non-optimal allocation $\{\Theta(a_i)\}_{1\le i\le m}$, there exists a swap contract that can change $\{\Theta(a_i)\}_{1\le i\le m}$ to another feasible allocation $\{\Theta'(a_i)\}_{1\le i\le m}$. By contradiction, Assume that there exists a non-optimal allocation $\psi=\{\Theta(a_1),\ldots,\Theta(a_n)\}$ that does not have any feasible swap contract among any pair of agents. For this allocation ψ, Proposition 3 indicates that there does exist a swap path SP that can change ψ to another feasible allocation, which means that $\{\Theta(a_i)\}_{1\le i\le m}$ is the only feasible allocation as well as the optimal allocation. Thus, for any non-optimal allocation $\{\Theta(a_i)\}_{1\le i\le m}$, there exists at least one swap contract that can change $\{\Theta(a_i)\}_{1\le i\le m}$ to another feasible allocation $\{\Theta'(a_i)\}_{1\le i\le m}$ and, thus it can achieve the optimal allocation along certain swap path. □

Theorem 2 suggests that theoretically, if global communication is allowed, the local swap contract is sufficiently effective to optimize system performance in reducing energy cost. However, in addition to feasibility satisfaction, the swap contract should also have the monotonicity property, that is, each VM swap contract should reduce energy cost. Why is monotonicity important? The cloud system might be halted arbitrarily due to system maintenance and non-monotonic VM consolidation mechanisms risk being terminated at highly inefficient allocation that consumes a large amount of energy.

Definition 7. Benefit of Swap Contract. Let $\Theta(a_i)$, $\Theta(a_j)$ be the VMs assigned on agents a_i and a_j and $\Theta'(a_i)$, $\Theta'(a_j)$ be their VM loads after executing the VM swap contract sc=<$a_i,\Theta_{i,j},a_j,\Theta_{j,i}$>, where $\Theta'(a_i)=\Theta(a_i)\cup\Theta_{j,i}\setminus\Theta_{i,j}$ and $\Theta'(a_j)=\Theta(a_j)\cup\Theta_{i,j}\setminus\Theta_{j,i}$. The benefit gained by the swap contract $sc(\cdot)$ is defined as the difference of energy costs between Θ and Θ', i.e.,

$$B(a_i,\Theta_{i,j},a_j,\Theta_{j,i})=e_i(u_i)+e_j(u_j)-e_i(u_i')-e_j(u_j') \quad (13)$$

where u_i (resp. u_j) and u_i' (resp. u_j') are the resource utilizations of a_i (resp. a_j) before and after the swap contract sc.

A swap contract sc is ***profitable*** if and only if it yields a positive benefit (i.e., $B(sc)>0$). The system energy cost of the allocation after executing a profitable swap sc is equal to the system energy cost of the allocation before executing swap sc minus the benefit of sc. Similarly, a swap path is profitable if and only if all of the swaps along the path are profitable.

Next, we will present the profitable swap contract between any pair of negotiable agents. For any pair of negotiable agents a_i and a_j, with VM loads $\Theta(a_i)$ and $\Theta(a_j)$, to find the optimal swap with the maximum benefit, one needs to consider $2^{|\Theta(a_i)|+|\Theta(a_j)|}$ VM exchange combinations ($|X|$ indicates the number of elements in set X). To address this computationally costly optimal problem, we propose a polynomial algorithm for a_i to make a profitable swap contract with a_j, shown in Algorithm 2. In Algorithm 2, a_i first sorts its own VMs $\Theta(a_i)$ in decreasing order by their size and ranks a_j's VMs $\Theta(a_j)$ in increasing order by their size (Steps 1~2). Then, in Steps 3~11, a_i attempts to exchange its VMs in order of these VMs' ranking with a_j's VMs in order of a_j's VMs' ranking. The motivation of this idea is that each agent prefers to migrate out its resource-consuming VMs to other agents in exchange for resource-saving VMs to reduce its own energy consumption. For each *out* VM set $\Theta_{i,j}\cup\theta_x$ ($1\le x\le|\Theta(a_i)|$ and initially $\Theta_{i,j}=\emptyset$), a_i identifies the profitable *in* VM set $\Theta_{j,i}$ from a_j as follows: a_i first constructs a set S including all VM combinations on a_j from θ_1 to θ_y ($y\le|\Theta(a_j)|$), i.e., $S=\{\{\theta_1\},\{\theta_1,\theta_2\},\ldots,\{\cup_{1\le z\le|\Theta(a_j)|}\theta_z\}\}$. Agent a_i then selects the most profitable *in* VM set $\cup_{1\le z\le y^*}\theta_z$ from S that yields the maximum benefit $B(a_i,\Theta_{i,j}\cup\theta_x,a_j,\cup_{1\le z\le y^*}\theta_z)$. If this VM swap contract $sc(a_i,\Theta_{i,j}\cup\theta_x,a_j,\cup_{1\le z\le y^*}\theta_z)$ has a greater benefit than the previous swap contract with benefit b, a_i then updates the *out* VMs $\Theta_{i,j}=\Theta_{i,j}\cup\theta_x$ and updates the *in* VMs, $\Theta_{j,i}=\cup_{1\le z\le y^*}\theta_z$ (Steps 7~8).

After presenting the swap contract between any pair of agents, another question arises: given the agent a_i, which negotiable agent $a_j \in D_i$ should a_i negotiate with that can make a profitable swap contract? And if there are multiple profitable swap contracts, which one should a_i choose to execute? To answer these questions, we propose Algorithm 3 for each agent a_i to make the most profitable swap contracts with its negotiable agents D_i. In Algorithm 3, before negotiation, a_i first initializes its state: the sets *out* and *in* store the VMs that are migrated out from a_i and the VMs that are migrated in to a_i, respectively. The variable *target* indicates the target agent that a_i would migrate VMs to (Step 1). In Step 2, a_i only negotiates with the agent $a_j \in D_i$ that has a smaller cost-capacity ratio than that of a_i. The motivation of this idea is that, compared with the agent that has the smaller cost-capacity ratio, the agent that has the greater cost-capacity ratio has more urgency to move its VMs out to reduce its own energy cost. For each negotiable agent $a_j \in D_i$, a_i computes the profitable VM swap <$a_i,\Theta_{i,j},a_j,\Theta_{j,i}$> (i.e., a_i migrates VMs $\Theta_{i,j}$ to a_j and a_j migrates VMs $\Theta_{j,i}$ to a_i) with a_j (Step 4). The profitable swap computation adopted by the agents to identify the profitable VMs exchange is described in Algorithm 2. If the current swap contract sc=<$a_i,\Theta_{i,j},a_j,\Theta_{j,i}$> committed with agent a_j yields a greater benefit than the benefit b of the previous

Algorithm 2. Compute Profitable Swap (a_i,$\Theta_{i,j}$,a_j,$\Theta_{j,i}$)

% a_i is the agent where swap contract is implemented; a_j is the agent that a_i is negotiating with; $\Theta_{i,j}$ is the VMs that a_i migrates to a_j; $\Theta_{j,i}$ is the VMs that a_j migrates to a_i.%

1. Sort the VMs $\Theta(a_i)$ in decreasing order by the amount of their required resources.
2. Sort the VMs $\Theta(a_j)$ in increasing order by the amount of their required resources.
3. Set b=0.
4. **for** $1\le x\le|\Theta(a_i)|$, **do**
5. $\quad b=B(a_i, \Theta_{i,j}\cup\theta_x, a_j, \emptyset)$.
6. $\quad$ **for** $1\le y\le|\Theta(a_j)|$, **do**
7. $\quad\quad$ **if** $B(a_i, \Theta_{i,j}\cup\theta_x, a_j, \cup_{1\le z\le y}\theta_z)>b$, **do**
8. $\quad\quad\quad \Theta_{i,j}=\Theta_{i,j}\cup\theta_x$, $\Theta_{j,i}=\cup_{1\le z\le y}\theta_z$, $b=B(a_i, \Theta_{i,j}, a_j, \Theta_{j,i})$.
9. $\quad\quad$ **end if**
10. $\quad$ **end for**
11. **end for**

contract with other negotiable agent, a_i prefers to make a new swap contract with a_j (Steps 5~7).

Computation and communication complexities of swap contract algorithm (i.e., Algorithm 3). In Algorithm 3, the agent a_i needs to negotiate with all of its local domain agents D_i to find the most profitable swap contract. Given a local domain agent $a_j\in D_i$, a_i utilizes Algorithm 2 to compute the profitable VM swaps between a_i and a_j. In Algorithm 2, a_i first takes $O(2n\log n)$ time to sort the VMs on a_i in decreasing order of their size and the VMs on a_j in increasing order of their size (Step 1~2 of Algorithm 2). Then, a_i takes $O(n^2)$ time to consider the n^2 combinations of VM swap to find the most profitable swaps between a_i and a_j (Step 4~11 of Algorithm 2). Therefore, Algorithm 3 takes total $O(k^\rho(2n\log n+n^2))=O(k^\rho n^2)$ computations to return the final VM swaps with its certain domain agent, where k is the average degree of each PM and ρ is agent a_i's negotiation radius, i.e., $|D_i|=k^\rho\le m$ (m is the number of PMs). Moreover, agent a_i needs to communicate with D_i for one time to acquire their VM load information, generating $O(k^\rho)$ communication cost.

Next, we will illustrate how the swap contract-based negotiation mechanism addresses the inefficient allocation problems emerged in Example 1 and Example 1 continued. In Example 1, the optimal allocation {{3},{1,2},{4}} can be achieved by two sequential profitable swaps, i.e., (a_1, {θ_2,θ_4},a_2,{θ_3}) and (a_2,{θ_4},a_3,{θ_1}). In Example 1 continued, we can achieve the optimal allocation {{4},{3},{2}} by executing two sequential profitable swaps, (a_1,{θ_3},a_2,{θ_2}) and (a_1,{θ_2},a_3,{θ_4}), here we assume the negotiation radius ρ=2. Although this swap contract mechanism is necessary to reduce energy cost, it is not sufficient to lead the system optimal with the minimum energy cost even though global communication is permitted.

Proposition 4. Given a VM allocation problem, even though global communication is permitted, there does not always exist a profitable swap path that can lead certain allocation $\{\Theta(a_i)\}_{1\le i\le m}$ to the optimal allocation $\{\Theta^*(a_i)\}_{1\le i\le m}$.

Proof. We achieve this conclusion by a concrete example. Fig. 5 shows a VM allocation example with four VMs {θ_5=7, θ_6=7, θ_7=8, θ_8=5} running in a cloud system (this cloud system is just the one described in Example 1). Assume that the initial VM allocation is {{5,6},{7},{8}}, i.e., θ_5 and θ_6 are placed on p_1, θ_7 is placed on p_2, and θ_8 is placed on p_3 (Fig. 5(a)). Fig. 5(b) depicts the profitable swap graph of this problem instance, where global communication is permitted. From Fig. 5(b), we observe that for the allocation {{5,6},{7},{8}}, no profitable swap path can lead {{5,6},{7},{8}} to the optimal allocation {{5},{6,8},{7}}. However, this optimal allocation can be achieved by multiple VM migrations among a cluster of agents a_1, a_2 and a_3, i.e., a_1 migrates θ_6 to a_2, a_2 migrates θ_7 to a_3, and a_3 migrates θ_8 to a_1 (Fig. 5(c)). □

Algorithm 3. Swap Contract *(SC)* (a_i)

% a_i: the agent where SC is implemented%

1. Set *out*=∅, *in*=∅, *target*=∅, and b=0.
2. **for** $\forall a_j\in D_i$ && $\lambda_j/c_j<\lambda_i/c_i$ **do**
3. $\quad$ Set $\Theta_{i,j}=\emptyset$ and $\Theta_{j,i}=\emptyset$.
4. $\quad$ Compute profitable swap $<a_i,\Theta_{i,j},a_j,\Theta_{j,i}>$ with a_j.
5. $\quad$ **if** $B(a_i,\Theta_{i,j},a_j,\Theta_{j,i})>b$, **do**
6. $\quad\quad out=\Theta_{i,j}$, $in=\Theta_{j,i}$, $target=a_j$, $b=B(a_i,out,a_j,in)$.
7. $\quad$ **end if**
8. **end for**

4.2.2.2 Cluster Contract

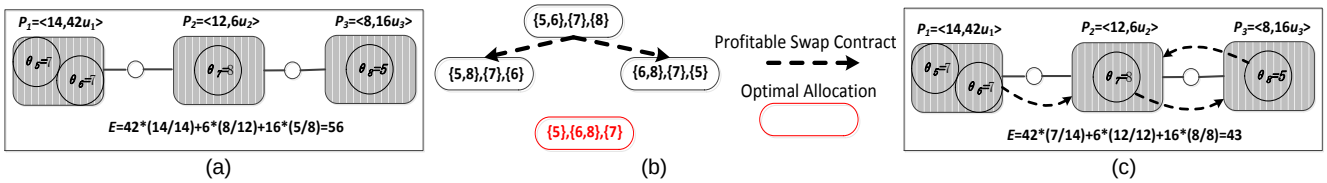
As discussed above, there are VM allocation problems where no profitable path of swap contract can lead certain allocation to the optimal allocation. In this section, we propose a complementary local cluster contract mechanism to address the inefficiencies of the swap contract mechanism in reducing energy cost, where VMs can be transferred among a cluster of agents.

Definition 8. *k*-Cluster Contract. Given an agent $a_i\in A$, a k-cluster contract cc_{ai}^k ($k\ge2$) is a combination of k *out* and *in* contracts between a_i and its negotiable agents D_i, i.e.,

$$cc_{a_i}^k\subseteq\bigcup_{a_j\in D_i,\theta_x\in\Theta(a_i)}out(a_i,\theta_x,a_j)\bigcup_{a_j\in D_i,\theta_x\in\Theta(a_j)}in(a_i,\theta_x,a_j) \quad (14)$$

Denoted by $A(cc_{ai}^k)$ the set of agents involved in the k-cluster contract cc_{ai}^k. Similar to the swap contract benefit definition, the benefit of a k-cluster contract cc_{ai}^k, $B(cc_{ai}^k)$ is also defined as the difference of energy costs of all contracted agents $A(cc_{ai}^k)$ before and after executing the k-cluster contract cc_{ai}^k, i.e.,

$$B(cc_{a_i}^k)=\sum_{a_j\in A(cc_{a_i}^k)}(e_j(u_j)-e_j(u_j')) \quad (15)$$



(a) (b) (c)

Fig. 5. A VM allocation example shows the reason for the cluster contract. (a) Initial allocation {{5,6},{7},{8}}. (b) Profitable swap graph, where negotiation radius ρ=2. For allocation {{5,6},{7},{8}}, there does not exist a profitable swap path leading this allocation to the optimal allocation {{5},{6,8},{7}}. (c) The optimal allocation can be achieved by multiple VM exchanges among a cluster of agents {a_i|$1\le i\le3$}.

Algorithm 4. Cluster Contract *(CC)* (a_i,K)

/a_i: the agent where CC is implemented; K is the predetermined value constraining the scale of the cluster contract*/*

1. Initialize M=∅.
2. Build a_i's *out* contracts to its negotiable agents D_i

$$Out(a_i, D_i) = \bigcup_{\theta_x \in \Theta(a_i), a_j \in D_i} out(a_i, \theta_x, a_j),$$

and a_i's *in* contracts from its negotiable agents D_i

$$In(a_i, D_i) = \bigcup_{a_j \in D_i, \theta_x \in \Theta(a_j)} in(a_i, \theta_x, a_j).$$

3. **for** $2 \le k \le K$
4. Identify and add all k-cluster contracts cc_{ai}^k from $Out(a_i,D_i) \cup In(a_i,D_i)$ to M.
5. **end for**
6. Select the most profitable k^*-cluster contracts $cc_{ai}^{k^*}$ from M, i.e.,

$$cc_{a_i}^{k^*} = \arg \max_{cc_{a_i}^k \subseteq Out(a_i,D_i) \cup In(a_i,D_i)} B(cc_{a_i}^k).$$

where $e_j(u_j)$ and $e_j(u_j')$ represent the energy costs of the contracted agents $a_j \in A(cc_{ai}^k)$ before and after the k-cluster contract cc_{ai}^k, respectively.

Next, we will present the cluster contract formally, shown in Algorithm 4. In Algorithm 4, before computing the cluster contract, the agent a_i first predetermines the scale K of cluster contract, indicating that there are at least K VMs involved in the cluster contract. Then in Step 1, a_i initializes k-cluster contract ($k \le K$) set M. In Step 2, a_i constructs all of its *out* contracts to its negotiable agents D_i, $Out(a_i, D_i) = \bigcup_{a_j \in D_i, \theta_x \in \Theta(a_j)} out(a_i, \theta_x, a_j)$ and all of its *in* contracts from D_i, $In(a_i, D_i) = \bigcup_{a_j \in D_i, \theta_x \in \Theta(a_j)} in(a_i, \theta_x, a_j)$. After constructing all single *out* and *in* contracts $Out(a_i,D_i) \cup In(a_i,D_i)$, a_i identifies and adds all k-cluster ($k \le K$) contracts (which can be implemented by the brute-force search approach) to M (Steps 3~5). Finally, a_i selects the optimal k^* cluster contract $cc_{ai}^{k^*}$ from M that has the maximal benefit.

Computation and communication complexities of cluster contract algorithm (i.e., Algorithm 4). In algorithm 4, each agent needs to find all k-cluster ($k \le K$) contracts among the union of all single *out* and *in* contracts $Out(a_i,D_i) \cup In(a_i,D_i)$. The number of feasible single *out* and *in* contracts $|Out(a_i,D_i) \cup In(a_i,D_i)| \le n$, because each VM can only be involved in only one *out* or *in* contracts, where n is the number of VMs. Therefore, the computation of Algorithm 4 is $\sum_{2 \le k \le K} \binom{n}{k} = O(n^K)$, where K is the maximum number of transferred VMs. On the other hand, in the cluster contract, each agent a_i only needs to communicate with its negotiable agents D_i to acquire their VM load information. This type of information has been collected in the swap contract process. Therefore, the cluster contract does not incur extra communication cost.

4.2.3 Algorithm of Negotiation-Based Consolidation

In this section, we formally present the *n*egotiation-based VM *c*onsolidation(*NC*) algorithm by integrating the swap and cluster contracts, shown as follows.

1) Each agent $a_i \in A$ invokes Algorithm 3 and Algorithm 4 to compute the swap contract $SC(a_i)$ and cluster contract $CC(a_i,K)$, respectively. If the swap contract has a greater benefit than the cluster contract, i.e., $B(SC(a_i)) \ge B(CC(a_i))$, then a_i sends the swap contract request <a_i, $\Theta_{i,j}$, a_j, $\Theta_{j,i}$, $B(SC(a_i))$, *SC*> to the target contracted agent a_j, where *SC* indicates the contract type (i.e., swap contract). Otherwise, i.e., $B(SC(a_i)) < B(CC(a_i))$, a_i sends the cluster contract request <a_i,$cc_{ai}^{k^*}$,$B(cc_{ai}^{k^*})$, *CC*> to all involved agents $A(cc_{ai}^{k^*})$ in cluster contract $cc_{ai}^{k^*}$, and *CC* indicates the cluster contract type.
2) Each agent a_i stores all contract requests and sends an acknowledge message <*Ack*> to the winner agent whose contract request has the maximum benefit.
3) The agent a_i that receives acknowledgments from all of the contracted agents, executes the contract by exchanging VMs with its contracted agents.

The *NC* repeats the above Steps 1~3 until there is no agent that can benefit by negotiating with other agents for VM migration**.** In *NC*, each VM migration operation is profitable to reduce energy cost, therefore *NC* has monotonicity. Furthermore, let $E(ini)$ and $E(opt)$ be the energy costs of the initial and optimal VM allocation, respectively, then *NC* can reach the optimal allocation in at most $(E(ini)\text{-}E(opt))/\tilde{B}$ steps ($\tilde{B}$ is the average benefit of all of the VM migration operations), indicating that *NC* is also convergent.

5 Simulation Validation and Analyses

We validate the advantages of the proposed multiagent (MA)-based resource allocation approach in two series of experimental settings: 1) a static setting, where we are only concerned with allocating a set of VMs to PMs (Section 5.1) and 2) a dynamic setting, where VMs arrive and depart the cloud systems dynamically (Section 5.2).

5.1 Validate the Advantage of the MA approach in a Static Setting

A. Experiment Setup

In the static experiment setting, each cloud system consists of 128 PMs. Three typical network architectures are used to simulate the underlying topology of these PMs, shown as follows:

- **Tree Network** [59]. The 128 PMs first are randomly classified into 16 clusters, each with 8 PMs. PMs that belong to the same cluster of 8 (i.e., 1~8, 9~16, etc.) can communicate with each other by a single switch, i.e., with one-hop communication distance. PMs that belong to the same cluster of 16 (1~16, 17~32, etc.) but that are not in the same cluster of 8, can communicate by 3 switches. Analogously, PMs that belong to the same cluster of 32, but not the cluster of 16 have 5-hop communication distances. The PMs that belong to the same cluster of 64 (but that are not in the cluster of 32) must communicate through 7 switches. Finally, PMs that belong to the cluster of 128 (but that are not in the cluster of 64) have 9-hop communication distances.
- **BCube Network** [60][61]. BCube is a 7-level network structure that can be constructed recursively. At level 0, $BCube_0$ consists of 2 PMs that can communicate via only one switch. Recursively, a $BCube_h$ ($1 \le h \le 6$) level is constructed from 2 $BCube_{h-1}$ levels interconnected by 2^h 2-port switches connecting each PM in the former

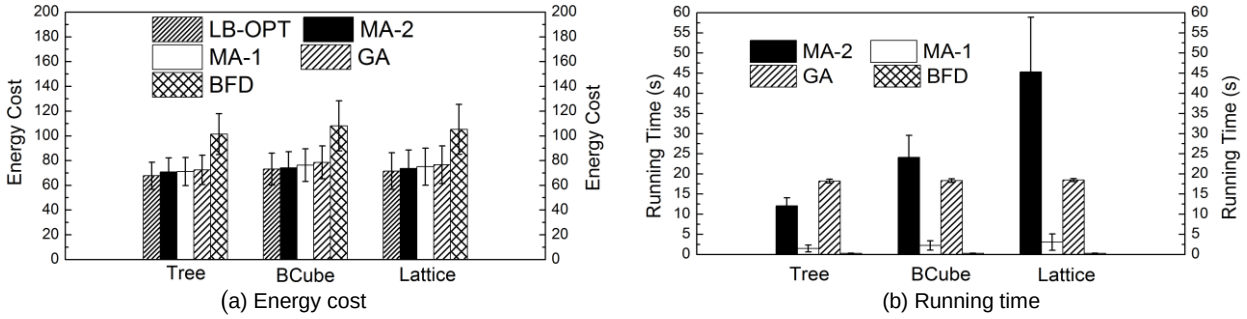


Fig. 6. The experiment results on (a) energy cost and (b) running time of the approaches in different static cloud systems.

$BCube_{h-1}$ to another PM in the latter $BCube_{h-1}$. Each PM in a BCube can be denoted by its address array $a_6a_5\ldots a_0$ ($a_i \in \{0,1\}, 0 \le i \le 6$). For example, if a PM connects the left port of the switch in the $BCube_0$ level and connects the right port of the switches in the $BCube_h$ ($1 \le h \le 6$) level, then its address array is 0000001. To compute the communication distance among PMs, we first assign each of the 128 PMs a 7-bit binary address, for example, PM p_0 and p_{127} can be addressed as 0000000 and 1111111, respectively. Then, the communication cost between any two PMs p_i and p_j can be defined as:

$$d(p_i,p_j)=hamdist(addrees(i),address(j)) \quad (16)$$

where the function $hamdist(addrees(i),address(j))$ is the hamming distance between the two strings $address(i)$ and $address(j)$, which is computed as the number of positions at which the corresponding symbols are different between $address(i)$ and $address(j)$, e.g., $hamdist(1\underline{1}10\underline{0}0\underline{1},1\underline{1}00\underline{1}10)=4$.

- **Lattice-Like Network** [48]. Each PM p_i connects with its local l PMs $\{(m-l/2)\%m,\ldots,(m+i-1)\%m,(m+i+1)\%m,\ldots,(m+l/2)\%m\}$, where m is the number of PMs, l is the degree of each PM (here we set l=8), and $a\%b$ returns the remainder of a/b.

After constructing the network of the cloud systems, we next set the configurations of the PMs. For each PM p_i, its capacity c_i distributes in the range [10, 30] randomly, its maximum energy consuming λ_i is selected in (0,10] randomly, and the idle energy consuming ratio α_i is selected from [0,1] uniformly. In this static setting, each experiment has 200 VMs to be satisfied, and each VM's resource requirement distributes in the interval [1,10] randomly.

B. Approaches

In the static setting, we compare our MA-based distributed approach[2] with three typical centralized static resource allocation approaches:

- **Multiagent-Based Approach (MA)**. This approach is proposed by us. We denote MA-1 and MA-2 as the MA approaches with negotiation radius ρ=1 and ρ=2, respectively.
- **Lower Bound of the Optimal Solution LB-OPT [21].** The lower bound of the integer programming resource allocation problem defined in Definition 1 is the optimal value of the relaxed liner programing problem where the VM can be allocated on PM fractionally. In this relaxed formulation, we first sort all the PMs in increasing order of their cost-capacity ratio, i.e., $c_1/\lambda_1 \le c_2/\lambda_2 \le \ldots \le c_m/\lambda_m$ and denote k as the minimal number of PMs that can be used to host all system VMs, i.e., $k=\min\{j \in \{1,\ldots,m\}: \Sigma_{1\le i\le j} c_i \ge R\}$, where R represents the sum of resources required by all VMs, i.e., $R=\Sigma_{\theta_j \in \Theta} r_j$, then the lower bound of the integer programming resource allocation problem is:

$$OPT_{LB} \le \sum_{i=1}^{k-1} \lambda_i + \frac{(R-\sum_{i=1}^{k-1} c_i)\lambda_k}{c_k} \quad (17)$$

- **Bin Packing-Based Best Fit Decreasing Approach (BFD)** [13]. For a set of VM requests submitted to the system, the system manager first sorts the VMs in decreasing order by their size and then places these sorted VM on the optimal PM that increases the least energy cost. After this greedy VM allocation, the manager then checks each PM p_i: if there exists any simple profitable migration by migrating the VM $\theta_k \in \Theta(p_i)$ from p_i to another PM p_j reduces system energy cost, the manger will transfer θ_k from p_i to p_j.
- **Static Genetic Approach (GA)** [23]. In this static setting, we set the number of the potential solution and the number of iterations both equal to 1000. The candidate solution's fitness value is defined as the energy cost, i.e., $fitness(S)=\Sigma_i e(u_i)$, S is an allocation solution.

C. Performance Metrics.

In the static setting, we compare these approaches on two performance metrics: 1) ***Energy cost*** and 2) ***Running time***. The energy cost is the sum of system PMs' energy cost for running VMs and the running time is measured as the computation time used for allocating VMs to PMs.

D. Experiment Results

Fig. 6 shows the energy cost (Fig. 6(a)) and running time (Fig. 6(b)) of these approaches in different cloud systems. From Fig. 6, we have the following observations.

With respect to energy cost (Fig. 6(a)), we can conclude that: 1) In all systems, MA-2, MA-1 and GA approaches produce as little energy cost as the optimal solution LB-OPT, indicating that these three approaches MA-2, MA-1 and GA perform well in the static setting on reducing energy cost. 2) MA-2 produces less energy cost than MA-1. This is because in MA-2, each PM can negotiate with much more PMs than each PM in MA-1, leading MA-2 to identify much more beneficial VM migrations for energy cost reduction. 3) BFD generates much energy cost than MA-2, MA-1 and GA. The potential reason is that in the static setting, once the VMs are allocated to PMs by BFD, there are few beneficial simple VM migrations (i.e., the migration of one VM from one PM to another PM that can reduce system energy cost) that can be identified by BFD. Therefore, BFD performs worse than MA-2, MA-1 and GA in the static setting.

With respect to running time (Fig. 6(b)), on a computer On a computer with 2.67 GHz CPU and 2 GB memory, for this scale application with 128 PMs and 200 VMs, BFD

[2] In the simulation, we implement MA in a centralized manner: at each running round, we first sort the agents in arbitrary order and then the agents run the centralized algorithm sequentially in this order.

and MA-1 can return the allocation solution within several seconds (<4*s*). However, MA-2 and GA approaches take almost 20 seconds to return the final allocation solution. This is because in MA-2, each PM has to negotiate with much more PMs (about 8 times more than that of MA-1), then each PM needs to search much more potential beneficial migrations (about 8 times more than that of MA-1), thereby consuming much more running time. For GA, because the number of iteration has to set large enough (e.g., 1000 rounds) to return the desirable solution, therefore, GA also needs to take much more running time.

In conclusion, in the static setting, compared with traditional VM allocation approaches (i.e., BFD and GA), our proposed MA approach with negotiation radius ρ=1 can reduce system energy cost significantly within tolerable running time.

5.2 Validate the Advantage of the MA approach in a Dynamic Setting

A. Experiment Setup

To imitate the dynamics of VMs' arrival and departure, we redefine each VM $\theta_i=<r_i,at_i,wt_i>$, where r_i represents θ_i's resource requirement, at_i represents the arrive time-slot and wt_i represents the workloads such that θ_i must use r_i unit resources for wt_i time-slots. In this dynamic setting, each VM θ_i's work load wt_i distributes in the range [1,4] uniformly. At each time-slot t, we assume that there are $X(t)$ VM requests newly submitted to the system, where $X(t)$ follows a Poisson distribution, i.e., $X\sim\pi(\mu)$. Here, we set the mean value μ=100. In this dynamic setting, VM migration is used to consolidate the VM resources to reduce system energy cost. Therefore, besides comparing with the static VM allocation approaches described in 5.1 (i.e., BFD), we also compare MA with other dynamic VM consolidation approaches.

- **Energy- and Migration-Cost-Aware Approach (pMapper)** [11]**.** This is an extension of the BFD approach that considers both energy and migration costs. In pMapper, a VM migration is implemented if and only if this VM migration reduces system energy cost while incurring tolerable migration cost. For example, if PM p_i wishes to migrate its VM $\theta_k\in\Theta(p_i)$ to another PM p_j, this migration should satisfy
$$[e_i(u_i)+e_j(u_j)-e_i(u_i')-e_j(u_j')]\cdot\beta-d(p_i,p_j)>0 \quad (18)$$
where u_i (resp. u_j) and u_i' (resp. u_j') are the resource utilization of p_i (resp. p_j) before and after the migration of θ_j from p_i to p_j. Parameter β determines the importance of energy cost reduction. Here, we set β=10.
- **Probability-Based Approach (PRO)** [15]. This is a distributed approach. Before implementing this approach, the system manager first needs to predetermine the high threshold t_h and lower threshold t_l of the resource utilization. For a set of unallocated VMs, each PM p_i determines whether to host these VMs with probability $f(u_i,t_h)$, which is based on t_h and the resource utilization u_i. After this VM allocation, each PM p_i employs a probabilistic VM migration procedure to avoid resource overutilization and under-utilization. To avoid PM p_i being overloaded, p_i can migrate its exceeded VMs out with probability $f^h_{migrate}(u_i,t_h)$. To avoid p_i being under-loaded, p_i can migrate all of its host VMs out to other PMs with probability $f^l_{migrate}(u_i,t_l)$. Here, we set t_h=0.9 and t_l=0.2.
- **Dynamic Genetic Approach (GA)** [23][27]. In the dynamic setting, the candidate solution's fitness is a tradeoff between the energy cost and migration cost, i.e.,
$$fitness(S)=\gamma\cdot(\sum_i e_i(u_i,I)-\sum_i e_i(u_i,S))-MC(I,S) \quad (19)$$
where I is the current VM allocation and S is the final allocation returned by GA, $e_i(u_i,I)$ (resp. $e_i(u_i,S)$) is the energy cost of the PM p_i in solution I (resp. S) and $MC(I,S)$ is the overall migration cost incurred by migrating VMs from solution I to S, i.e.,
$$MC(I,S)=\sum_{\theta_i} d(p(\theta_i,I),p(\theta_i,S)) \quad (20)$$
where $p(\theta_i, I)$ (resp. $p(\theta_i, S)$) denotes the host PM of VM θ_i in solution I (resp. S).

Table I
The properties of networks

Property \ Network	BCube	Tree	Lattice
Diameter (*L*)	7.00	9.00	16.00
Average Path length (*Apl*)	3.47	7.06	8.31

In this dynamic setting, we are mainly concerned with energy cost and migration cost metrics[3]. Migrating a VM from PM p_i to another PM p_j will incur $d(p_i,p_j)$ migration cost. More specifically, we record two types of energy cost. The first one is the system energy cost $E(t)$ of each time slot t:
$$E(t)=\sum_{p_i\in P} e_i(u_i,t) \quad (21)$$
which is generated by all of the PMs at time slot t. The second type is the system cumulative energy cost $CE(t)$, which is generated by all of system PMs before time slot t:
$$CE(t)=\sum_{0\le\tau\le t}\sum_{p_i\in P} e_i(u_i,t) \quad (22)$$

B. Experiment Results

Table I shows the properties of network diameter (L) and average path length (Apl) of the BCube, Tree and Lattice cloud systems. Network diameter L is defined as the longest communication distance among any two PMs, and average path length Apl is computed as the average communication distance within all pairs of PMs.

Fig. 7 and Fig. 8 show the energy cost of each time slot and the cumulative energy cost of the resource allocation approaches within different cloud systems, respectively. From Fig. 7 and Fig. 8, we have the following conclusions: 1) In BCube and Lattice systems (i.e., Fig. 7(a) and Fig. 7(c)), MA-1 generates nearly as little energy cost as MA-2, GA, BFD and pMapper. However, in the Tree system (i.e., Fig. 7(b)), MA-1 approach generates slight more energy cost than the energy costs of MA-2, GA, BFD and pMapper. This phenomenon can be explained by the special odd communication distance among PMs in the Tree system. In the Tree system, the communication distance among agents is odd value, i.e., 1, 3, 5, 7, and 9. With the local negotiation constraint in MA-1 (in MA-1, the negotiation radius ρ=1, where each PM can only negotiate with its direct connected PMs), one VM migration can only affect the VM allocation of a specific domain. For example, a_1 migrates VM θ_j to its domain agents $\{a_k|2\le k\le 8\}$) and this VM migration of a_i can only affect the surrounding of its local domain agents $\{a_k|1\le k\le 8\}$), which does not affect other domains (e.g., $\{a_k|9\le k\le 16\}$}) because other domains do not belong to the negotiation domain of $\{a_k|1\le k\le 8\}$.

[3] As discussed in Section 5.1, all of the approaches can return the final VM allocation within limited seconds (<30s) and hence, it makes no sense to compare the metric of running time in this dynamic setting where each time slot (e.g., half an hour) is much larger than the running time.

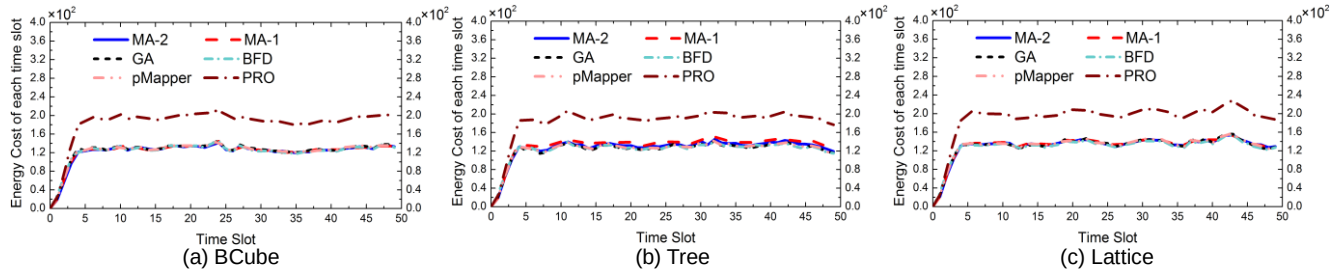


Fig. 7. Energy cost of each time slot of the resource allocation approaches in different cloud systems.

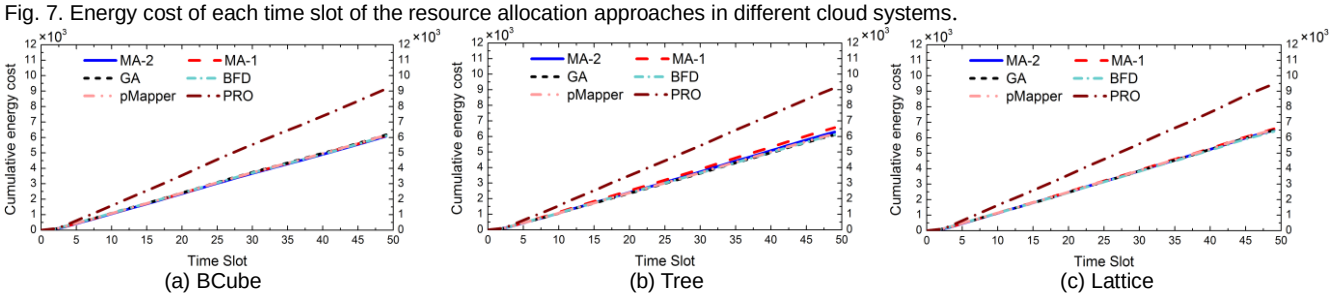


Fig. 8. Cumulative energy cost of the resource allocation approaches in different cloud systems.

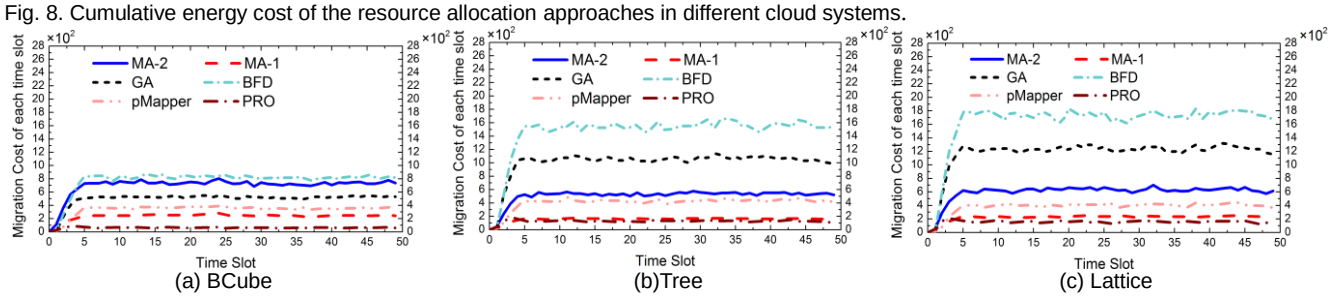


Fig. 9. Migration cost of each time slot of the resource allocation approaches in different cloud systems.

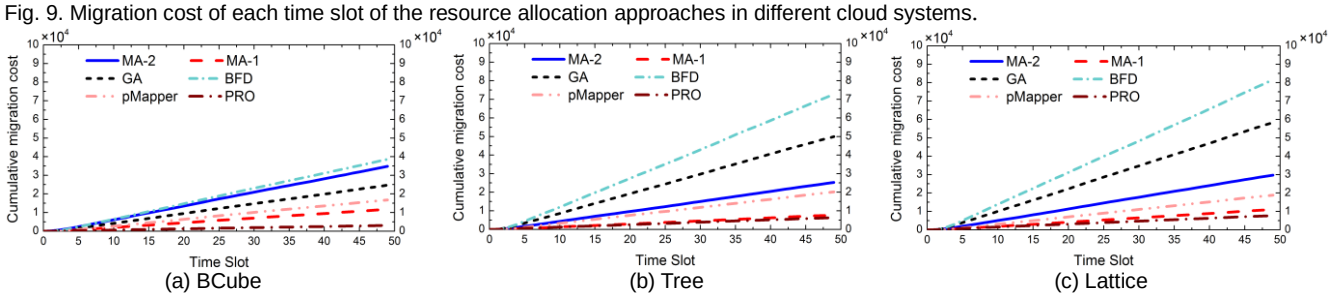


Fig. 10. Cumulative migration cost of the resource allocation approaches in different cloud systems.

Therefore, in the Tree system, MA-1 approach will miss a number of potential beneficial VM migrations. However, in MA-2 with the negotiation radius ρ=2, each VM migration not only affects the VM allocation of the local domain but also other connected domains (e.g., $\{a_k | 9 \le k \le 16\}$), thereby much more beneficial migrations will be identified. 2) In all systems, PRO generates much more energy cost than other approaches MA-1, MA-2, GA, BFD and pMapper, indicating that PRO performs worse on reducing system energy cost in the dynamic applications.

Fig. 9 and Fig. 10 show the migration cost of each time slot and the cumulative migration cost of these approaches in different cloud systems, respectively. From Fig. 9 and Fig. 10, we have the following conclusions: 1) In all systems, MA-1 produces much less migration cost than the migration costs produced by MA-2, GA, BFD and pMapper. This observation can be explained as follows: GA, BFD and pMapper all permit global VM migration and MA-2 also permits the VM migration to happen among remotely connected PMs with two hop communication distance, while MA only transfers VMs among locally connected PMs. Thus, MA-2, GA, BFD and pMapper will produce much more migration overhead than MA-1 does. 2) For BCube and Lattice systems, the shorter the average path length *Apl* (or network diameter), the more migration cost MA-1 incurs. A possible reason is that in the cohesive systems (e.g., the BCube system with *Apl*=3.47), a change of one PM's VM load easily influences the loads of its surrounding PMs. Thus, many chain VM migrations will be trigged, resulting in an increase in migration cost. However, although the Tree system has a shorter *Apl* than that of the Lattice systems, MA-1 approach incurs less migration cost in Tree than it does in Lattice. This phenomenon can be explained by the special odd communication distance among PMs in the Tree system (the detailed explanation can be seen in the above paragraph). 3) In all systems, the larger the network *Apl* (or network diameter), the more migration costs BFD and PRO generate. This is because BFD and PRO mainly focus on minimizing energy cost, does not consider migration cost when transferring VMs. Therefore, the migration costs of BFD and PRO are proportional to *Apl*. 4) In all systems, the migration costs of GA and pMapper do not increase proportionally to network diameter or *Apl*. The

reason is that when *Apl* becomes large, GA and pMapper will not execute some migrations because of their costly migration overhead. 5) In all systems, PRO generates the minimal migration cost compared with the MA-2, MA-1, GA, BFD and pMapper. A possible reason is that there are only a small number of VM migrations executed in PRO.

In summary, in the dynamic setting, on the one hand, the MA approach generates as little energy cost as GA, BFD and pMapper approaches. On the other hand, the MA approach incurs much less migration overhead than these approaches do. Considering the dramatic advantage in reducing energy cost within acceptable migration overhead, MA is a more desirable approach that can balance energy cost reduction and SLA performance guarantee.

6 CONCLUSIONS AND FUTURE WORK

This paper presents a distributed multiagent(MA)-based resource allocation approach to minimize system energy cost. The proposed MA approach consists of two complementary mechanisms: 1) an auction-based VM allocation mechanism, which is devised for agents to decide which PM should host which VM. Through the theoretical analyses, we can determine that the auction-based VM allocation mechanism has a low approximation ratio on energy cost compared with the optimal solution. 2) A negotiation-based VM consolidation mechanism, which is designed for agents to exchange their assigned VMs to save energy costs and address system dynamics. Experimental results show that in the static setting, the MA approach generates the least energy cost within tolerable running time compared with traditional centralized approaches. Moreover, in the dynamic setting, the MA approach can generate as little energy cost as the centralized benchmark approaches, but significantly reduces the migration overhead. These advantages make MA approach a preferable choice for resource management to reduce system energy cost in near real time, while consuming tolerable amounts of network traffic.

In this paper, we only focus on allocating VMs to PMs with the aim of minimizing system energy cost, ignore the objective of maximizing CSP's revenue of delivering scalable VM resources to users. In the future work, it would be very interesting and necessary to integrate the two objectives (often conflicting) together: in the front-end level, that is allocating VM resources to users, the CSP would like to take full advantage of VM resources to satisfy as many users' requests as possible, thereby increasing CSP's revenue. In the back-end level, however, the CSP would like to allocate the VMs to PMs efficiently to generate as little energy cost as possible, thereby decreasing CSP's operations cost of running user's applications.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (No.61170164, No. 61472079 and No. 71201077), the Funds for Distinguished Young Scholars of the Natural Science Foundation of Jiangsu Province (No.BK2012020), and the Program for Distinguished Talents of Six Domains in Jiangsu Province (No.2011-DZ023).

REFERENCES

[1] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg and Ivona Brandic, "Cloud Computing and Emerging IT platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, 25(6):599-616, 2009.

[2] Michael Armbrust, Armando Fox, Ren Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica and Matei Zaharia, "A View of Coud Computing," *Communnication of the ACM*, 53(4):50-58, 2010.

[3] Junwei Cao, Kai Hwang, Keqin Li, and Albert Y. Zomaya, "Optimal Multiserver Configuration for Profit Maximization in Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1087-1096, 2013.

[4] Hong Xu and Baochun Li, "Dynamic Cloud Pricing for Revenue Maximization," *IEEE Transactions on Cloud Computing*, 1(2):158-171, 2013.

[5] Minghong Lin, Adam Wierman, Lachlan L.H. Andrew and Eno Thereskaet, "Dynamic Right-Sizing for Power-Proportional Data Centers," *Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM'11)*, pp.1098-1106, Shanghai, China, April 10-15, 2011.

[6] Hong Xu and Baochun Li, "A Study of Pricing for Cloud Resources," *ACM SIGMETRICS Performance Evaluation Review*, 40(4):3-12, 2013.

[7] Sivaon Chaisiri, Bu-Sung Lee and Dusit Niyato, "Optimization of Resource Provisioning Cost in Cloud Computing," *IEEE Transactions on Services Computing*, 5(2): 164-177, 2012.

[8] Massoud Pedram, "Energy-Efficient Datacenters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(10):1465-1484, 2012.

[9] Luiz Audré Barroso and Urs Hölzle, "The Case for Energy-Proportional Computing," *Computer*, 40(12):33-37, 2007.

[10] Albert Greenberg, James Hamilton, David A. Maltz and Parveen Patel, "The Cost of A Cloud: Research Problems in Data Center Networks," *ACM SIGCOMM Computer Communication Review*, 39(1):68-73, 2009.

[11] Akshat Verma, Puneet Ahuja and Anindya Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems," *Middleware, Lecture Notes in Computer Science*, 5346:243-264, 2008.

[12] Dabilo Ardagna, Barbara Panicucci, Marco Trubian and Li Zhang, "Energy-Aware Autonomic Resource Allocation in Multitier Virtualized Environments," *IEEE Transactions on Services Computing*, 5(1):2-19, 2012.

[13] Anton Beloglazov, Jemal Abawajy and Rajkumar Buyya, "Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing," *Future Generation Computer Systems*, 28(5):755-768, 2012.

[14] Anton Beloglazov and Rajkumar Buyya, "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, 2012.

[15] Carlo Mastroianni, Michela Meo and Giuseppe Papuzzo, "Probabilistic Consolidation of Virtual Machines in Self-Organizing Cloud Data Centers," *IEEE Transactions on Cloud Computing*, 1(2):215-228, 2013.

[16] Siva Theja Maguluri, R. Srikant and Lei Ying, "Stochastic Models of Load Balancing and Scheduling in Cloud Computing Clusters," *Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOM'12)*, pp.702-710, Orlando, Florida, USA, March 25-30, 2012.

[17] Xin Li, Zhuzhong Qian, Sanlu Lu and Jie Wu, "Energy Efficient Virtual Machine Placement Algorithm with Balanced and Improved Resource Utilization in A Data Center," *Mathematical and Computer Modelling*, 58(5–6): 1222-1235, 2013.

[18] Xin Li, Jie Wu, Shaojie Tang, and Sanlu Lu, "Let's Stay Together: Towards Traffic Aware Virtual Machine Placement in Data Centers," *Proceedings of the 33th IEEE International Conference on Computer Communications (INFOCOM'14)*, pp.1842-1850, Toronto, Canada, April 27- May 2, 2014.

[19] Weijia Song, Zhen Xiao, Qi Chen and Haipeng Luo, "Adaptive Resource Provisioning for the Cloud Using Online Bin Packing," *IEEE Transactions on Computers*, 2014, in press.

[20] Timothy Wood, Prashant Shenoy, Arun Venkataramani and Mazin Yousif, "Black-Box and Gray-box Strategies for Virtual Machine Migration," *Proceedings of the 4th USENIX conference on Networked systems design & implementation (NSDI'07)*, pp.17-30, Cambridge, MA, USA, April 11-13, 2007.

[21] Hadrien Cambazard, Deepak Mehta, Barry O'Sullivan and Helmut Simonis, "Bin Packing with Linear Usage Costs–An Application to Energy Management in Data Centres," *Proceedings of the 19th Conference on Principles and Practice of Constraint Programming (CP-2013)*, 8124:47-62, 2013.

[22] Negin Kord and Hassan Haghighi, "An Energy-Efficient Approach for Virtual Machine Placement in Cloud Based Data Centers," *Proceddings of 5th Conference on Information and Knowledge Technology (IKT'13*), pp.44-49, Mollasadra st., Shiraz, Iran, May 28-30, 2013.

[23] Pardeep Kumar and Amandeep Verma, "Scheduling Using Improved Genetic Algorithm in Cloud Computing for Independent Tasks," *Proceedings of the International Conference on Advances in Computing, Communications and Informatics(ICACCI'12)*, pp.137-142, Chennai, India, August 3-5, 2012.

[24] Xiaoli Wang, Yuping Wang and Hai Zhu "Energy-Efficient Multi-Job Scheduling Model for Cloud Computing and Its Genetic algorithm," *Mathematical Problems in Engineering*, Article ID 589243, 1–16, 2012.

[25] Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru and Rajkumar Buyya, "A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments," *Proceedings of the 24th IEEE International Conference on Advanced Information Networks and Application* (*ANIA-10*), pp.400–407, Perth, Australia, April 20-23, 2010.

[26] Hai-Hao Li, Y. W. Fu, Zhi-Hui Zhan, and J. J. Li, "Renumber Strategy Enhanced Particle Swarm Optimization for Cloud Computing Resource Scheduling," *Proceedings of the IEEE Congress Evolution Compututation*(*CEC'15*), Sendai, Japan, May 25-28, 2015, in press.

[27] Fahimeh Farahnakian, Adnan Ashraf, Tapio Pahikkala, Pasi Liljeberg, Juha Plosila, Ivan Porres, and Hannu Tenhunen, "Using Ant Colony System to Consolidate VMs for Green Cloud Computing," *IEEE Transactions on Services Computing*, 8(2):187-198, 2015.

[28] Patrick Wendell, Joe Wenjie Jiang, Michael J. Freedman and Jennifer Rexford, "Donar: Decentralized Server Selection for Cloud Services," *Proceedings of the ACM SIGCOMM 2010 conference (SIGCOMM'10)*, pp.231-242, New Delhi, India, August 30–September 3, 2010.

[29] Duong-Ba Thuan, Thinh Nguyen, Bella Bose and Duc A. Tran, "Distributed Client-Server Assignment for Online Social Networks Applications," *IEEE Transactions on Emerging Topics in Computing*, 2(4):422-435, 2014.

[30] William Voorsluys, James Broberg, Srikumar Venugopal and Rajkumar Buyya, "Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation," *Proceedings of the 1st International Conference on Cloud Computing (CloudCom'09),* pp.254-265, Beijing, China, December 1-4, 2009.

[31] Mansoor Alicherry and T. V. Lakshman, "Network Aware Resource Allocation in Distributed Clouds," *Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOM'12)*, pp.963-971, Orlando, Florida, USA, March 25-30, 2012.

[32] Mahyar Movahed Nejad, Lena Mashayekhy and Daniel Grosu, "A PTAS Mechanism for Provisioning and Allocation of Heterogeneous Cloud Resources," *IEEE Transactions on Parallel and Distributed Systems*, 26(9):2386-2399, 2014.

[33] Sharrukh Zaman and Daniel Grosu, "Combinatorial Auction-based Allocation of Virtual Machine Instances in Clouds," *Journal of Parallel and Distributed Computing*, 73(4): 495-508, 2013.

[34] Mahyar Movahed Nejad, Lena Mashayekhy and Daniel Grosu, "Truthful Greedy Mechanisms for Dynamic Virtual Machine Provisioning and Allocation in Clouds," *IEEE Transactions on Parallel and Distributed Systems*, 26(2):594-603, 2015.

[35] Lena Mashayekhy, Mahyar Movahed Nejad, Daniel Grosu, Athanasios V. Vasilakos, "Incentive-Compatible Online Mechanisms for Resource Provisioning and Allocation in Clouds," *Proceedings of IEEE International Conference on Cloud Engineering (CloudCom'14)*, pp.312-319, Boston, Massachusetts, USA, March 10-14, 2014.

[36] Linquan Zhang, Zhongpeng Li and Chuan Wu, "Dynamic Resource Provisioning in Cloud Computing: A Randomized Auction Approach,"*Proceedings of the 33th IEEE International Conference on Computer Communications (INFOCOM'14)*, pp.433-441, Toronto, Canada, April 27-May 2, 2014.

[37] Hong Zhang, Bo Li, Hongbo Jiang, Fangming Liu, Athanasios V. Vasilakos and Jiangchuan Liu, "A Framework for Truthful Online Auctions in Cloud Computing with Heterogeneous User Demands," *Proceedings of the 32nd IEEE International Conference on Computer Communications (INFOCOM'13)*, pp.1510-1518, Turin, Italy, April 14-19, 2013.

[38] Tram Truong Huu and Chen-Khong Tham, "An Auction-Based Resource Allocation Model for Green Cloud Computing," *Proceedings of IEEE International Conference on Cloud Engineering (CloudCom'13)*, pp.269-278, San Francisco, California, USA, March 25-28, 2013.

[39] Zhen Xiao, Weijia Song and Qi Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment," *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1107-1117, 2013.

[40] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen,Eric Jul, Christian Limpach, Ian Pratt and Andrew Warfield, "Live migration of virtual machines," *Proceedings of the 2nd USENIX conference on Networked systems design & implementation (NSDI'05)*, pp. 273-286, Boston, Massachusetts, USA, May 2-4, 2005.

[41] William Voorsluys, James Broberg, Srikumar Venugopal and Rajkumar Buyya, "Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation," *Proceedings of the 1st International Conference on Cloud Computing (CloudCom'09),* pp. 254-265, Beijing, China, December 1-4, 2009.

[42] Mansoor Alicherry and T. V. Lakshman, "Network Aware Resource Allocation in Distributed Clouds," *Proceedings of the 31th IEEE International Conference on Computer Communications (INFOCOM'12)*, pp.963-971, Orlando, Florida, USA, March 25-30, 2012.

[43] Thomas L. Saaty, "Decision Making with the Analytic Hierarchy Process," *International Journal of Services Sciences*, 1(1):83-98, 2008.

[44] Zhihui Zhan, Xiaofang Liu, Yuejiao Gong and JunZhang, "Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches," *ACM Computing Survey*, 47(4): 1-33, 2015.

[45] Yichuan Jiang, "A Survey of Task Allocation and Load Balancing in Distributed Systems," *IEEE Transactions on Parallel and Distributed Systems*, DOI: 10.1109/TPDS.2015.2407900, in press, 2015.

[46] Sarit Kraus, Onn Shehory and Gilad Taase, "Coalition Formation with Uncertain Heterogeneous Information," *Proceedings of the 2nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, pp.1-8 , Melbourne, Australia, July 14-18, 2003.

[47] Xiaoming Zheng and Seven Koenig, "K-Swaps: Cooperative Negotiation for Solving Task-Allocation Problems," *Proceedings of 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pp.373-378, Pasadena, California, USA, July 11-17, 2009.

[48] Jiming Liu, Xiaolong Jin and Yuanshi Wang, "Agent-based Load Balancing on Homogeneous Minigrids: Macroscopic Modeling and Characterization," *IEEE Transactions on Parallel and Distributed Systems*, 16(7): 586-598, 2005.

[49] Dayong Ye, Minjie Zhang and Danny Sutanto, "Self-Adaptation-Based Dynamic Coalition Formation in a Distributed Agent Network: A Mechanism and a Brief Survey." *IEEE Transactions on Parallel and Distributed Systems*, 24(5):1042-1051, 2013.

[50] Yichuan Jiang, Yifeng Zhou and Wanyuan Wang, "Task Allocation for Undependable Multiagent Systems in Social Networks," *IEEE Transactions on Parallel and Distributed Systems*, 24(8):1671-1681, 2013.

[51] Federico Bergenti, Enrico Franchi and Agostino Poggi, "Agent-based Interpretations of Classic Network Models," *Computational and Mathematical Organization Theory*, 19(2):105-127, 2013.

[52] Bo An, Victor Lesser, David Irwin and Michazel Zink, "Automated Negotiation with Decommitment for Dynamic Resource Allocation in Cloud Computing," *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10),* pp.981-988, Toronto, Canada, May 10-14, 2010.

[53] Han Zhao, Xinxin Liu and Xiaolin Li, "Towards Efficient and Fair Resource Trading in Community-Based Cloud Computing," *Journal of Parallel and Distributed Computing*, 74(11):3087-3097, 2014.

[54] Kwang Mong Sim, "Agent-Based Cloud Computing," *IEEE Transactions on Services Computing*, **5**(4):564-577, 2012.

[55] Chao Chen, Xiaomin Zhu, Weidong Bao, Lidong Chen and Kwang Mong Sim, "An Agent-Based Emergent Task Allocation Algorithm in Clouds," *IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded*

and Ubiquitous Computing, pp.1490-1497, Zhangjiajie, China, Nov 13-15, 2013.

[56] Mauro Naria Baldi, Teodor Gabriel Crainic, Guido Perboli and Roberto Tadei, "The Generalized Bin Packing Problem," *Transportation Research Part E: Logistics and Transportation Review*, 48(6): 1205-1220, 2012.

[57] Chengbin Chu and Rémy La, "Variable-Sized Bin Packing: Tight Absolute Worst-Case Performance Ratios for Four Approximation Algorithms," *SIAM Journal on Computing*, 30(6): 2069-2083, 2001.

[58] Tuomas W. Sandholm, "Contract Types for Satisficing Task Allocation: I Theoretical Results," *In Proceedings of AAAI Spring Symposium Series: Satisficing Models*, pages 68–75, Madison, Wisconsin, USA, July 26–30, 1998.

[59] Mansoor Alicherry and T.V. Lakshman, "Optimizing Data Access Latencies in Cloud Systems by Intelligent Virtual Machine Placement," *Proceedings of the 32th IEEE International Conference on Computer Communications (INFOCOM'13)*, pp.647-655, Turin, Italy, April 14-19, 2013.

[60] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang and Songwu Lu, "BCube: A High Performance, Server-Centric Network Architecture for Modular Data Centers," *Proceedings of the ACM SIGCOMM 2009 conference (SIGCOMM'09)*, pp.63-74, Barcelona, Spain, August 17-21,2009.

[61] Xiaoqiao Meng, Vasileios Pappas and Li Zhang, "Improving the Scalability of Data Center Networks with Traffic-Aware Virtual Machine Placement," *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM'10)*, pp.1154-1162, San Diego, USA, March 15-19,2010.

Wanyuan Wang received his BS degree in information and computing science from Nanjing University of Aeronautics and Astronautics, Nanjing, China, 2011. He is currently pursuing a Ph.D. degree at the Distributed Intelligence and Social Computing Laboratory, School of Computer Science and Engineering, Southeast University. He has published several articles in refereed journals and conference proceedings, such as the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Cybernetics, the IEEE Transactions on Emerging Topics in Computing, and the IEEE International Conference on Tools with Artificial Intelligence (ICTAI). He won the best student paper award from ICTAI'14. His main research interests include multiagent systems and distributed systems.

Yichuan Jiang received his PhD degree in computer science from Fudan University, Shanghai, China, in 2005. He is currently a full professor and the director of the Distributed Intelligence and Social Computing Laboratory, School of Computer Science and Engineering, Southeast University, Nanjing, China. His main research interests include multiagent systems, social networks, and complex distributed systems. He has published more than 80 scientific articles in refereed journals and conference proceedings, such as the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Cybernetics, the IEEE Transactions on Systems, Man, and Cybernetics: Systems, the IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, the IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, the IEEE Transactions on Emerging Topics in Computing, the Journal of Parallel and Distributed Computing, the ACM Transactions on Autonomous and Adaptive Systems, the International Joint Conference on Artificial Intelligence (IJCAI), the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), and the IEEE International Conference on Tools with Artificial Intelligence (ICTAI). He won the best paper award and best student paper award from PRIMA and ICTAI, respectively. He is a senior member of IEEE, a member of the editorial board of Advances in Internet of Things, an editor of the International Journal of Networked Computing and Advanced Information Management, an editor of Operations Research and Fuzziology, and a member of the editorial board of the Chinese Journal of Computers.

Weiwei Wu received his PhD degree in computer science from both City University of Hong Kong and University of Science and Technology of China in 2011. He was a postdoctoral researcher at Nanyang Technological University in Singapore from 2011 to 2012. He is currently an associate professor at Southeast University, China. His main research interests include optimizations for networks, game theory, and mechanism design. He has published several articles in refereed journals and conference proceedings such as the IEEE Journal on Selected Areas in Communications, Theoretical Computer Science, the Journal of Combinatorial Optimization, the IEEE International Conference on Computer Communications (INFOCOM), and the International Symposium on Algorithms and Computation (ISAAC). He is a member of IEEE.