Cloud Federations in the Sky: Formation Game and Mechanism

Lena Mashayekhy, *Student Member, IEEE,* Mahyar Movahed Nejad, *Student Member, IEEE,* and Daniel Grosu, *Senior Member, IEEE*

Abstract—The amount of computing resources required by current and future data-intensive applications is expected to increase dramatically, creating high demands for cloud resources. The cloud providers' available resources may not be sufficient enough to cope with such demands. Therefore, the cloud providers need to reshape their business structures and seek to improve their dynamic resource scaling capabilities. Federated clouds offer a practical platform for addressing this service management issue. We introduce a cloud federation formation game that considers the cooperation of the cloud providers in offering cloud IaaS services. Based on the proposed federation formation game, we design a cloud federation formation mechanism that enables the cloud providers to dynamically form a cloud federation maximizing their profit. In addition, the proposed mechanism produces a stable cloud federation structure, that is, the participating cloud providers in the federation do not have incentives to break away from the federation. We analyze the performance of the proposed mechanism by performing extensive experiments. The results of the experiments show that the cloud federation obtained by our proposed mechanism is stable, yielding high profit for the participating cloud providers.

Index Terms—Cloud federation, virtual machine, game theory.

1 Introduction

Clouds are large-scale distributed computing systems built around core concepts such as computing as utility, virtualization of resources, on demand access to computing resources, and outsourcing computing services [1]. These concepts have positioned the clouds as an attractive platform for businesses enabling them to outsource some of their IT operations. In fact, the clouds services market share in the IT business has rapidly increased, and it is estimated to reach $150 billion by 2015 [2]. Cloud services are offered as three main categories: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). In this paper, we focus on IaaS, where cloud providers offer different types of resources in the form of virtual machine (VM) instances.

Cloud computing systems' ability to provide on demand access to always-on computing utilities has attracted many enterprises due to their cost-benefit ratios, leading to rapid growth of the cloud computing market. Such market, however, presents a host of new challenges due to the dynamic nature of users' demands. The variability of users' demands increases when it comes to their requests for data-intensive applications. The amount of computing resources that data-intensive applications require can dramatically increase, and cloud providers' available resources may not be sufficient enough to cope with such demands. This emerging service management problem in cloud computing necessitates that cloud providers reshape their business structures and seek to improve their dynamic resource scaling capabilities. Federated clouds offer a practical platform for addressing this service management problem. A cloud provider can dynamically scale-up its resource capabilities by forming a cloud federation with other cloud providers. On the other hand, other cloud providers that have unused capacities can make profit by participating in a federation. Users' requests can be satisfied by federating resources belonging to several cloud providers [3], [4]. A cloud federation is a collection of cloud providers that cooperate in order to provide the resources requested by users. Forming cloud federations helps achieve greater scalability and performance. If a cloud provider does not have enough resources to provide all the requested resources to the customer, it will reject the requests which leads not only to profit loses, but also to reputation losses. However, by forming a federation with other cloud providers, it can provide part of the requested resources to make some profit. In addition, the federation may provide the resources at a lower cost. Employing only one cloud provider may lead to issues such as lock-in and a single point of failure. Interoperability among clouds can eliminate the single point of failure problem [5]. Federating cloud resources can be a solution for interoperability among multiple clouds, enabling the formation of a pool of resources used for providing on demand services.

Virtualization is a major breakthrough enabling cloud providers to abstract the physical infrastructure, and to hide the complexity of underlying resources. They create a pool of virtualized resources which are offered to users as different types of VM instances. In this paper, we model the cloud federation formation as a coalitional game, where cloud providers decide to form a coalition

• *L. Mashayekhy, M. Nejad, and D. Grosu are with the Department of Computer Science, Wayne State University, Detroit, MI, 48202. E-mail: mlena@wayne.edu, mahyar@wayne.edu, dgrosu@wayne.edu*

(cloud federation) to allocate VMs dynamically, based on users' requests. The cloud federation tries to maximize the total profit obtained by serving the users' requests. The model that we consider consists of a set of cloud providers and a user that submits a request consisting of a number of different VM instances. A subset of cloud providers will form a federation in order to provide the requested VM instances. Based on the proposed federation formation game we design a cloud federation formation mechanism.

1.1 Our Contribution

We model the cloud federation formation as a hedonic game, a type of coalitional game, satisfying fairness and stability properties. We define the federation preference relations for the cloud federation formation game and design a cloud federation formation mechanism which allows the cloud providers to make their own decisions to form a federation yielding the highest total profit. In the proposed mechanism, federations of cloud providers decide to merge and split in order to form a federation providing requested resources as a service to the user. The mechanism also determines the individual profit of each participating cloud provider in the federation. Each cloud provider covers its incurred costs, and obtains a profit based on its market power. The mechanism provides a stable federation structure, that is, none of the cloud providers has incentives to merge to another federation or split from a federation to form another federation. We analyze the properties of our proposed cloud federation formation mechanism and perform extensive experiments to investigate its properties.

1.2 Related Work

The primary requirements for forming federations of cloud providers are discussed by Rochwerger *et al.* [6]. In order to support these requirements, Rochwerger *et al.* [3] introduced the Reservoir (resources and services virtualization without boundaries) model which allows two or more cloud providers to pool their resources together in order to provide services as a federated cloud. However, Reservoir does not provide any mechanism for forming cloud federations. Buyya *et al.* [7] presented the vision, challenges, and architectural elements of federated cloud computing environments. Their proposed framework supports scaling of applications across multiple cloud providers. Celesti *et al.* [8] introduced a cloud architecture that allows a cloud to build a federation with other clouds. Their model considers two types of clouds, home and foreign, where a home cloud is the cloud that is unable to fulfill its users' requests and forwards the requests to foreign clouds. Their model assumes that the foreign clouds could provide resources. It does not consider the incentives of the foreign clouds for providing resources to the home cloud. However, in our paper we consider the incentives of the cloud providers for forming federations and provide a mechanism for sharing the profit among the cloud providers in a federation. Goiri *et al.* [9] provided models that assist a cloud provider in making decisions on forming federations with public clouds in order to maximize its profit. Their study does not consider the incentives of the other clouds for providing resources. In addition, they did not consider different types of VMs and their heterogeneous resources. However, our work takes into account the incentives of all the participating cloud providers and the heterogeneity of VMs and cloud resources. Toosi *et al.* [10] proposed several resource provisioning policies helping the cloud providers increase their resource utilization and profit. They considered a model where the cloud providers can terminate the VMs, called spot VMs, whenever the profit for running such VMs is negative. Van den Bossche *et al.* [11] proposed a binary integer program formulation that minimizes the cost of outsourcing using a mix of public and private clouds. Their mechanism tries to maximize the utilization of the private cloud. However, their proposed method cannot find the solution on hybrid clouds in reasonable amount of time. Nordal *et al.* [12] proposed a system for managing computations in federated clouds. They also considered the applications' confidentiality constraints. Bin *et al.* [13] proposed a VM placement approach in a cloud federation considering multiple data privacy constraints. However, they did not consider the cost of outsourcing in the objective of their method. Chaisiri *et al.* [14] proposed an optimal VM provisioning algorithm using stochastic programming considering several cloud providers with the objective of maximizing profit. Bruneo [15] proposed performance evaluation techniques based on stochastic reward nets for federated clouds to predict and quantify the cost-benefit of a strategy portfolio and the corresponding quality of service experienced by users. Mihailescu and Teo [16] evaluated the impact of users' rationality in a federated cloud. Yang *et al.* [17] proposed a business-oriented federated cloud computing architecture for a specific type of applications, the real-time online interactive applications, such as multi-player online computer games. Their model is built on the concept of computation migration instead of VMs, and it does not consider the federation formation problem.

Researches approached the cloud resource management problem considering different objectives and points of view. Kesavan *et al.* [18] proposed a set of low-overhead management methods for managing the cloud infrastructure capacity to achieve a scalable capacity allocation for thousands of machines. Rodriguez and Buyya [19] proposed a meta-heuristic algorithm based on Particle Swarm Optimization for VM provisioning and scheduling strategies on IaaS. The proposed strategies minimize the overall workflow execution cost while meeting deadline constraints. Their approach considers dynamic provisioning, the heterogeneity of computing resources, and the variations in the VMs performance. Doyle et al. [20] proposed an algorithm that determines

to which data center the user requests should be routed, based on the relative priorities of the cloud operator. Their algorithm reduces the latency, carbon emissions, and operational costs. Mastroianni *et al.* [21] proposed an approach for the consolidation of VMs on two resources, that minimizes the power consumption while ensuring a good level of QoS. In their approach, decisions on the assignment and migration of VMs are driven by probabilistic processes and are based exclusively on local information. All these works addressed the resource management issues within a single cloud and not within federations of clouds.

Game theory-based resource allocation mechanisms for single clouds were proposed by Wei *et al.* [22] and Jalaparti *et al.* [23]. Zhang *et al.* [24] proposed online auction mechanisms for resource allocation in clouds. In our previous studies [25], [26], we proposed strategy-proof mechanisms for VM provisioning and allocation in clouds in order to maximize the profit. A game theoretic solution for dynamic resource allocation in a cloud federation was proposed by Hassan *et al.* [27]. The authors defined a price function for a cloud provider that gives incentives to other clouds to contribute resources and to form a federation. Mihailescu and Teo [28] proposed a strategy-proof dynamic pricing scheme for federated cloud environments. A revenue sharing mechanism for multiple cloud providers using stochastic linear programming games was proposed by Niyato *et al.* [29]. Their mechanism does not consider the cost that each cloud provider incurs. As a result, the solution of the revenue sharing is in the core of the proposed game. The model considers a fixed cooperation cost for each cloud provider. A cloud provider decides to join or not to join the federation based on the cooperation cost. Mashayekhy and Grosu [30] proposed a mechanism for solving the virtual organization formation problem in grids. The mechanism considers the incentives of the grid service providers while providing the required capabilities to execute the user's application. They also proposed a distributed mechanism for dynamic virtual organization formation in grids [31]. These mechanisms cannot be employed in cloud settings because the unique characteristics of cloud systems bring about new problems. Clouds necessitate the design of novel mechanisms considering virtualization, VM provisioning, VM allocation, and profit sharing among cloud providers. Li *et al.* [32] investigated profit maximization strategies in cloud federations, where VMs are sold through auctions. They proposed a truthful double auction-based mechanism for trading VMs within a federation, where clouds can buy and sell their resources. Samaan [33] proposed an economic model based on repeated games, to regulate capacity sharing in a cloud federation. Cloud providers' objective is to sell their unused capacity in the spot market, but they are uncertain of future workload fluctuations. Our paper is different from these studies since we consider the federation formation problem, where our proposed mechanism determines how cloud providers should provide the resources to fulfill users' requests.

1.3 Organization

This paper is organized as follows. In Section 2, we describe the cloud federation formation problem and the system model we consider. In Section 3, we describe the game theoretic framework used to design the proposed cloud federation formation mechanism. Then, we present the proposed mechanism and characterize its properties. In Section 4, we evaluate the mechanism by extensive simulation experiments. In Section 5, we summarize our results and present possible directions for future research.

2 Cloud Federation Framework

In this section, we describe the model of the system and introduce the problem of maximizing the profit within a cloud federation. We also introduce a coalitional game, called the cloud federation game, that serves as a basis for the development of our proposed cloud federation formation game and mechanism that will be presented in Section 3.

2.1 System Model

We first describe the system model consisting of a set of cloud providers, a broker as a mediator, and several cloud users. The broker is a trusted third party responsible for handling the federation formation tasks such as, receiving requests, executing the federation formation mechanism, receiving the payment from users, and dividing the profit among participating providers. We assume that a set of cloud providers $\mathcal{I} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m\}$ is available to provide resources in the form of VM instances to cloud users. The cloud providers offer n types of VM instances: $\mathcal{VM} = \{VM_1, \ldots, VM_n\}$, where each instance provides a specific number of cores, amount of memory, and amount of storage. The VM instance of type VM_j ($j = 1, \ldots, n$) is characterized by: (i) the number of cores, w_j^c; (ii) the amount of memory, w_j^m; and (iii) the amount of storage provided, w_j^s.

Each cloud provider $\mathcal{C}_i \in \mathcal{I}$ has a specific number of cores, amount of memory, and amount of storage available to share in a federation. Note that each provider reserves a specific capacity for its own users, and specifies the available capacity to be shared in the federation based on its load. We denote by N_i, the number of available cores of cloud provider $\mathcal{C}_i$, by M_i, the amount of available memory of cloud provider $\mathcal{C}_i$, and by S_i, the amount of available storage of cloud provider $\mathcal{C}_i$. Each provider $\mathcal{C}_i$ incurs cost when providing resources. For a cloud provider $\mathcal{C}_i$, we denote by c_{ij}, the cost associated with each VM instance of type VM_j, where $j = 1, \ldots, n$.

A user sends a request to a broker, consisting of the number of VM instances of each type needed. A request is denoted by $\mathcal{R} = \{r_1, \ldots, r_n\}$, where r_j is the number

TABLE 1: Notation

$\mathcal{I}$	Set of cloud providers $\{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$
N_i	Number of available cores of $\mathcal{C}_i \in \mathcal{I}$
M_i	Amount of available memory of $\mathcal{C}_i \in \mathcal{I}$
S_i	Amount of available storage of $\mathcal{C}_i \in \mathcal{I}$
$\mathcal{VM}$	Set of VMs $\{VM_1, \ldots, VM_n\}$
w_j^c	Number of cores in $VM_j \in \mathcal{VM}$
w_j^m	Amount of memory in $VM_j \in \mathcal{VM}$
w_j^s	Amount of storage in $VM_j \in \mathcal{VM}$
p_j	Price for an instance of type $VM_j \in \mathcal{VM}$
c_{ij}	Cost of an instance of type VM_j provided by $\mathcal{C}_i \in \mathcal{I}$
r_j	Number of requested VM instances of type VM_j
$\mathcal{R}$	User's request, $\mathcal{R} = \{r_1, \ldots, r_n\}$

of requested VM instances of type VM_j, $j = 1, \ldots, n$. The broker bills a user based on the allocated VM instances. To do so, the broker sets a price p_j on each type of VM instance VM_j, where $j = 1, \ldots, n$. The final price that the user pays for her request is independent of the cloud provider providing the VM instances. The final price paid by the user for each of the r_j VM instances of type VM_j is $r_j p_j$, where p_j is a fixed price for an instance of type VM_j. A broker has all the information about cloud providers such as their available resources and associated cost, and it is responsible for forming the federation. Table 1 summarizes the notation used throughout the paper.

A cloud federation $\mathcal{F}$ is a set of cloud providers, i.e., $\mathcal{F} \subseteq \mathcal{I}$. The objective of a cloud federation $\mathcal{F}$ is to maximize its profit. We formulate the cloud federation profit maximization problem for a given federation $\mathcal{F}$ as an integer program (IP), called IP-CFPM, as follows:

$$\text{Maximize} \sum_{\mathcal{C}_i \in \mathcal{F}} \sum_{j=1}^{n} x_{ij}(p_j - c_{ij}), \tag{1}$$

Subject to:

$$\sum_{j=1}^{n} w_j^c x_{ij} \leq N_i, \ (\forall \mathcal{C}_i \in \mathcal{F}), \tag{2}$$

$$\sum_{j=1}^{n} w_j^m x_{ij} \leq M_i, \ (\forall \mathcal{C}_i \in \mathcal{F}), \tag{3}$$

$$\sum_{j=1}^{n} w_j^s x_{ij} \leq S_i, \ (\forall \mathcal{C}_i \in \mathcal{F}), \tag{4}$$

$$\sum_{\mathcal{C}_i \in \mathcal{F}} x_{ij} = r_j, \ (\forall j = 1, \ldots, n), \tag{5}$$

$$\sum_{j=1}^{n} x_{ij} \geq 1, \ (\forall \mathcal{C}_i \in \mathcal{F}), \tag{6}$$

$$x_{ij} \geq 0, \text{ and is integer} \quad (\forall \mathcal{C}_i \in \mathcal{F} \text{ and } \forall j = 1, \ldots, n), \tag{7}$$

The decision variables x_{ij} represent the number of VM instances of type VM_j provided by cloud provider $\mathcal{C}_i$. The objective function (1) is the total profit obtained by the participating cloud providers in the federation $\mathcal{F}$. The total profit is equal to the revenue received from the user minus the cost incurred by the cloud providers. Constraints (2) ensure that the number of cores provided by a cloud provider participating in the federation is less than its available number of cores. Constraints (3) guarantee that the amount of memory provided by a cloud provider participating in the federation is less than the amount of its available memory. Constraints (4) ensure that the amount of storage provided by a cloud provider participating in the federation is less than the amount of its available storage. Constraints (5) guarantee that the number of VM instances assigned to the user for each type of VM by all cloud providers is exactly the number of VM instances requested by the user for that type of VM instance. Constraints (6) ensure that each cloud provider in the federation contributes at least one VM instance. These constraints force the cloud providers to contribute resources to the federation. Constraints (7) represent the integrality requirements for the decision variables.

2.2 Cloud Federation Game

In this section, we introduce the *cloud federation game*, a coalitional game that allows us to model federations and investigate the stability of different federation structures. This game model will be extended in Section 3 into a hedonic game, called the cloud federation formation game, that characterizes the process of federation formation and will serve as the basis for the design of our proposed federation formation mechanism. The reader is referred to [34] for preliminaries on coalitional game theory. We define a *cloud federation game* $(\mathcal{I}, v)$, as a coalitional game with transferable utility, where each cloud provider in $\mathcal{I}$ is a player in the game, and v is the *characteristic function*, defined on $\mathcal{F} \subseteq \mathcal{I}$. The characteristic function is the profit obtained when the cloud providers of federation $\mathcal{F}$ cooperate as a coalition. This function is a real-valued function such that $v : \mathcal{F} \rightarrow \mathbb{R}^+$ and $v(\emptyset) = 0$. We consider each cloud federation $\mathcal{F} \subseteq \mathcal{I}$ as a *coalition*. If all the cloud providers form a federation, i.e., $\mathcal{F} = \mathcal{I}$, we call it the *grand federation*.

We define the characteristic function for our proposed cloud federation game as follows:

$$v(\mathcal{F}) = \begin{cases} 0 & \text{if } |\mathcal{F}| = 0 \text{ or IP-CFPM is not feasible,} \\ P & \text{if } |\mathcal{F}| > 0 \text{ and IP-CFPM is feasible,} \end{cases} \tag{8}$$

where $|\mathcal{F}|$ is the cardinality of $\mathcal{F}$, and P is the total profit obtained by the federation (i.e., the value of IP-CFPM objective function).

A cloud federation game should satisfy two main properties, fairness and stability. The profit obtained by a federation should be *fairly* divided among the participating cloud providers. A federation should be *stable*, that is, the participating cloud providers should not have incentives to leave the federation. In the following, we explain these two properties of the proposed game in more details.

The value $v(\mathcal{F})$ of the federation $\mathcal{F}$ must be divided among its participating cloud providers based on a given rule that satisfies fairness. In this study, we consider a fair profit division rule based on the market share of the cloud providers. A cloud provider that contributes more resources in all the possible federations in which it participates should receive higher profit regardless of its resource allocation in the selected federation. We define, $\psi_{\mathcal{C}_i}(\mathcal{F})$, the *payoff* or the *share* of cloud provider $\mathcal{C}_i$ that is part of federation $\mathcal{F}$. In order to determine the payoff $\psi_{\mathcal{C}_i}(\mathcal{F})$, we employ the normalized Banzhaf value [35]. The *Banzhaf value* is a division of payoffs for the grand federation that takes into account the power of the players. The Banzhaf value of cloud provider $\mathcal{C}_i$ in the cloud federation game $(\mathcal{I}, v)$ is defined as follows:

$$\beta_{\mathcal{C}_i}(\mathcal{I}) = \frac{1}{2^{m-1}} \sum_{\mathcal{F} \subseteq \mathcal{I} \setminus \{\mathcal{C}_i\}} [v(\mathcal{F} \cup \{\mathcal{C}_i\}) - v(\mathcal{F})]. \quad (9)$$

The Banzhaf value represents the average marginal contribution of cloud provider $\mathcal{C}_i$ over all possible federations containing $\mathcal{C}_i$. The marginal contribution of $\mathcal{C}_i$ in a federation $\mathcal{F}$ is $v(\mathcal{F} \cup \{\mathcal{C}_i\}) - v(\mathcal{F})$, i.e., the difference between the value of a federation with and without $\mathcal{C}_i$. The *normalized Banzhaf value* is defined as

$$\mathcal{B}_{\mathcal{C}_i}(\mathcal{I}) = \frac{\beta_{\mathcal{C}_i}(\mathcal{I})}{\sum_{\mathcal{C}_j \in \mathcal{I}} \beta_{\mathcal{C}_j}(\mathcal{I})}. \quad (10)$$

The normalized Banzhaf value gives a fair way of dividing the grand federation's profit among its members. The profit that each member $\mathcal{C}_i$ receives in the grand federation is calculated as follows:

$$\psi_{\mathcal{C}_i}(\mathcal{I}) = \mathcal{B}_{\mathcal{C}_i}(\mathcal{I}) v(\mathcal{I}). \quad (11)$$

The payoff vector $\Psi(\mathcal{I}) = (\psi_{\mathcal{C}_1}(\mathcal{I}), \cdots, \psi_{\mathcal{C}_m}(\mathcal{I}))$ gives the payoff division for the grand federation. Computing the Banzhaf value for a game with a large number of players is NP-hard [36].

We now define, $\psi_{\mathcal{C}_i}(\mathcal{F})$, the payoff of cloud provider $\mathcal{C}_i$ participating in federation $\mathcal{F}$, as follows:

$$\psi_{\mathcal{C}_i}(\mathcal{F}) = \frac{\psi_{\mathcal{C}_i}(\mathcal{I})}{\sum_{\forall \mathcal{C}_j \in \mathcal{F}} \psi_{\mathcal{C}_j}(\mathcal{I})} v(\mathcal{F}). \quad (12)$$

In our setting, using the Banzhaf value for payoff division is more reasonable than using the Shapley value. The Shapley value [37] considers the order of the players entering the federations, when determining the payoffs. However, in our case such an order does not affect the value of the federations. The Banzhaf value assumes that each player is equally likely to join any federation. That means, each federation will form with equal probability.

TABLE 2: The characteristics of available VM instances.

	Small VM_1	Medium VM_2	Large VM_3	Extralarge VM_4
w_j^c (1.6GHz CPU)	1	2	4	8
w_j^m (GB Memory)	1.7	3.75	7.5	15
w_j^s (TB Storage)	0.22	0.48	0.98	1.99
p_j (price)	0.12	0.24	0.48	0.96

We analyze the stability of the grand federation using a solution concept for coalitional games, called the *core*. To define the core, we first need to introduce the concept of imputation, as follows.

Definition 1 (Imputation): An *imputation* is a payoff vector $(\psi_{\mathcal{C}_1}(\mathcal{I}), \cdots, \psi_{\mathcal{C}_m}(\mathcal{I}))$ satisfying:
i) $\psi_{\mathcal{C}_i}(\mathcal{I}) \geq v(\mathcal{C}_i), \forall \mathcal{C}_i \in \mathcal{I}$, and
ii) $\sum_{\mathcal{C}_i \in \mathcal{I}} \psi_{\mathcal{C}_i}(\mathcal{I}) = v(\mathcal{I})$.

Condition (i) guarantees that the profit obtained by each cloud provider $\mathcal{C}_i$ participating in the grand federation is not less than its profit obtained by acting alone. Condition (ii) ensures that the entire profit of the grand federation is divided among all cloud providers.

Definition 2 (Core): The *core* is a set of imputations satisfying $\sum_{\mathcal{C}_i \in \mathcal{F}} \psi_{\mathcal{C}_i}(\mathcal{I}) \geq v(\mathcal{F}), \forall \mathcal{F} \subseteq \mathcal{I}$.
That means, the profit of any federation is not greater than the sum of the payoffs of its participating cloud providers in the grand federation. The existence of a payoff vector in the core shows that the grand federation is stable. As a result, a payoff division is in the core if there is no incentive for any cloud provider to leave the grand federation to join another federation. In the case that the core does not exist (i.e., the grand federation is not stable), independent and disjoint federations would form.

In the following we consider an example that shows that the core of the proposed cloud federation game can be empty. We consider three cloud providers $\mathcal{I} = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$, and four types of VM instances $\mathcal{VM} = \{VM_1, VM_2, VM_3, VM_4\}$ representing small, medium, large, and extra large VM instances, respectively. The description of the VM instances is provided in Table 2. The instance types and pricing are similar to the ones used by Microsoft Azure [38].

We consider that a user requests one VM instance of type small, one VM instance of type medium, and one VM instance of type extra large. That is, the request of the user is $\mathcal{R} = \{1, 1, 0, 1\}$. In Table 3, we give the computing and storage capacity of each cloud provider, the cost and the price of each type of VM instance. For simplicity, we assume that all cloud providers have the same specifications. As an example, $\mathcal{C}_1$ incurs a cost of \$0.072 to provide one VM of type VM_1. The computing, memory, and storage capacity of $\mathcal{C}_1$ is 8 cores, 16 GB, and 2000 GB, respectively. Based on the request, the user pays \$1.32 to the federation, that is \$0.12+\$0.24+\$0.96.

TABLE 3: The cloud providers' settings.

	N_i	M_i	S_i	$\mathcal{VM}$	c_j	p_j
$\mathcal{C}_i$	8	16 GB	2000 GB	VM_1	0.072	0.12
				VM_2	0.168	0.24
				VM_3	0.378	0.48
				VM_4	0.839	0.96

TABLE 4: The value for each federation.

$\mathcal{F}$	$v(\mathcal{F})$
$\{\mathcal{C}_1\}$	0
$\{\mathcal{C}_2\}$	0
$\{\mathcal{C}_3\}$	0
$\{\mathcal{C}_1, \mathcal{C}_2\}$	0.24
$\{\mathcal{C}_1, \mathcal{C}_3\}$	0.24
$\{\mathcal{C}_2, \mathcal{C}_3\}$	0.24
$\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$	0.24

If $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ provide resources individually, then none of them can satisfy the user's request. The values $v(\mathcal{F})$ for all possible federations are given in Table 4. If the sum of the payoffs to individual cloud providers $\{\mathcal{C}_1, \mathcal{C}_2\}$, $\{\mathcal{C}_1, \mathcal{C}_3\}$, and $\{\mathcal{C}_2, \mathcal{C}_3\}$ is less than 0.24 in the grand federation $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$, then the set of cloud providers has incentive to deviate from the grand federation. Since $\psi_{\mathcal{C}_1}(\mathcal{I}) + \psi_{\mathcal{C}_2}(\mathcal{I}) \geq v(\{\mathcal{C}_1, \mathcal{C}_2\})$, $\psi_{\mathcal{C}_1}(\mathcal{I}) + \psi_{\mathcal{C}_3}(\mathcal{I}) \geq v(\{\mathcal{C}_1, \mathcal{C}_3\})$, and $\psi_{\mathcal{C}_2}(\mathcal{I}) + \psi_{\mathcal{C}_3}(\mathcal{I}) \geq v(\{\mathcal{C}_2, \mathcal{C}_3\})$, are not simultaneously satisfied, there is no payoff vector in the core, and thus, the core of the cloud federation game is empty. Therefore, the grand federation would not form.

Since the grand federation may not be stable, we propose a cloud federation formation mechanism in order to find a stable cloud federation. In the next section, we introduce the cloud federation formation game, where the focus is on how to form independent and disjoint federations. The cloud federation formation game will be the basis for the design of our proposed cloud federation formation mechanism.

3 Cloud Federation Formation Mechanism

As we showed in the previous section, the core of the cloud federation game can be empty. If the grand federation does not form, independent and disjoint federations would form. In this section, we introduce the proposed cloud federation formation game, present the proposed cloud federation formation mechanism and characterize its properties.

3.1 Federation Formation Framework

In this section, we investigate the coalitional structures in the cloud federation game when the grand federation does not form, i.e., the grand federation is not stable. A federation structure $\mathcal{FS} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_h\}$ is a partition of the set of cloud providers $\mathcal{I}$ such that each provider is a member of exactly one federation, *i.e.*, $\mathcal{F}_i \cap \mathcal{F}_j = \emptyset$ for all i and j, where $i \neq j$ and $\bigcup_{\mathcal{F}_i \in \mathcal{CF}} \mathcal{F}_i = \mathcal{I}$. We denote by Π the set of all federation structures. The total number of federation structures is B_m, where B_m is the m-th Bell number [39], and $m = |\mathcal{I}|$. Thus, finding the optimal federation structure via exhaustive search through all federation structures is not feasible.

The coalition formation [40] investigates the partitioning of the players into disjoint sets. In general, the problem of finding the optimal coalition structure in coalition formation is NP-complete [41]. Note that, only one of the federations in the federation structure is selected to provide the resources requested by the user. The remaining cloud providers may form other federations to service other requests of users. As a result, the formation of other federations with cloud providers outside of the selected federation does not affect the decision of the cloud providers participating in the selected federation.

To be able to model the cloud federation formation process, we augment the cloud federation game presented in subsection 2.2 with a preference relation over federations. The newly obtained game, called the *cloud federation formation game*, is a hedonic game [42], that is, a special type of coalitional game that considers players' preferences over coalitions. In hedonic games, players have preferences over coalitions. In the following, we present the definition of a hedonic game.

Definition 3 (Hedonic game): A hedonic game is a tuple $(\mathcal{I}, \succeq)$, where $\mathcal{I}$ is the set of players in the game, $\succeq_i$ is a reflexive, complete, and transitive preference relation defined on Π_i for player i, and Π_i is the set of subsets in $\mathcal{I}$ containing player i.

Each player knows whether it prefers to be in company of some players rather than others. If $A \succeq_i B$, player i prefers coalition B at most as much as coalition A. As in the case of the cloud federation game, the core does not exist for the cloud federation formation game.

Definition 4 (Cloud federation formation game): A cloud federation formation game is a pair $(\mathcal{I}, \succeq)$, where $\succeq_i$ is a reflexive, complete, and transitive binary relation on Π_i, and Π_i is the set of federations in $\mathcal{I}$ containing $\mathcal{C}_i$.

We define the *federation preference relation* $\succeq_i$ for each $\mathcal{C}_i$. This allows $\mathcal{C}_i$ to compare two federations and to indicate its preference to be a part of one of them. $A \succeq_i B$ implies that $\mathcal{C}_i$ prefers to be a member of federation A than to be a member of federation B, or at least it prefers both federations equally. In addition, $A \succ_i B$ indicates that $\mathcal{C}_i$ strictly prefers to be a member of A than a member of B.

To model the cloud federation formation as a hedonic game, we need to define the federation preference relation over all pairs of federations in Π_i. For all $\mathcal{C}_i \in \mathcal{I}$ and for all $\mathcal{F}, \mathcal{F}' \in \Pi_i$, we define $\succeq_i$ as

$$\mathcal{F} \succeq_i \mathcal{F}' \iff v(\mathcal{F}) \geq v(\mathcal{F}'). \quad (13)$$

That means a cloud provider prefers the federation that gives the higher profit. Using this preference relation,

every cloud provider can evaluate its preferences over the set of possible federations that the cloud provider can be a member of. Therefore, the objective of each cloud provider is to determine the membership in a federation that gives the highest profit.

Next, we define two comparison relations based on the preference relation $\succeq_i$. These comparison relations will be used in the design of our proposed cloud federation formation mechanism. The two comparison relations, called *merge comparison* $\gg_m$ and *split comparison* $\gg_s$, will allow us to decide if a federation is more preferred than other federations.

The *merge comparison* $\gg_m$ is defined as follows:

$$\begin{aligned}\{\mathcal{F} \cup \mathcal{F}'\} \;&\gg_m \{\mathcal{F}, \mathcal{F}'\} \Longleftrightarrow \\ &\{\forall \mathcal{C}_i \in \mathcal{F}; \{\mathcal{F} \cup \mathcal{F}'\} \succ_i \mathcal{F} \text{ and} \\ &\forall \mathcal{C}_j \in \mathcal{F}'; \{\mathcal{F} \cup \mathcal{F}'\} \succ_j \mathcal{F}'\}\end{aligned} \tag{14}$$

Equation (14) implies that federation $\{\mathcal{F} \cup \mathcal{F}'\}$ is preferred over two disjoint federations $\{\mathcal{F}, \mathcal{F}'\}$, if the profit obtained by federation $\{\mathcal{F} \cup \mathcal{F}'\}$ is greater than the profit obtained by the providers in $\mathcal{F}$, and it is greater than the profit obtained by the providers in $\mathcal{F}'$. As a result, all providers are able to improve the total profit.

The *split comparison* $\gg_s$ is defined as follows:

$$\begin{aligned}\{\mathcal{F}, \mathcal{F}'\} \;&\gg_s \{\mathcal{F} \cup \mathcal{F}'\} \Longleftrightarrow \\ &\{\exists \mathcal{C}_i \in \mathcal{F}; \mathcal{F} \succeq_i \{\mathcal{F} \cup \mathcal{F}'\} \text{ or} \\ &\exists \mathcal{C}_j \in \mathcal{F}'; \mathcal{F}' \succeq_j \{\mathcal{F} \cup \mathcal{F}'\}\}\end{aligned} \tag{15}$$

Equation (15) implies that $\{\mathcal{F}, \mathcal{F}'\}$ is preferred over $\{\mathcal{F} \cup \mathcal{F}'\}$, if at least one federation is able to keep the same amount of profit or to increase the profit of its members. Such a split preference is irrespective of the other cloud providers' preferences outside of that federation.

The two comparison relations defined above induce two rules [40] that will be used in the design of our proposed cloud federation formation mechanism:

Merge Rule: For any pair of federations $\mathcal{F}$ and $\mathcal{F}'$:
$\{\mathcal{F} \cup \mathcal{F}'\} \gg_m \{\mathcal{F}, \mathcal{F}'\} \Rightarrow$ Merge $\mathcal{F}$ and $\mathcal{F}'$.

Split Rule: For any federation $\{\mathcal{F} \cup \mathcal{F}'\}$:
$\{\mathcal{F}, \mathcal{F}'\} \gg_s \{\mathcal{F} \cup \mathcal{F}'\} \Rightarrow$ Split $\{\mathcal{F} \cup \mathcal{F}'\}$.

The merge rule implies that two federations join to form a larger federation if operating all of their cloud providers together strictly improves the total profit. The split rule implies that a federation splits only if there exists one sub-federation that obtains at least the same total profit with its constituent cloud providers. A split happens irrespective of the effect on the profit of the other sub-federations, i.e., their profit may decrease. Note that only one of the federations in the federation structure is selected to provide the resources requested by the user. Therefore, the goal of the cloud providers is to find a federation with maximum profit. This federation is obtained through an iterative application of the merge and the split rules.

As we mentioned before, computing the Banzhaf value for a game is NP-hard. However, through the iterative application of merge and split rules some of the possible federations are checked and their values are calculated. Based on those values, we define the *estimated Banzhaf value* of $\mathcal{C}_i$ as follows:

$$E_{\mathcal{C}_i}(\mathcal{I}) = \frac{1}{\lambda} \sum_{\substack{\mathcal{F} \subseteq \mathcal{I} \setminus \{\mathcal{C}_i\} \\ \mathcal{F} \in \mathcal{V} \\ \mathcal{F} \cup \mathcal{C}_i \in \mathcal{V}}} [v(\mathcal{F} \cup \{\mathcal{C}_i\}) - v(\mathcal{F})]. \tag{16}$$

where $\mathcal{V}$ is the set of all checked federations (i.e., federations that were already produced during the merge and split iterations), and λ is the total number of checked federations containing $\mathcal{C}_i$. That means, $\lambda = 2^{m-1} - \alpha$, where α is the number of non-checked federations. The estimated Banzhaf value is based only on the value of federations that are checked during the merge and split process. The *normalized estimated Banzhaf value* is defined as follows:

$$\mathcal{E}_{\mathcal{C}_i}(\mathcal{I}) = \frac{E_{\mathcal{C}_i}(\mathcal{I})}{\sum_{\mathcal{C}_j \in \mathcal{I}} E_{\mathcal{C}_j}(\mathcal{I})}. \tag{17}$$

The profit that each member $\mathcal{C}_i$ receives in the grand federation is calculated as follows:

$$\psi_{\mathcal{C}_i}(\mathcal{I}) = \mathcal{E}_{\mathcal{C}_i}(\mathcal{I}) v(\mathcal{I}). \tag{18}$$

The payoff vector $\Psi(\mathcal{I}) = (\psi_{\mathcal{C}_1}(\mathcal{I}), \cdots, \psi_{\mathcal{C}_m}(\mathcal{I}))$ gives the payoff divisions of the grand federation. We define $\psi_{\mathcal{C}_i}(\mathcal{F})$, the payoff of cloud provider $\mathcal{C}_i \in \mathcal{F}$, as follows:

$$\psi_{\mathcal{C}_i}(\mathcal{F}) = \frac{\psi_{\mathcal{C}_i}(\mathcal{I})}{\sum_{\forall \mathcal{C}_j \in \mathcal{F}} \psi_{\mathcal{C}_j}(\mathcal{I})} v(\mathcal{F}). \tag{19}$$

During the merge-and-split we estimate the Banzhaf value for each provider based only on the federations that were already explored. The profit obtained by the federation is divided among participating cloud providers in proportion to their power in the federation.

We define a new concept of stability similar to the stability of a coalition structure defined in the context of the hedonic games [42]. The difference from the stability concept in hedonic games is that we consider only one federation instead of a federation structure. This is due to the fact that only one federation of cloud providers is needed to form in order to fulfill a user request. As a result, our proposed stability notion is defined on the participating cloud providers in the obtained federation by the proposed mechanism. We define the *individual federation stability* as follows.

Definition 5 (Individual federation stability): A federation $\mathcal{F}$ is *individually federation stable* if there is no member $\mathcal{C}_i \in \mathcal{F}$ such that $\mathcal{F} \setminus \{\mathcal{C}_i\} \succeq_j \mathcal{F}$ for all $j \in \mathcal{F}$.

In other words, a federation $\mathcal{F}$ is individually federation stable if there is no cloud provider $\mathcal{C}_i \in \mathcal{F}$ that can leave $\mathcal{F}$ without making at least one cloud provider $\mathcal{C}_j \in \mathcal{F}$ unhappy.

In the next section, we introduce our proposed cloud federation formation mechanism and prove that it produces individually stable federations.

Algorithm 1 Cloud Federation Formation Mechanism (CFFM)

1: **Input:** Request $\mathcal{R}$
2: $\mathcal{V} = \emptyset$
3: $\mathcal{FS} = \{\{\mathcal{C}_1\}, \cdots, \{\mathcal{C}_m\}\}$
4: **for all** $\mathcal{F}_i \in \mathcal{FS}$ **do**
5: $v(\mathcal{F}_i)$ = Solve IP-CFPM($\mathcal{F}_i$)
6: $\mathcal{V} = \mathcal{V} \cup \mathcal{F}_i$
7: **repeat**
8: MergeFederations();
9: SplitFederation();
10: **until** No split happens
11: Find $\mathcal{F}_k = \arg\max_{\mathcal{F}_i \in \mathcal{FS}} \{v(\mathcal{F}_i)\}$
12: **for all** $\mathcal{C}_i \in \mathcal{F}_k$ **do**
13: Calculate $\psi_{\mathcal{C}_i}(\mathcal{F}_k)$ based on $\mathcal{V}$
14: $\mathcal{F}_k$ allocates and provides the requested VM instances.

Algorithm 2 MergeFederations()

1: **repeat**
2: Select two non-checked federations $\mathcal{F}_i, \mathcal{F}_j \in \mathcal{FS}$
3: $v(\mathcal{F}_i \cup \mathcal{F}_j)$ = Solve IP-CFPM($\mathcal{F}_i \cup \mathcal{F}_j$)
4: $\mathcal{V} = \mathcal{V} \cup \{\mathcal{F}_i \cup \mathcal{F}_j\}$
5: **if** $\mathcal{F}_i \cup \mathcal{F}_j \gg_m \{\mathcal{F}_i, \mathcal{F}_j\}$ **then**
6: $\mathcal{F}_i \leftarrow \mathcal{F}_i \cup \mathcal{F}_j$
7: $\mathcal{F}_j \leftarrow \emptyset$ {$\mathcal{F}_j$ is removed from $\mathcal{FS}$}
8: **until** No merge happens

3.2 Cloud Federation Formation Mechanism (CFFM)

The proposed cloud federation formation mechanism (CFFM), presented in Algorithm 1, relies on the merge and split rules defined in the previous section. The mechanism is executed by a broker.

CFFM receives a user request as input (line 1). CFFM uses $\mathcal{V}$ to store all checked federations during the merge-and-split process. The algorithm sets $\mathcal{V}$ to the empty set (line 2). First, an initial federation structure $\mathcal{FS}$ in which every individual cloud provider $\mathcal{C}_i \in \mathcal{I}$ is a federation $\mathcal{F}_i$, is formed (line 3). Then, CFFM solves IP-CFPM to find $v(\mathcal{F}_i)$ for each federation $\mathcal{F}_i$ (line 5). Note that if IP-CFPM is not feasible for a federation, i.e., IP does not have a solution, the value of the federation is zero. All singleton federations are added to $\mathcal{V}$ (line 6).

CFFM iteratively calls MergeFederations() and SplitFederation() functions (lines 7-10). CFFM exits from the merge-and-split process when there is no possibility for a further merge or a further split. Then, CFFM finds a federation, $\mathcal{F}_k$, with the highest total profit among all federations in the final federation structure (line 11). CFFM calculates the normalized estimated Banzhaf value as the individual profit for each participating cloud provider in $\mathcal{F}_k$ based on its marginal contributions in the set of checked federations, $\mathcal{V}$ (lines 12-13). The selected federation, $\mathcal{F}_k$, provides the requested VM instances to the user (line 14).

The MergeFederations() procedure is presented in Algorithm 2. This procedure checks all merge possibilities of any pair of federations in the federation structure. First, MergeFederations() randomly selects two non-checked federations in $\mathcal{FS}$, e.g., $\mathcal{F}_i$ and $\mathcal{F}_j$ (line 2). Then, it solves IP-CFPM to find the value of a new federation $\{\mathcal{F}_i \cup \mathcal{F}_j\}$, and adds the new federation to the set $\mathcal{V}$ (lines 3-4). If merge is more preferred, then federations $\mathcal{F}_i$ and $\mathcal{F}_j$ are merged. The new federation is saved in $\mathcal{F}_i$, and $\mathcal{F}_j$ is removed from $\mathcal{FS}$ (lines 5-7). Note that the members of $\mathcal{F}_i$ have changed, and thus, $\mathcal{F}_i$ can be selected again for the merge. The MergeFederations() procedure tries to find another pair of non-checked federations in the federation structure by repeating the procedure. The MergeFederations() procedure terminates if all pairs of federations are checked and a merge does not happen, or the grand federation forms.

Algorithm 3 SplitFederation()

1: **for all** $\mathcal{F}_i \in \mathcal{FS}$ where $|\mathcal{F}_i| > 1$ **do**
2: **for all** partitions $\{\mathcal{F}_j, \mathcal{F}_k\}$ of $\mathcal{F}_i$, where $\mathcal{F}_i = \mathcal{F}_j \cup \mathcal{F}_k, \mathcal{F}_j \cap \mathcal{F}_k = \emptyset$ **do**
3: $v(\mathcal{F}_j)$ = Solve IP-CFPM($\mathcal{F}_j$)
4: $v(\mathcal{F}_k)$ = Solve IP-CFPM($\mathcal{F}_k$)
5: $\mathcal{V} = \mathcal{V} \cup \mathcal{F}_j$
6: $\mathcal{V} = \mathcal{V} \cup \mathcal{F}_k$
7: **if** $\{\mathcal{F}_j, \mathcal{F}_k\} \gg_s \mathcal{F}_i$ **then**
8: $\mathcal{F}_i \leftarrow \mathcal{F}_j$
9: $\mathcal{FS} = \mathcal{FS} \cup \mathcal{F}_k$
10: **break**

The SplitFederation() procedure is presented in Algorithm 3. This procedure checks all split possibilities of any federation with more than one member in the federation structure. The SplitFederation() procedure tries to split a federation, e.g., $\mathcal{F}_i$, into two disjoint federations $\mathcal{F}_j$ and $\mathcal{F}_k$, where $\mathcal{F}_j \cup \mathcal{F}_k = \mathcal{F}_i$. Then, it solves IP-CFPM twice to find the value of $\mathcal{F}_j$ and $\mathcal{F}_k$ (lines 3-4). In addition, it adds the two federations into the set $\mathcal{V}$ since their values are calculated (lines 5-6). If the split is more preferred, then one of the splited federations, $\mathcal{F}_j$, is saved in $\mathcal{F}_i$, and the other splited federation, $\mathcal{F}_k$, is added to the federation structure (lines 7-10). Since the federation structure has changed after the split happened, CFFM executes another iteration of merge-and-split. Note that if none of the existing federations splits then CFFM terminates. This is due to the fact that the existing federations already have been checked for the merge in the MergeFederations() procedure. If only the MergeFederations() procedure is applied without applying the SplitFederation() procedure, then the mechanism converges very fast, but the obtained solution would be far from the optimal, since it does not allow the formation of smaller intermediate federations that can later on lead to better federations. However, by applying the SplitFederation() procedure, the mechanism iteratively improves the solution until it finds one that is closer to the optimal.

For a given user request, only one federation will form. When another request arrives and the cloud providers participating in some already formed federations have resources available, they can participate again along with other cloud providers to form a federation in order to serve the request. If a cloud provider does not have

any resources available, it waits until the federation that it belongs to dissolves, and then participates again in the federation formation mechanism to serve future requests.

3.3 CFFM Properties

In this section, we show that CFFM converges to a stable federation structure and analyze CFFM's time complexity. First, we prove that the proposed CFFM converges to a final federation structure.

Theorem 1: CFFM converges to a federation structure composed of disjoint federations of cloud providers.

Proof: Based on the proposed preference relation $\gg_m$, the resulting federation after each merge is more preferred than previous federations. This is also true for the resulting federations after each split using the proposed preference relation $\gg_s$. As a result, if a federation structure is created during the merge-and-split iterations, the mechanism cannot create that federation structure again by any further merge-and-split iterations. This is due to the fact that by any merge-and-split iteration, the mechanism finds more preferred federations. In addition, the total number of federation structures is finite. Therefore, the merge-and-split iterations terminate, and the final federation structure cannot be subject to any further merge and split. As a result, CFFM always converges. □

Since CFFM converges to a final federation structure, there is no possibility for further merge and split. Therefore, the final federation structure cannot be subject to any further change, that is, none of the federations in the federation structure can merge to another federation (or split into sub-federations) to form another federation structure.

Now, we prove that the final federation satisfies the individual stability property.

Theorem 2: CFFM produces an individually stable federation.

Proof: A federation $\mathcal{F}$ is individually federation stable if there is no cloud provider $\mathcal{C}_i \in \mathcal{F}$ that can leave $\mathcal{F}$ without making at least one cloud provider $\mathcal{C}_j \in \mathcal{F}$ unhappy. In the split procedure, CFFM checks if any cloud provider in a federation wants to leave its current federation by checking all the possibilities for split. If it finds such a cloud provider, CFFM applies the split rule. As a result, no cloud provider that is part of the final federation has incentive to leave the federation. □

Solving the federation formation problem optimally has the same complexity as performing an exhaustive search on all the possible partitions of the set of cloud providers. However, since CFFM is using the merge and split procedures, it does not perform an exhaustive search on the set of partitions. The time complexity of CFFM is determined by the number of merge and split operations and the size of the sub-partitions. In the worst case scenario, each federation needs to make a merge attempt with all the other federations in $\mathcal{FS}$. In the initial federation structure, where each of the m individual cloud providers is a federation, the first merge occurs after $\frac{m(m-1)}{2}$ attempts in the worst case. The second merge requires $\frac{(m-1)(m-2)}{2}$ attempts and so on. As a result, the total worst case number of merges is in $O(m^3)$. In the worst case scenario, splitting a federation F is in $O(2^{|F|})$, which involves finding all the possible partitions of size two of the participating federations.

TABLE 5: Cost

Amazon EC2 Regions	US East (N. Virginia)	US West (Oregon)	US West (Northern California)	EU (Ireland)	South America (Sao Paulo)	Asia Pacific (Singapore)	Asia Pacific (Sydney)	Asia Pacific (Tokyo)
Cloud Provider	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$	$\mathcal{C}_7$	$\mathcal{C}_8$
VM_1 (Small)	$0.03	$0.045	$0.048	$0.033	$0.055	$0.04	$0.058	$0.044
VM_2 (Medium)	$0.06	$0.091	$0.096	$0.065	$0.111	$0.08	$0.115	$0.088
VM_3 (Large)	$0.12	$0.182	$0.192	$0.130	$0.222	$0.16	$0.230	$0.175
VM_4 (Extralarge)	$0.24	$0.364	$0.384	$0.260	$0.444	$0.32	$0.460	$0.350

4 Experimental Results

We perform a set of simulation experiments which allows us to investigate how effective the proposed federation formation mechanism is in producing stable cloud federations.

4.1 Experimental Setup

For our experiments, we consider that eight independent cloud providers are participating. We set the costs and the types of VMs offered by each of these cloud providers to the types and costs of VMs offered by Amazon EC2 [43] in each of its eight regions. These cloud providers offer four types of VM instances as presented in Table 2. For the cost of VMs, we use the VM prices of On-Demand Instance Prices offered by Amazon EC2. We set the cost of VM instances to the half of the actual VM prices of Amazon EC2 regions. The detailed information regarding the cost of VMs are presented in Table 5. We relied only on EC2 information for its different regions because it is publicly available. Each of the cloud providers considered in the simulation are independent and the cost information from each of the Amazon EC2 regions is used only to setup their cost structure. Following Samaan [33] and Toosi *et al.* [10], we consider 1024 cores, 1740 GB of memory, and 225 TB of storage as the average of capacities of the cloud providers from which 40% is available for the federation. We generate 100 requests such that requests with less

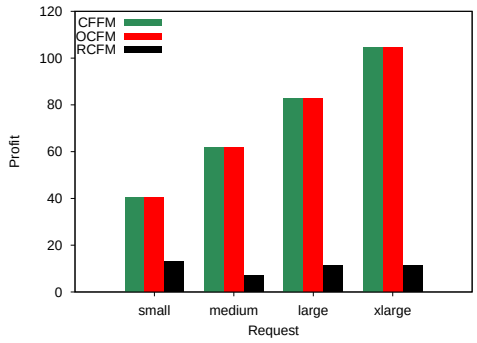

Fig. 1: Total profit of the cloud federation

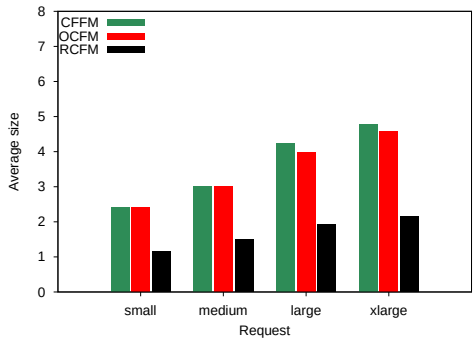

Fig. 2: Average size of the cloud federation

TABLE 6: Parameters

Param.	Description	Value(s)
m	Number of cloud providers	8
n	Number of VM types	4
N_i	Number of cores	[512, 1536]
M_i	Memory (GB)	[870, 2610]
S_i	Storage (TB)	[112, 338]
c_j	VM cost vector (4)	Based on Amazon Regions
p_j	VM price vector (4)	Based on Microsoft Azure

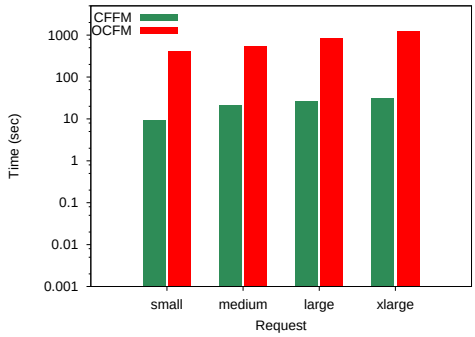

Fig. 3: Execution time of the mechanisms

than 15%, 25%, 35%, and 45% of the total available capacity belong to the small, medium, large, and extra-large classes, respectively. All requests cannot be served by only one cloud provider and they need to form a federation in order to serve the user. Each class contains 25 requests, and in the plots we represent the average of the obtained results. The parameters used in our experiments and their values are listed in Table 6.

While it is desirable to compare our proposed mechanism with several mechanisms, we found out that the existing mechanisms and approaches are not directly comparable to ours and decided to compare it with only two other mechanisms, Optimal Cloud Federation Mechanism (OCFM), and Random Cloud Federation Mechanism (RCFM). The OCFM mechanism finds the optimal solution to the federation formation problem, that is, it finds a cloud federation with maximum profit. This is achieved by exhaustively enumerating all the possible federations and solving IP-CFPM optimally for each of these federations. We rely on the optimal results obtained by OCFM as a benchmark for our experiments. We use the IBM ILOG Concert Technology APIs [44] in C++ to solve the integer program IP-CFPM associated with the mechanism. IBM ILOG provides optimization APIs and its engine is the CPLEX Optimizer that solves integer programming problems. The RCFM mechanism selects several cloud providers randomly and forms a federation. All the mechanisms are implemented in C++, and the experiments are conducted on AMD 2.93GHz hexa-core dual-processor systems with 90GB of RAM which are part of the Wayne State Grid System.

4.2 Analysis of Results

In Fig. 1, we compare the total profit obtained by CFFM with that obtained by the other two mechanisms. In all cases CFFM yields the highest profit which is very close to the optimal profit obtained by OCFM. These results show that RCFM achieves profits that are not even half the profits obtained by the other two mechanisms. In addition, the obtained total profit obtained by CFFM and OCFM increases with the increase in the size of the requests.

Fig. 2 shows the average number of cloud providers participating in the federation for each class of requests. As it is expected, with the increase in the size of the requests, more cloud providers need to participate in the federation to serve the requests. The size of the formed federations obtained by our proposed mechanism, CFFM, is close to that of the optimal solution. Note that in RCFM some of the federations are not feasible (i.e., the randomly chosen cloud providers cannot fulfill the request). Therefore, the average federation size of RCFM is very small.

Fig. 3 shows the execution time of the two mechanisms. The execution times of RCFM are negligible compared to that of CFFM and OCFM, and thus, we chose not to present them in the figure. From 255 federations that the eight cloud providers could form, CFFM only considers some of them in the merge-and-split process based on the merge and split rules. On average, CFFM explores 48 federations until it finds the final federation.

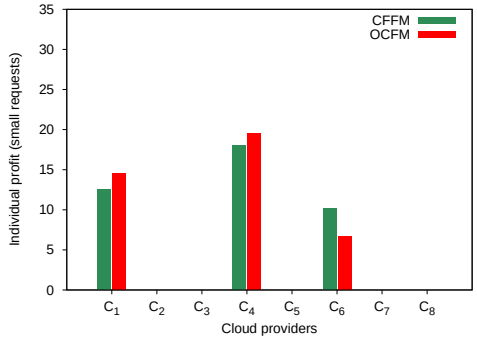

Fig. 4: Profit of cloud providers (small requests)

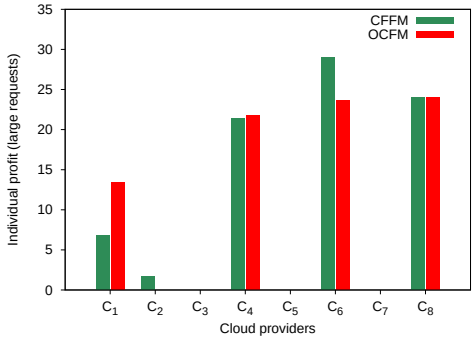

Fig. 6: Profit of cloud providers (large requests)

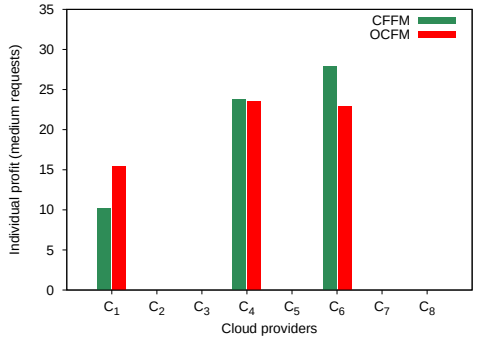

Fig. 5: Profit of cloud providers (medium requests)

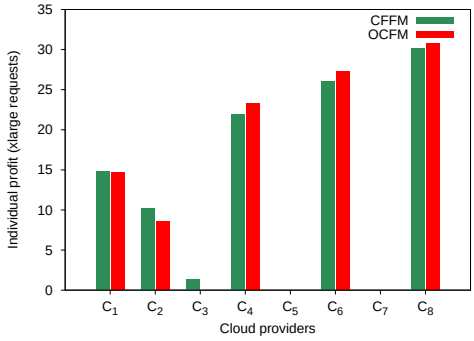

Fig. 7: Profit of cloud providers (extralarge requests)

As a result, the execution time of CFFM is a lot less than that of OCFM which goes through all the federations. The mechanisms require more time for larger requests. The execution time of our proposed mechanism, CFFM, is about two orders of magnitude less than that of the optimal mechanism OCFM.

Figs. 4 to 7 show the individual profit of each participating cloud provider in the federation, separately, for each class of requests. The mechanisms use different profit division rules as follows. CFFM uses the estimated normalized Banzhaf value, while the OCFM uses the normalized Banzhaf value. As it is shown in the figures, the individual profits of the participating cloud providers are very close in CFFM and OCFM. The difference between the average individual profit of CFFM and OCFM is 17%, 17%, 17% and 6% for small, medium, large, and extra-large requests presented in Figs. 4 to 7, respectively. The reason that the difference between the average individual profit of CFFM and OCFM for the extra-large requests are much less than those for the other requests is that the number of federations that are checked during the process of merge-and-split is higher, leading to a precise estimated normalized Banzhaf value. This is due to the fact that for larger requests the number of participating cloud providers in a federation increases resulting in more possibilities for merge-and-split operations. However, this comes at the cost of higher execution time.

We now investigate the performance of the proposed mechanism in more details. Figs. 8 to 11 show the percentage of participation of cloud providers in the federation, separately, for each class of requests. In all cases, the cloud provider with the lowest cost (i.e., $\mathcal{C}_1$) is a member of the formed federation. Fig. 8 shows that $\mathcal{C}_1$ and $\mathcal{C}_4$ participated in all formed federations obtained by CFFM and OCFM for small requests. For 40% of the requests a federation with size two does not have enough capacity to fulfill the requests, and thus, $\mathcal{C}_6$ participated in the formed federation on such cases. Figs. 9 shows that $\mathcal{C}_1$, $\mathcal{C}_4$, and $\mathcal{C}_6$ participated in all formed federations for the medium requests. This is due to the fact that their capacity is sufficient for the medium size requests. Note that all three cloud providers have to participate in the federations for the medium size requests unlike the small size requests case. In addition, as the size of the requests becomes larger, the cloud providers with higher costs are selected to participate in federations. For example, comparing Fig. 8 and Fig. 10 we observe that $\mathcal{C}_1$, $\mathcal{C}_4$, and $\mathcal{C}_6$ are members of the formed federations obtained by CFFM and OCFM. Based on Table 5, these cloud providers have the lowest costs. By increasing the size of the requests from small to large, the available capacity of these cloud providers are not sufficient enough. To fulfill the large requests (Fig. 10) federations with larger capacity are needed. The next lowest cost cloud provider is $\mathcal{C}_8$, and thus, $\mathcal{C}_8$ is included in the federations formed by CFFM and OCFM to provide the requested VMs. Note that the percentage of participation of cloud providers in the federation obtained by RCFM is distributed over

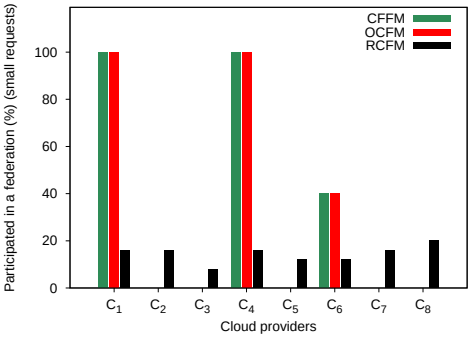

Fig. 8: Percentage participation of cloud providers (small requests)

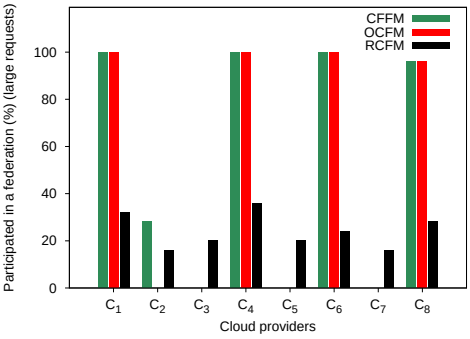

Fig. 10: Percentage participation of cloud providers (large requests)

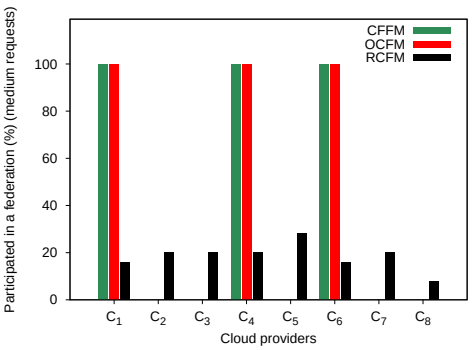

Fig. 9: Percentage participation of cloud providers (medium requests)

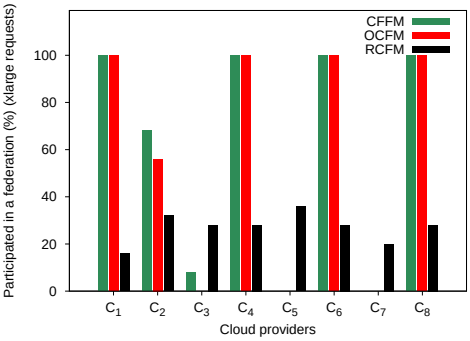

Fig. 11: Percentage participation of cloud providers (extralarge requests)

all cloud providers. This is due to the fact that RCFM selects the participating cloud providers in a federation randomly without considering their cost.

Figs. 12 to 15 show the percentage of requested cores provided to the federation by each cloud provider. For example, Fig. 12 shows that using CFFM and OCFM, 41.8% of the small requests are provided by $\mathcal{C}_1$, 53.4% of are provided by $\mathcal{C}_4$, and 4.8% are provided by $\mathcal{C}_6$. However, RCFM selects the cloud providers randomly to provide VMs for the small requests, and thus, the percentage of provided requests by cloud providers is far from the optimal solution. As shown in Fig. 13, the percentage of provided requests by $\mathcal{C}_1$, $\mathcal{C}_4$, and $\mathcal{C}_6$ changes to 27.5%, 41.3%, and 31.2% , respectively. The results show that with the increase in the size of the requests, the percentage of requests provided by $\mathcal{C}_6$ increases. This is due to the fact that more resources are needed to fulfill the demand of medium requests, and thus, $\mathcal{C}_6$ needs to provide more resources in the federation. Note that the amount of provided resources depends on the available capacity of the cloud providers.

From the above results, we conclude that our proposed mechanism, CFFM, is able to form stable federations with total profit very close to the optimal profit. In addition, CFFM finds the results in reasonable amount of time making it suitable for real cloud settings.

5 Conclusion

In this paper, we proposed a mechanism that improves the cloud providers' dynamic resource scaling capabilities to fulfill users' demands. We proposed a cloud federation formation game that characterizes the process of federation formation and then proposed a novel cloud federation formation mechanism called CFFM. In the proposed mechanism, cloud providers dynamically cooperate to form a federation in order to provide the requested resources to a user. The resources are provisioned as VM instances of different types. The proposed mechanism forms cloud federations yielding the highest total profit. The mechanism also determines the individual profit of each participating cloud providers in the federation using the normalized estimated Banzhaf value. In addition, our proposed mechanism produces a stable cloud federation structure, that is, the participating cloud providers in the federation do not have incentives to break away from the federation. We performed extensive experiments to investigate the properties of our proposed mechanism. The results showed that our proposed mechanism is able to form stable federations with total profit very close to the optimal profit. In addition, our mechanism finds the stable cloud federation in a reasonable amount of time making it suitable for real cloud settings. For the future work, we plan to incorporate the data privacy concerns into the federation

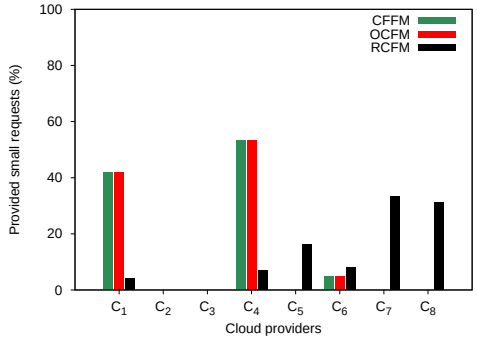

Fig. 12: Percentage of request provided by cloud providers (small requests)

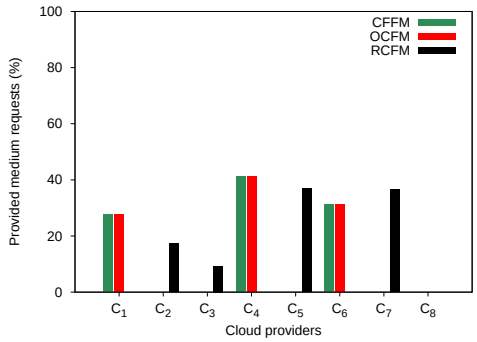

Fig. 13: Percentage of request provided by cloud providers (medium requests)

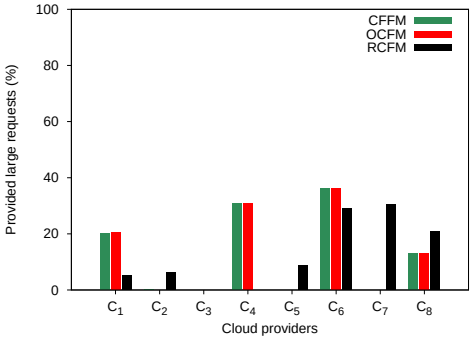

Fig. 14: Percentage of request provided by cloud providers (large requests)

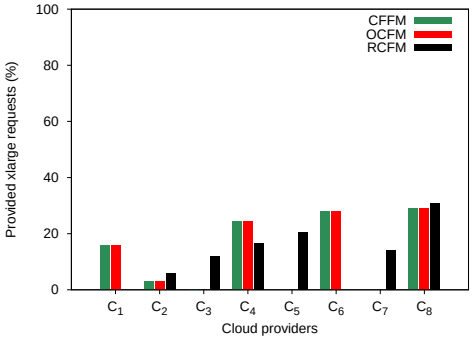

Fig. 15: Percentage of request provided by cloud providers (extralarge requests)

formation problem and to investigate the influence of cloud providers' policies on the federation formation process.

ACKNOWLEDGMENTS

This paper is a revised and extended version of [45] presented at the 5th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2012). The authors wish to express their thanks to the editor and the anonymous referees for their helpful and constructive suggestions, which considerably improved the quality of the paper. This research was supported, in part, by NSF grants DGE-0654014 and CNS-1116787.

REFERENCES

[1] L. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2008.

[2] M. Parashar, M. AbdelBaky, I. Rodero, and A. Devarakonda, "Cloud paradigms and practices for computational and data-enabled science and engineering," *Computing in Science & Engineering*, vol. 15, no. 4, pp. 10–18, 2013.

[3] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman *et al.*, "Reservoir-when one cloud is not enough," *Computer*, vol. 44, no. 3, pp. 44–51, 2011.

[4] B. Rochwerger, C. Vázquez, D. Breitgand, D. Hadas, M. Villari, P. Massonet, E. Levy, A. Galis *et al.*, "An architecture for federated cloud computing," *Cloud Computing*, pp. 391–411, 2010.

[5] D. Hilley, "Cloud computing: A taxonomy of platform and infrastructure-level offerings," *CERCS Report. College of Computing, George Institute of Technology*, 2009.

[6] B. Rochwerger, D. Breitgand, E. Levy, A. Galis *et al.*, "The reservoir model and architecture for open federated cloud computing," *IBM J. of Res. and Dev.*, vol. 53, no. 4, pp. 4–1, 2009.

[7] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Algorithms and architectures for parallel processing*. Springer, 2010, pp. 13–31.

[8] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation," in *Proc. of the 3rd IEEE Intl. Conf. on Cloud Computing*, 2010, pp. 337–345.

[9] I. Goiri, J. Guitart, and J. Torres, "Characterizing cloud federation for enhancing providers' profit," in *Proc. IEEE Intl. Conf. on Cloud Computing*, 2010, pp. 123–130.

[10] A. Toosi, R. Calheiros, R. Thulasiram, and R. Buyya, "Resource provisioning policies to increase iaas provider's profit in a federated cloud environment," in *Proc. of the 13th IEEE Intl. Conf. on High Performance Computing and Communications*, 2011, pp. 279–287.

[11] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads," in *Proc. of the 3rd IEEE Intl. Conf. on Cloud Computing*, 2010, pp. 228–235.

[12] A. Nordal, A. Kvalnes, J. Hurley, and D. Johansen, "Balava: Federating private and public clouds," in *Proc. of the IEEE World Congress on Services*, 2011, pp. 569–577.

[13] E. Bin, O. Biran, O. Boni, E. Hadad, E. K. Kolodner, Y. Moatti, and D. H. Lorenz, "Guaranteeing high availability goals for virtual machine placement," in *Proc. of the 31st IEEE Intl. Conf. on Dist. Comp. Syst.*, 2011, pp. 700–709.

[14] S. Chaisiri, B. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 164–177, 2012.

[15] D. Bruneo, "A stochastic model to investigate data center performance and qos in iaas cloud computing systems," *IEEE Transac-*

tions on Parallel and Distributed Systems, vol. 25, no. 3, pp. 560–569, 2014.
[16] M. Mihailescu and Y. M. Teo, "The impact of user rationality in federated clouds," in *Proc. of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2012, pp. 620–627.
[17] X. Yang, B. Nasser, M. Surridge, and S. Middleton, "A business-oriented cloud federation model for real-time applications," *Future Generation Computer Systems*, vol. 28, no. 8, pp. 1158–1167, 2012.
[18] M. Kesavan, R. Soundararajan, A. Gavrilovska, I. Ahmad, O. Krieger, and K. Schwan, "Practical compute capacity management for virtualized datacenters," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, pp. 88–100, 2013.
[19] M. Rodriguez and R. Buyya, "Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds," *IEEE Transactions on Cloud Computing*, vol. 99, no. PrePrints, p. 1, 2014.
[20] J. Doyle, R. Shorten, and D. O'Mahony, "Stratus: Load balancing the cloud for carbon emissions control," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, pp. 116–128, 2013.
[21] C. Mastroianni, M. Meo, and G. Papuzzo, "Probabilistic consolidation of virtual machines in self-organizing cloud data centers," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 215–228, 2013.
[22] G. Wei, A. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services," *The Journal of Supercomputing*, vol. 54, no. 2, pp. 252–269, 2010.
[23] V. Jalaparti and G. Nguyen, "Cloud resource allocation games," *Technical Report, University of Illinois*, 2010.
[24] H. Zhang, B. Li, H. Jiang, F. Liu, A. V. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," in *Proc. of IEEE INFOCOM*, 2013.
[25] L. Mashayekhy, M. M. Nejad, and D. Grosu, "A truthful approximation mechanism for autonomic virtual machine provisioning and allocation in clouds," in *Proc. of the ACM Cloud and Autonomic Computing Conference*, 2013, pp. 1–10.
[26] M. M. Nejad, L. Mashayekhy, and D. Grosu, "A family of truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds," in *Proc. of the 6th IEEE Intl. Conf. on Cloud Computing*, 2013, pp. 188–195.
[27] M. Hassan, B. Song, and E. Huh, "Distributed resource allocation games in horizontal dynamic cloud federation platform," in *Proc. IEEE Intl. Conf. on High Perf. Comp. and Comm.*, 2011, pp. 822–827.
[28] M. Mihailescu and Y. M. Teo, "Dynamic resource pricing on federated clouds," in *Proc. of the 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2010, pp. 513–517.
[29] D. Niyato, A. Vasilakos, and Z. Kun, "Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach," in *Proc. IEEE/ACM Intl. Symp. on Cluster, Cloud and Grid Comp.*, 2011, pp. 215–224.
[30] L. Mashayekhy and D. Grosu, "A merge-and-split mechanism for dynamic virtual organization formation in grids," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 540–549, 2014.
[31] ——, "A distributed merge-and-split mechanism for dynamic virtual organization formation in grids," in *Proc. 11th IEEE Intl. Conf. on Network Computing and Applications*, 2012, pp. 36–43.
[32] H. Li, C. Wu, Z. Li, and F. Lau, "Profit-maximizing virtual machine trading in a federation of selfish clouds," in *Proc. of the IEEE INFOCOM*, 2013, pp. 25–29.
[33] N. Samaan, "A novel economic sharing model in a federation of selfish cloud providers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 12–21, 2014.
[34] G. Owen, *Game Theory*, 3rd ed. New York, NY, USA: Academic Press, 1995.
[35] ——, "Multilinear extensions and the banzhaf value," *Naval Research Logistics Quarterly*, vol. 22, no. 4, pp. 741–750, 1975.
[36] Y. Matsui and T. Matsui, "Np-completeness for calculating power indices of weighted majority games," *Theoretical Computer Science*, vol. 263, no. 1-2, pp. 305–310, 2001.
[37] L. S. Shapley, "A value for n-person games," in *Contributions to the Theory of Games II*, ser. Ann. Math. Studies, H. W. Kuhn and A. W. Tucker, Eds. Princeton, New Jersey, USA: Princeton University Press, 1953, vol. 28, pp. 307–317.
[38] WindowsAzure: Purchase Options - Pricing. [Online]. Available: http://www.windowsazure.com/en-us/pricing/calculator/
[39] D. Knuth, *The Art of Computer Programming, Volume 4, Combinatorial Algorithms, Part 1*. Addison-Wesley, 2011.
[40] K. Apt and A. Witzel, "A generic approach to coalition formation," *International Game Theory Review*, vol. 11, no. 3, pp. 347–367, 2009.
[41] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme, "Coalition structure generation with worst case guarantees," *Artificial Intelligence*, vol. 111, pp. 209–238, 1999.
[42] A. Bogomolnaia and M. Jackson, "The stability of hedonic coalition structures," *Games & Econ. Behavior*, vol. 38, no. 2, pp. 201–230, 2002.
[43] Amazon EC2 Pricing. [Online]. Available: http://aws.amazon.com/ec2/pricing/
[44] IBM ILOG CPLEX V12.1 user's manual. [Online]. Available: ftp://public.dhe.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps_usrmancplex.pdf
[45] L. Mashayekhy and D. Grosu, "A coalitional game-based mechanism for forming cloud federations," in *Proc. of the 5th IEEE/ACM Intl. Conf. on Utility and Cloud Computing*, 2012, pp. 223–227.

Lena Mashayekhy received her BSc degree in computer engineering-software from Iran University of Science and Technology, and her MSc degree from the University of Isfahan. She is currently a PhD candidate in computer science at Wayne State University, Detroit, Michigan. Her research interests include distributed systems, cloud computing, big data, game theory and optimization. She is a student member of the ACM, the IEEE, and the IEEE Computer Society.

Mahyar Movahed Nejad received his BSc degree in mathematics from Iran University of Science and Technology. He received his MSc degree in socio-economic engineering from Mazandaran University of Science and Technology. He is currently a MSc student in computer science, and a PhD candidate in industrial and systems engineering at Wayne State University, Detroit. His research interests include distributed systems, big data analytics, game theory, network optimization, and integer programming. He is a student member of the IEEE and the INFORMS.

Daniel Grosu received the Diploma in engineering (automatic control and industrial informatics) from the Technical University of Iaşi, Romania, in 1994 and the MSc and PhD degrees in computer science from the University of Texas at San Antonio in 2002 and 2003, respectively. Currently, he is an associate professor in the Department of Computer Science, Wayne State University, Detroit. His research interests include parallel and distributed systems, cloud computing, parallel algorithms, resource allocation, computer security, and topics at the border of computer science, game theory and economics. He has published more than ninety peer-reviewed papers in the above areas. He has served on the program and steering committees of several international meetings in parallel and distributed computing. He is a senior member of the ACM, the IEEE, and the IEEE Computer Society.