A Trust-Aware Mechanism for Cloud Federation Formation

Lena Mashayekhy, *Member, IEEE,* Mark M. Nejad, *Member, IEEE,* and Daniel Grosu, *Senior Member, IEEE*

Abstract—Cloud providers can form cloud federations by pooling their resources together to balance their loads, reduce their costs, and manage demand spikes. However, forming cloud federations is a challenging problem, especially when considering the incentives of the cloud providers making their own decisions to participate in cloud federations. In this paper, we model the formation of cloud federations necessary to provide resources to execute Map-heavy/Reduce-heavy programs while considering the trust and reputation among the participating cloud providers. The objective is to form cloud federations with highly reputable cloud providers that achieve maximum profit for their participation. This is an NP-hard bicriteria optimization problem. We introduce a coalitional graph game, called trust-aware cloud federation formation game, to model the cooperation among cloud providers. We design a mechanism for cloud federation formation that enables the cloud providers with high reputation to organize into federations reducing their costs. Our proposed mechanism guarantees the highest profits for the participating cloud providers in the federations, and ensures high reliability of the formed federations in executing the applications. We perform extensive experiments to characterize the properties of the proposed mechanism. The results show that our proposed mechanism produces Pareto optimal and stable cloud federations that not only guarantee that the participating cloud providers have high reputation, but also high individual profits.

Index Terms—Cloud federation, Trust, Reputation, Coalitional game theory.

1 Introduction

Clouds are large-scale distributed computing systems offering their services based on the pay-as-you-go model. Due to attractive cost benefit ratios of cloud services, outsourcing is of massive interest to enterprises and individuals. A cloud provider provisions low level resources (e.g., CPUs, storage, etc.) of its physical machines (PMs) in the form of virtual machines (VMs) and containers which are then allocated to jobs. Clouds provide essential resources for compute- and data-intensive processing required to solve various challenging problems in science and engineering. With rapid changes in technologies (e.g., containers and Cloudlets) and emerging new paradigms for data-intensive processing (e.g., MapReduce and Spark), cloud providers are faced with many challenges when managing their resources. Efficient cloud resource management leads to efficient utilization of resources, faster execution of applications, and lower payment for cloud users.

Clouds can elastically scale up/down their resources through virtualization. However, the availability of virtual resources is limited by the capacities of the cloud providers' physical resources. To overcome such limitations, cloud providers can form federations with other cloud providers to borrow/lend resources according to particular agreements. Forming cloud federations not only allows small-medium cloud providers to cooperate and increase their market shares, but also provides cloud users with more cost-effective services and flexibility. In addition to managing demand spikes, cloud providers can also reduce their costs by forming federations. For example, when the electricity price is high at one cloud provider, that cloud provider can schedule jobs onto other cloud providers with lower electricity charges at that time to reduce its costs. Our focus is on designing mechanisms for formations of cloud federations to provide necessary resources for executing application programs.

Cloud users execute many types of large-scale application programs including MapReduce and Spark. A MapReduce program comprises a specific number of map and reduce jobs executed on clouds. In the map phase, each map job is allocated to a map slot on a node, and processes a portion of the input data producing key-value pairs. In the reduce phase, the key-value pairs with the same key are then processed by a reduce job allocated to a reduce slot. The output of the reduce phase is written back to the distributed file system [1], [2]. A Spark program runs an independent set of executor processes. Within each Spark program, multiple jobs (i.e., Spark actions) may be running concurrently. Each Spark program is divided into stages such as map and reduce phases. In this study, we consider applications that are either map-heavy (e.g., Wordcount) or reduce-heavy (e.g., TeraSort) [3].

We argue that the incentives of cloud providers are the main driving forces in the formation of cloud federations, and thus, it is essential to take them into account when designing cloud federation formation mechanisms. One key element in the formation of cloud federations in such highly dynamic environments is the cloud providers' reliability of

• *L. Mashayekhy is with the Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716.*
M. Nejad is with the Department of Civil and Environmental Engineering, University of Delaware, Newark, DE 19716.
D. Grosu is with the Department of Computer Science, Wayne State University, Detroit, MI 48202.
E-mail: mlena@udel.edu, nejad@udel.edu, dgrosu@wayne.edu

executing requested programs. If a cloud provider agrees to provide some resources in a federation, but it fails to deliver the promised resources, then the application program could not be executed by that federation. For example, a cloud provider promises to provide a VM with a specific number of CPUs and a specific amount of memory, but it actually provides a VM with fewer CPUs and less memory than promised, leading to the inability of the federation to complete the execution of the application by its deadline. Therefore, selecting highly trusted cloud providers to be part of the cloud federation is necessary to avoid this problem. In addition, a cloud provider desires to be a member of a cloud federation to obtain high profit. Therefore, a cloud federation formation mechanism should consider both profit and the trust among cloud providers when making cloud federation formation decisions.

In this paper, we define trust as how likely a cloud provider is to provide the requested resources (guarantee SLA) to another cloud provider based on their past direct interactions. If a cloud provider does not have past interactions with another cloud provider, it may use the reputation of that cloud provider. We define reputation as how other cloud providers that have had direct interactions with both cloud providers evaluate that cloud provider. Also, we define global reputation of a cloud provider as how likely the cloud provider is to provide the requested resources, based on the opinion of all cloud providers.

1.1 Our Contribution

We address the problem of cloud federation formation considering the trust relationships among cloud providers. The objective is to maximize the individual profit of the participating cloud providers in cloud federations and the overall reputation among the cloud providers. This problem is a bicriteria optimization problem, and it is NP-hard. We propose a framework for calculating the global reputation for each cloud provider considering direct trust and reputation. We then introduce a novel coalitional graph game, called the trust-aware cloud federation formation game with preferences, to model the problem in cloud environments. In addition, we propose a trust-aware cloud federation formation mechanism, TCFF, to solve this bicriteria optimization problem. Our proposed mechanism considers the incentives of the cloud providers and allows them to make their own decisions to participate in cloud federations. In addition, our proposed mechanism produces Pareto optimal solutions (i.e., there is no other solution, in our case a cloud federation, with both a higher payoff and a higher average reputation), and it converges to a stable cloud federation for requested applications. The performance of the proposed trust-aware cloud federation mechanism is analyzed both theoretically and experimentally. The experimental results show that the proposed mechanism produces stable cloud federations that not only guarantee the highest reputation among participating cloud providers, but also maximize their individual payoffs.

1.2 Organization

The rest of the paper is organized as follows. In Section 2 we review the related work. In Section 3, we describe the system model and the trust-aware cloud federation model. In Section 4, we describe the cloud federation game, the trust-aware cloud federation formation game, and the proposed mechanism. In Section 5, we evaluate the mechanism by extensive experiments. In Section 6, we summarize our results and present possible directions for future research.

2 Related Work

There exists an extensive body of research on resource allocation and VM placement in clouds focusing on optimizing system-wide performance objectives without taking into account the behavior of both users and cloud providers and their incentives to participate and contribute resources. Recently, economic-based cloud resource management mechanisms have been proposed that take into account the incentives of cloud users and cloud providers in VM provisioning, allocation, and pricing [4], [5], [6]. However, these studies focused on systems with a single cloud provider. The concept of federated clouds, where cloud providers can dynamically scale-up their resource capabilities by forming cloud federations, was introduced and studied by several researchers (e.g., [7], [8]). Goiri *et al.* [9] studied the design of cost-effective cloud federations and proposed mathematical models to help cloud providers decide when to outsource/insource resources to other cloud providers. Mashayekhy *et al.* [10] addressed the problem of federation formation in clouds and designed a coalitional game-based mechanism that enables the cloud providers to dynamically form a cloud federation maximizing their profit. Samaan [11] proposed an economic model based on repeated games, to regulate capacity sharing in a cloud federation, where each provider aims at maximizing its profit. Bruneo [12] 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 (QoS) experienced by users. However, none of the above mentioned studies considered the design of trust-aware mechanisms for cloud federation formation. Our framework is more suitable for modeling the cloud federation formation problem, since trust plays a key role in delivering the desired service level agreement.

Scheduling MapReduce jobs with different objectives has attracted a great deal of attention. For scheduling multiple MapReduce jobs [1], Hadoop originally employed a FIFO scheduler. By default, Spark's scheduler [13] also runs jobs in FIFO fashion. Zaharia *et al.* [14] proposed a robust scheduling algorithm, Longest Approximate Time to End (LATE), for the problem of speculative execution in MapReduce, which uses estimated finish times. Chang *et al.* [15] proposed MapReduce scheduling algorithms based on solving a linear program relaxation to minimize the overall MapReduce completion times. Zhan *et al.* [16] proposed a cooperative resource provisioning solution for MapReduce jobs using statistical multiplexing. However, the abovementioned studies either focus on single cloud provider or did not consider the underlying trust relations among cloud providers.

Several studies have focused on measuring reputation in different domains by using graphs in which the weight asso-

ciated with each edge represents the value of trust. Hang *et al.* [17] used trust propagation to find the reputation. Guha *et al.* [18] proposed a trust transitivity model based on the number of participants on a path. Agrawal *et al.* [19] used the network flow to find the reputation. Another approach to design reputation systems is to use graph centrality measures [20], where the centrality of a node determines its reputation. Trust is one major concern in grid [21], [22] and cloud [23], [24] environments. Several studies focused on considering the user's trust in cloud services (i.e., the trust between a user and a cloud provider) [25], [26], while our work considers the effects of the trust among cloud providers when forming a cloud federation. Ko *et al.* [23] proposed TrustCloud to address accountability in the cloud environment by technical and policy-based approaches. Messina *et al.* [27] proposed models to measure the trust and reliability of cloud services. Premarathne *et al.* [28] proposed a cloud-based utility service for user identity management based on trust establishment between the cloud service provider and the identity providers. Mashayekhy *et al.* [29] proposed a data protection framework when outsourcing VMs to a cloud federation. In this paper, we take into consideration both profit and reputation when deciding the formation of cloud federations.

3 Cloud Federation Framework

In this section, we describe the system model and the trust-aware cloud federation model.

3.1 System Model

We consider a system model in which a set of m cloud providers, $\mathcal{I} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m\}$, are willing to put their resources together to form federations. Each cloud provider $\mathcal{C} \in \mathcal{I}$ contributes a maximum number $R_{\mathcal{C}}$ of VM instances of a given type to a federation. Users submit their requests in order to execute their large-scale application programs on clouds. An application program $\mathcal{J}$ consists of a set of n independent jobs $\{J_1, J_2, \ldots, J_n\}$, and it needs to be completed by a given deadline d. Each application requests VM instances of the same type and a job is assigned to each VM instance (which is usually the case in practice). The cloud providers cooperate to execute $\mathcal{J}$. These cloud providers are autonomous entities driven by incentives and are self-interested. In game-theoretic sense, they are rational agents (i.e., utility maximizers). Each job J is characterized by its execution time at cloud provider $\mathcal{C}$ given by the execution time function $e : \mathcal{J} \times \mathcal{I} \rightarrow \mathbb{R}^+$. Once a job is assigned for execution to a given cloud provider, it is neither preempted nor migrated. A cloud provider $\mathcal{C} \in \mathcal{I}$ incurs cost for executing a job $J \in \mathcal{J}$. We characterize cloud providers' cost by the following cost function, $c : \mathcal{J} \times \mathcal{I} \rightarrow \mathbb{R}^+$. In addition, we consider that a user is willing to pay a price P for the service if the program is executed to completion by deadline d. If the program execution time exceeds the deadline d, the user will pay 0.

To make sure that a subset of cloud providers $\mathcal{F} \subseteq \mathcal{I}$, or a cloud federation, is able to execute the program $\mathcal{J}$, we need to find an assignment of all jobs $J \in \mathcal{J}$ to the cloud providers $\mathcal{C} \in \mathcal{F}$ in such a way that the job assignment satisfies all constraints. The job assignment problem finds an assignment of the n jobs of the application to a subset of cloud providers $\mathcal{F} \subseteq \mathcal{I}$. We define the following decision variables:

$$x_{J\mathcal{C}} = \begin{cases} 1 & \text{if job } J \text{ is assigned to cloud provider } \mathcal{C} \in \mathcal{F}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

We formulate the job assignment problem as an Integer Program (IP), called IP-ASSIGN, as follows:

$$\text{Minimize } C(\mathcal{J}, \mathcal{F}) = \sum_{J \in \mathcal{J}} \sum_{\mathcal{C} \in \mathcal{F}} c(J, \mathcal{C}) x_{J\mathcal{C}}, \tag{2}$$

Subject to:

$$\sum_{J \in \mathcal{J}} \sum_{\mathcal{C} \in \mathcal{F}} c(J, \mathcal{C}) x_{J\mathcal{C}} \leq P, \tag{3}$$

$$\sum_{J \in \mathcal{J}} x_{J\mathcal{C}} \leq R_{\mathcal{C}}, \ (\forall \mathcal{C} \in \mathcal{F}), \tag{4}$$

$$\sum_{J \in \mathcal{J}} x_{J\mathcal{C}} \geq 1, \ (\forall \mathcal{C} \in \mathcal{F}), \tag{5}$$

$$\sum_{\mathcal{C} \in \mathcal{F}} x_{J\mathcal{C}} = 1, \ (\forall J \in \mathcal{J}), \tag{6}$$

$$e(J, \mathcal{C}) x_{J\mathcal{C}} \leq d, \ (\forall \mathcal{C} \in \mathcal{F}, \forall J \in \mathcal{J}), \tag{7}$$

$$x_{J\mathcal{C}} \in \{0, 1\}, \ (\forall \mathcal{C} \in \mathcal{F} \text{ and } \forall J \in \mathcal{J}). \tag{8}$$

The objective function (2) is to minimize the execution cost of the application program $\mathcal{J}$ on $\mathcal{F}$. Constraint (3) ensures that the execution cost of the program $\mathcal{J}$ on $\mathcal{F}$ does not exceed the payment. Constraints (4) ensure that the assigned jobs to each cloud provider does not exceed the cloud provider's capacity. Constraints (5) ensure that each cloud provider $\mathcal{C} \in \mathcal{F}$ is assigned at least one job. Constraints (6) ensure that each job is assigned to exactly one cloud provider. Constraints (7) guarantee that the execution of the assigned jobs to each cloud provider does not exceed the deadline. Constraints (8) specify that the decision variables are binary.

3.2 Trust-Aware Cloud Federation Model

We model the trust relationship among cloud providers as a weighted directed graph $(\mathcal{I}, E)$, called the *trust graph*, where $\mathcal{I}$ is a set of cloud providers representing the nodes, and E is a set of edges. The nodes in the trust graph do not have self-loops (i.e., there is no edge $(\mathcal{C}_i, \mathcal{C}_i) \in E$). The weight $\tau(\mathcal{C}_i, \mathcal{C}_j)$ associated with edge $(\mathcal{C}_i, \mathcal{C}_j)$ represents the amount of *trust* that $\mathcal{C}_i$ assigns to $\mathcal{C}_j$, showing the strength of the trust relationship from $\mathcal{C}_i$ to $\mathcal{C}_j$, based on their past interactions (i.e., it does not depend on other cloud providers' trust on $\mathcal{C}_j$). The weight can take any value within the interval $[0, V]$, where V is chosen by the designer of the system and corresponds to the highest trust. Trust can be an asymmetric relationship. In cases that two cloud providers did not have any interactions in the past (i.e., $\tau(\mathcal{C}_i, \mathcal{C}_j) = 0$), they can rely on opinions of the other cloud providers.

A cloud provider rates another cloud provider based on its direct trust which can be used for local ratings. In doing so, the value of direct trust is normalized between 0

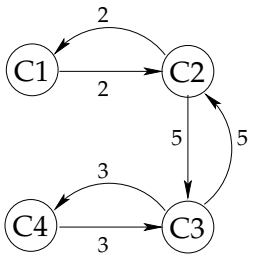

Fig. 1: An example of a trust graph.

and 1 so that each cloud provider assigns a single rating (so-called *local trust value* or normalized trust) to another cloud provider. We define $w(\mathcal{C}_i, \mathcal{C}_j) \in [0, 1]$ to be the *normalized trust* that $\mathcal{C}_i$ assigns to $\mathcal{C}_j$. Each cloud provider $\mathcal{C}_i$ computes the normalized trust value $w(\mathcal{C}_i, \mathcal{C}_j)$ by dividing the local trust $\tau(\mathcal{C}_i, \mathcal{C}_j)$ by the sum of the local trust values assigned to all its neighbor cloud providers as follows:

$$w(\mathcal{C}_i, \mathcal{C}_j) = \frac{\tau(\mathcal{C}_i, \mathcal{C}_j)}{\sum_{\mathcal{C}_k \in \mathcal{N}(\mathcal{C}_i)} \tau(\mathcal{C}_i, \mathcal{C}_k)} \tag{9}$$

where, $\mathcal{N}(\mathcal{C}_i) = \{\mathcal{C}_j | \exists (\mathcal{C}_i, \mathcal{C}_j) \in E\}$ is the set of $\mathcal{C}_i$'s neighbors. In addition, for each cloud provider $\mathcal{C}_i \in \mathcal{I}$, we have:

$$\sum_{\mathcal{C}_j \in \mathcal{I}, \mathcal{C}_i \neq \mathcal{C}_j} w(\mathcal{C}_i, \mathcal{C}_j) = 1. \tag{10}$$

Note that a normalized trust is not used to rank neighbors of different cloud providers or across all providers. We denote by W, the matrix of normalized trust of the graph $(\mathcal{I}, E)$, where its elements, $w(\mathcal{C}_i, \mathcal{C}_j)$, represent the normalized trust values.

In Fig. 1 we give an example of a trust graph with four cloud providers $\mathcal{I} = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$. As an example, $\tau(\mathcal{C}_2, \mathcal{C}_3)$ is 5, which is the amount of trust that $\mathcal{C}_2$ assigns to $\mathcal{C}_3$. Then, the normalized trust computed using equation (9) is $w(\mathcal{C}_2, \mathcal{C}_3) = 0.71$.

To form a cloud federation, $\mathcal{C}_i$ should be able to select a cloud provider $\mathcal{C}_j$ based not only on its local trust $w(\mathcal{C}_i, \mathcal{C}_j)$, but also on the trust the other cloud providers have on $\mathcal{C}_j$. In doing so, we consider the reputation of cloud providers in the federation formation rather than their local trust, considering all the paths that ends with that node and the quality of the paths. We need to define a metric that characterizes cloud providers' reputations, where the metric measures how likely a cloud provider is to provide the requested resources based on all other cloud providers' opinions.

To determine the *global reputation* of each cloud provider within a given set $\mathcal{I}$ of cloud providers, we use the following procedure, called Cloud-REP, which is based on the power method [30]. We denote by $\rho^q_{\mathcal{C}_i \to \mathcal{C}_j}$ the trust $\mathcal{C}_i$ assigns to $\mathcal{C}_j$ based on the opinion of the q^{th} neighboring cloud providers (e.g., for $q = 1$ we consider neighbors of $\mathcal{C}_i$, for $q = 2$ we consider neighbors of neighbors, and so on). Cloud-REP starts by determining $\rho^0_{\mathcal{C}_i \to \mathcal{C}_j}$, the local trust $\mathcal{C}_i$ assigns to $\mathcal{C}_j$, as follows:

$$\rho^0_{\mathcal{C}_i \to \mathcal{C}_j} = w(\mathcal{C}_i, \mathcal{C}_j) \tag{11}$$

To find the reputation between cloud providers $\mathcal{C}_i$ and $\mathcal{C}_j$, $\mathcal{C}_i$ uses its neighbors opinions about $\mathcal{C}_j$ by weighting their opinions using the trust $\mathcal{C}_i$ places on them. As a result, we calculate $\rho^1_{\mathcal{C}_i \to \mathcal{C}_j}$ as follows:

$$\rho^1_{\mathcal{C}_i \to \mathcal{C}_j} = \sum_{\mathcal{C}_k \in \mathcal{I}} w(\mathcal{C}_k, \mathcal{C}_j) \cdot \rho^0_{\mathcal{C}_i \to \mathcal{C}_k} \tag{12}$$

The intuition behind equation (12) is that it aggregates the local trust values of all cloud providers and computes the reputation of cloud providers using the transitive property of the trust. As a result, Cloud-REP facilitates trust propagation and trust aggregation. In trust propagation, the transitivity of trust is considered, and in the trust aggregation, the trust transitivity of different paths is aggregated. Cloud-REP continues this procedure by considering the neighbors of neighbors, and so on, which improves the accuracy of trust propagation. As a result, $\rho^q_{\mathcal{C}_i \to \mathcal{C}_j}$ is calculated as follows:

$$\rho^q_{\mathcal{C}_i \to \mathcal{C}_j} = \sum_{\mathcal{C}_k \in \mathcal{I}} w(\mathcal{C}_k, \mathcal{C}_j) \cdot \rho^{q-1}_{\mathcal{C}_i \to \mathcal{C}_k} \tag{13}$$

Let $\boldsymbol{\rho}^q_{\mathcal{C}_i}$ denote the vector that contains all the reputation scores that $\mathcal{C}_i$ assigns to the other cloud providers using q^{th} neighboring cloud providers. In other words, the length of a path from $\mathcal{C}_i$ to other cloud providers in the trust graph $(\mathcal{I}, E)$ is q. Using the matrix notation, equation (13) for all cloud providers $\mathcal{C}_i \in \mathcal{I}$, can be written as follows:

$$\boldsymbol{\rho}^q_{\mathcal{C}_i} = W^T \cdot \boldsymbol{\rho}^{q-1}_{\mathcal{C}_i} \tag{14}$$

If q is large, $\mathcal{C}_i$ will assign a reputation score to each cloud provider considering the opinion of all cloud providers. Note that if all other cloud providers do the same to find the reputation scores of all cloud providers, they will find the same reputation scores as in $\boldsymbol{\rho}^q_{\mathcal{C}_i}$. As a result, $\boldsymbol{\rho}^q_{\mathcal{C}_i}$ converges to the *global reputation* of the cloud providers (i.e., the global reputation vector $\boldsymbol{\rho}$). This vector is the left principal eigenvector of W, that is, it satisfies:

$$\lambda \boldsymbol{\rho} = W^T \boldsymbol{\rho} \tag{15}$$

where λ is the eigenvalue of W. As a result, the Cloud-REP procedure determines the global reputation of each cloud provider. By using this method, we convert the trust values between each pair of cloud providers into a global reputation for each cloud provider. The i-th component ρ_i of the eigenvector $\boldsymbol{\rho}$, then, gives the global reputation score of $\mathcal{C}_i$. For simplicity, we use the notation $\rho_{\mathcal{C}_i}$ to represent ρ_i of the eigenvector $\boldsymbol{\rho}$. Using this method, a cloud provider has high reputation to the extent that the cloud provider is connected to other cloud providers who have high reputation [31]. Here, the eigenvector $\boldsymbol{\rho}$ determines the *centrality* of the cloud providers based on their reputation.

We also define the *average global reputation* for a set of cloud providers $\mathcal{I}$ as follows:

$$\bar{\rho}(\mathcal{I}) = \frac{1}{|\mathcal{I}|} \sum_{\mathcal{C}_i \in \mathcal{I}} \rho_{\mathcal{C}_i} \tag{16}$$

The average reputation will be used in the next sections as a metric to characterize the aggregate reputation of the members of a cloud federation.

4 Trust-Aware Cloud Federation Formation Game and Mechanism

In this section, we formulate the trust-aware cloud federation formation as a coalitional game, and we describe our proposed trust-aware mechanism for cloud federation formation.

4.1 Cloud Federation Game

We now introduce the cloud federation game as a coalitional game. Coalitional game theory studies the interactions between groups of decision-makers (i.e., players). In coalitional games, players can cooperate and form alliances, while ultimately trying to maximize utility. We define a *cloud federation game* as a coalitional game with transferable utility as follows:

Definition 1 (cloud federation game). A cloud federation game is a pair $(\mathcal{I}, v)$, where $\mathcal{I}$ is the set of cloud providers (i.e., players in the game) and v is the *characteristic function* of the game. The characteristic function is defined on a federation $\mathcal{F} \subseteq \mathcal{I}$, such that $v : \mathcal{F} \rightarrow \mathbb{R}^+$ and $v(\emptyset) = 0$.

We define a federation (or a *coalition*) as a subset of cloud providers (i.e., $\mathcal{F} \subseteq \mathcal{I}$) working together to provide a service. If all the cloud providers form a federation, we call it the *grand federation*.

A federation has a *value* which is equal to the profit obtained by its members working as a group. Such a value is determined by the characteristic function $v(\mathcal{F})$. We define the characteristic function of a federation $\mathcal{F}$ for our proposed cloud federation game $(\mathcal{I}, v)$ as follows:

$$v(\mathcal{F}) = \begin{cases} 0 & \text{if } |\mathcal{F}| = 0 \text{ or IP is not feasible,} \\ P - C(\mathcal{J}, \mathcal{F}) & \text{if } |\mathcal{F}| > 0 \text{ and IP is feasible} \end{cases} \tag{17}$$

In this function, $|\mathcal{F}|$ is the cardinality of $\mathcal{F}$, P is the user payment to execute the program $\mathcal{J}$, $C(\mathcal{J}, \mathcal{F})$ is the execution cost of the program on the federation $\mathcal{F}$, and $P - C(\mathcal{J}, \mathcal{F})$ is the obtained profit by the federation.

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.

A federation $\mathcal{F}$ would have to divide its value $v(\mathcal{F})$ among its members if the federation was formed. Traditionally, the Shapley value [32] would be employed to determine the individual payoff (or share), but computing the Shapley value is NP-complete [33] which requires iterating over every partition of a coalition. Another rule for payoff division is equal sharing of the profit among members. Equal sharing provides a tractable way to determine the shares and has been extensively used as a payoff division rule in other systems where tractability is critical (e.g., [34]). Following these studies, *equal sharing* is considered as a fair profit division rule among cloud providers in a federation. This is motivated by the fact that each cloud provider in the cloud federation plays a critical role in executing a subset of jobs of the application in order for the cloud federation to be able to execute the whole application. The execution of the application depends on delivering the promised resources by cloud providers in the cloud federation. As a result, it is in cloud providers' interest to equally share the profit for executing the application. In addition, this method provides a tractable way to divide the profit. We define the individual payoff of cloud provider $\mathcal{C}$ in federation $\mathcal{F}$, denoted by $\psi_{\mathcal{C}}(\mathcal{F})$, as follows:

$$\psi_{\mathcal{C}}(\mathcal{F}) = \frac{P - C(\mathcal{J}, \mathcal{F})}{|\mathcal{F}|}. \tag{18}$$

Therefore, the payoff divisions of the grand federation is represented by the payoff vector $\psi(\mathcal{I}) = (\psi_{\mathcal{C}_1}(\mathcal{I}), \cdots, \psi_{\mathcal{C}_m}(\mathcal{I}))$.

We analyze the stability of the grand federation using the most popular solution concept of a coalitional game, called the *core*. The core is the set of payoff vectors that make it favorable for the cloud providers to form the grand federation. The existence of a payoff vector in the core guarantees that the grand federation is stable with respect to any deviation by any group of cloud providers. The core of the cloud federation game $(\mathcal{I}, v)$ can be empty. This is due to the fact that cloud providers incur costs to form a federation to execute a program, thus, they prefer to participate in smaller coalitions instead of the grand coalition. In such a case, the grand federation does not form, leading to the formation of independent and disjoint federations. Partitioning the set of cloud providers into disjoint sets is called *cloud federation formation* which is a coalitional game in partition form.

Definition 2 (federation structure). A federation structure $\mathcal{FS} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_h\}$ is a partitioning of $\mathcal{I}$ such that each cloud 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 $\bigcup_{\mathcal{F}_i \in \mathcal{FS}} \mathcal{F}_i = \mathcal{I}$.

We denote by Π the set of all federation structures. In the next subsection, we introduce the trust-aware cloud federation formation game, where the focus is on how to form independent and disjoint federations considering trust relationship among cloud providers.

4.2 Trust-Aware Cloud Federation Formation Game

The formation of a federation depends on not only the value of the federation, but also the trust relations among cloud providers which are members of the federation. Therefore, a cloud provider prefers to join a federation with higher value which is composed of cloud provider members that have higher reputation scores. The reputation of cloud providers in a federation means how much reputation each cloud provider has, based on the opinions of all cloud providers in that federation.

We define a *trust-aware cloud federation game* as a coalitional graph game as follows:

Definition 3 (trust-aware cloud federation game). A trust-aware cloud federation formation game is a 3-tuple $(\mathcal{I}, E, v)$, where $\mathcal{I}$ is the set of cloud providers (i.e., players in the game), $\mathcal{G} = (\mathcal{I}, E)$ is a graph representing the trust relationship among the cloud providers, and v is the characteristic function of the game.

Cloud providers in the game are identified with nodes in a graph $\mathcal{G}$. The trust among cloud providers in a federation $\mathcal{F}$ influences the cloud federation formation.

The proposed game $(\mathcal{I}, E, v)$ captures two competing issues: cloud providers want to minimize the costs they incur in forming a federation by working with high-reputation cloud providers, but at the same time they want this federation to provide them with a high profit. As a result, the objective is to find a federation $\mathcal{F}^*$ such that its members have the highest average global reputation (as defined in Section 3.2) and the federation provides the maximum individual profit for its members.

We investigate the federation structures in the trust-aware cloud federation game when the grand federation does not form, i.e., the grand federation is not stable. To be able to model the federation structures, we need to augment the trust-aware cloud federation game presented in Definition 3 with a preference relation over federations, and define a *trust-aware cloud federation formation game with preferences* as follows:

Definition 4 (trust-aware cloud federation formation game with preferences). A trust-aware cloud federation formation game is a 4-tuple $(\mathcal{I}, E, v, \succeq)$, where $\mathcal{I}$ is the set of cloud providers, $\mathcal{G} = (\mathcal{I}, E)$ is the trust graph, v is the characteristic function, $\succeq$ is the federation preference relation defined over a set of graph partitions $\Pi_{\mathcal{C}}^{\mathcal{G}}$ for cloud provider $\mathcal{C}$, and $\Pi_{\mathcal{C}}^{\mathcal{G}}$ is the set of sub-graphs in $\mathcal{G}$ containing cloud provider $\mathcal{C}$.

Our proposed game includes a partitioning of the trust graph $\mathcal{G} = (\mathcal{I}, E)$ into disjoint sub-graphs. We define a sub-graph $\mathcal{S} = (\mathcal{F}, \mathcal{E})$ of $\mathcal{G} = (\mathcal{I}, E)$, where $\mathcal{F}$ is the set of cloud providers in the federation, and $\mathcal{E}$ is a set of trust edges among cloud providers in $\mathcal{F}$. We denote by $W_{\mathcal{F}}$, the matrix containing the trust values of the cloud providers in $\mathcal{F}$. We define reputation scores of the cloud providers in a federation $\mathcal{F}$ using the sub-graph $\mathcal{S}$. We define a graph-based federation structure $\mathcal{GFS} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_h\}$ as partitioning of $\mathcal{G}$ into sub-graphs $\mathcal{S}_1(\mathcal{F}_1, \mathcal{E}_1)$, $\mathcal{S}_2(\mathcal{F}_2, \mathcal{E}_2)$, ..., $\mathcal{S}_h(\mathcal{F}_h, \mathcal{E}_h)$ such that each cloud provider is a member of exactly one federation represented by a sub-graph. We denote by $\Pi^{\mathcal{G}}$ the set of all graph-based federation structures.

In order to compare federations, we define the *federation preference relation* $\succeq_{\mathcal{C}}$ for each $\mathcal{C}$. This allows $\mathcal{C}$ to compare two federations in $\Pi_{\mathcal{C}}^{\mathcal{G}}$ and to indicate its preference to be a part of one of them. $A \succeq_{\mathcal{C}} B$ implies that $\mathcal{C}$ 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, $A \succ_{\mathcal{C}} B$ indicates that $\mathcal{C}$ strictly prefers to be a member of A than a member of B.

Each cloud provider prefers to maximize its profit, thus, participating in a low profit federation with high profit shares is more preferred than participating in a high profit federation with low profit shares. As a result, a cloud provider prefers a federation that provides the highest individual profit among all possible federations. In addition, a cloud provider prefers a federation which members have the highest average reputation. The average reputation for cloud providers in $\mathcal{F}$ is defined as follows:

$$\bar{\rho}(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{\mathcal{C}_i \in \mathcal{F}} \rho_{\mathcal{C}_i} \tag{19}$$

where $\rho_{\mathcal{C}_i}$ is the reputation score of $\mathcal{C}_i$ as computed by the Cloud-REP procedure presented in Section 3.2. As a result, we define the federation preference relation $\succeq_{\mathcal{C}}$ as follows:

Definition 5 (federation preference relation). For all $\mathcal{F}, \mathcal{F}' \subseteq \mathcal{I}$ and $\mathcal{C} \in \mathcal{F}, \mathcal{F}'$, we have:

$$\mathcal{F} \succeq_{\mathcal{C}} \mathcal{F}' \text{ if } [\psi_{\mathcal{C}}(\mathcal{F}) \geq \psi_{\mathcal{C}}(\mathcal{F}') \text{ and } \bar{\rho}(\mathcal{F}) \geq \bar{\rho}(\mathcal{F}')] \tag{20}$$

The main intuition behind the federation preference relation $\succeq_{\mathcal{C}}$ is to allow a cloud provider to choose to join a federation that gives higher profit and has higher average reputation. 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, in order to determine its preferred federation $\mathcal{F}$, each cloud provider $\mathcal{C}$ solves a bicriteria optimization problem as follows:

$$\begin{cases} \max_{(\mathcal{F})} & \psi_{\mathcal{C}}(\mathcal{F}), \text{ and} \\ \max_{(\mathcal{F})} & \bar{\rho}(\mathcal{F}). \end{cases} \tag{21}$$

This is a bicriteria optimization problem in which the cloud provider's goal is to maximize both the obtained profit share and the average global trust within a federation. Since the job assignment problem (i.e., equations (2)-(8)) is NP-hard for $m \geq 2$ [35], the bicriteria problem in equation (21) is also NP-hard.

To solve this bicriteria optimization problem, we will asses the optimality of the solutions using the concept of *Pareto optimality*. Pareto optimality refers to the set of solutions of the bicriteria problem that are not dominated by other solutions in both criteria. In our problem, the two criteria are the individual payoff and the average reputation. Thus, a solution $\mathcal{F}$ yielding an individual payoff of ψ and an average reputation of $\bar{\rho}$ is Pareto optimal if there is no other solution $\mathcal{F}'$ with both a higher payoff ψ' and a higher average reputation $\bar{\rho}'$ (i.e., $\psi' \geq \psi$ and $\bar{\rho}' \geq \bar{\rho}$). The Pareto optimal solutions are not unique and they form a set of Pareto optimal solutions. In the next section, we will show that our proposed mechanism obtains one such solution from the set of Pareto optimal solutions.

We are interested in characterizing the solution. In order to do this, we formally define the rationality and stability concepts that we use in this paper.

Definition 6 (Individual rationality). A game satisfies the *individual rationality* if for any formed federation $\mathcal{F}$ and all its members $\mathcal{C} \in \mathcal{F}$, we have $\psi_{\mathcal{C}}(\mathcal{F}) \geq 0$.

In other words, each cloud provider does not suffer any loss by participating in the game.

We define *individual stability* as follows:

Definition 7 (Individual stability). A federation $\mathcal{F}$ is *individually stable* if there is no member $\mathcal{C} \in \mathcal{F}$ such that $\mathcal{F} \setminus \{\mathcal{C}\} \succeq_{\mathcal{C}'} \mathcal{F}$ for all $\mathcal{C}' \in \mathcal{F}$.

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

In the next subsection, we present our proposed mechanism for forming trust-aware cloud federations, and we investigate its properties.

Algorithm 1 Trust-Aware Cloud Federation Formation Mechanism (TCFF)

1: **Input:** $\mathcal{J}$, $\mathcal{I}$, and $\mathcal{G}$
2: $\mathcal{L} = \emptyset$ \\ *initial set of federations*
3: $\mathcal{F} = \mathcal{I}$ \\ *initial cloud federation*
4: $\mathcal{S} = \mathcal{G}$ \\ *the trust graph*
5: **repeat**
6: *flag* $\leftarrow$ TRUE
7: Solve IP-ASSIGN for $\mathcal{J}$ on federation $\mathcal{F}$
8: **if** FEASIBLE **then**
9: $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{F}$
10: *flag* $\leftarrow$ FALSE
11: **end if**
12: $\boldsymbol{\rho}$ = Cloud-REP($\mathcal{S}(\mathcal{F}, \mathcal{E})$)
13: Find a cloud provider $\mathcal{C}$ with the lowest reputation in $\boldsymbol{\rho}$
14: $\mathcal{F} = \mathcal{F} \setminus \mathcal{C}$
15: Update $\mathcal{S}$
16: **until** *flag*
17: Find $\mathcal{F}^* = \arg\max_{\mathcal{F} \in \mathcal{L}} \{v(\mathcal{F})/|\mathcal{F}|\}$ \\ *final federation*
18: Assign and execute program $\mathcal{J}$ on federation $\mathcal{F}^*$
19: **for all** $\mathcal{C} \in \mathcal{F}^*$ **do**
20: $\psi_{\mathcal{C}}(\mathcal{F}^*) = \frac{v(\mathcal{F}^*)}{|\mathcal{F}^*|}$ \\ *final payoffs*
21: **end for**

4.3 Trust-Aware Cloud Federation Formation Mechanism (TCFF)

The proposed trust-aware cloud federation formation mechanism (TCFF) is given in Algorithm 1. TCFF has three input parameters: the set of jobs of the application program $\mathcal{J}$, the set of cloud providers $\mathcal{I}$, and their trust graph $\mathcal{G}$. The mechanism uses $\mathcal{L}$ to keep a set of feasible federations, and it is initialized with the empty set. TCFF initializes $\mathcal{F}$ with the grand federation, that is, all cloud providers form a federation initially. It also initializes $\mathcal{S}$ with the trust graph $\mathcal{G}$. In every iteration (lines 5-16), TCFF solves IP-ASSIGN using a branch-and-bound method to find an optimal assignment for the application program $\mathcal{J}$ on federation $\mathcal{F}$. If IP-ASSIGN finds a feasible assignment of all jobs $J \in \mathcal{J}$ to $\mathcal{F}$ satisfying the constraints, TCFF adds federation $\mathcal{F}$ to $\mathcal{L}$. Then, TCFF calculates the reputation values of all cloud providers in the sub-graph $\mathcal{S}(\mathcal{F}, \mathcal{E})$ using the method described in Section 4 by calling Cloud-REP($\mathcal{S}$) function presented in Algorithm 2. TCFF selects a cloud provider $\mathcal{C}$ with the lowest reputation in sub-graph $\mathcal{S}(\mathcal{F}, \mathcal{E})$ using the output of Cloud-REP function, $\boldsymbol{\rho}$. Then, the mechanism removes $\mathcal{C}$ from the federation and the sub-graph (by removing all edges with direct trust to $\mathcal{C}$). If more than one cloud provider have the same lowest reputation, TCFF chooses one of them randomly to be removed from the federation. Note that the mechanism uses greedy choice in each step by removing the cloud provider with the lowest reputation from the federation. In every iteration, TCFF recalculates the reputation scores for all remaining cloud providers in the federation. It is worth noting that this recalculation step is necessary since the reputation of the federation members should be based on their opinion about the participating cloud providers in the federation. The opinion of the cloud providers with the lowest reputation should not affect the value of the reputation of the other cloud providers. This recalculation affects cloud providers' reputations in the entire federation. As a result, the mechanism considers only the opinions of the participating cloud providers when calculating the global reputation, that is, the opinions of cloud providers outside the federation do not have any effect on the reputation of the federation's members.

Algorithm 2 Cloud-REP($\mathcal{S}(\mathcal{F}, \mathcal{E})$)

1: **Input:** Trust graph: $\mathcal{S}(\mathcal{F}, \mathcal{E})$
2: $W_{\mathcal{F}}$ = the normalized trust matrix of $\mathcal{S}(\mathcal{F}, \mathcal{E})$
3: **for all** $\mathcal{C} \in \mathcal{F}$ **do**
4: $\rho_{\mathcal{C}}^0 \leftarrow \frac{1}{|\mathcal{F}|}$
5: **end for**
6: **repeat**
7: $\boldsymbol{\rho}^{q+1} \leftarrow W_{\mathcal{F}}^T \boldsymbol{\rho}^q$
8: $\delta \leftarrow ||\boldsymbol{\rho}^{q+1} - \boldsymbol{\rho}^q||$
9: **until** $\delta < \epsilon$
10: **return** $\boldsymbol{\rho}^{q+1}$

These iterations continue until TCFF finds a federation that could not execute the program. Finally, TCFF chooses a federation $\mathcal{F}^*$ from $\mathcal{L}$ that yields the highest individual payoff for its members, meaning that it maximizes the payoff that an individual cloud provider can receive. The program is executed by the federation $\mathcal{F}^*$, a trusted subset of $\mathcal{I}$. This federation contains members with high reputation and yields the highest individual payoff for them.

Cloud-REP function, given in Algorithm 2, receives the subgraph $\mathcal{S}$ as an input parameter and finds its normalized trust matrix of $W_{\mathcal{F}}$. Cloud-REP function starts by assigning the same reputation score to all cloud providers in the federation $\mathcal{F}$. Then, it recomputes the reputation scores of each cloud provider $\mathcal{C}$ as the weighted sum of the scores of all cloud providers in $\mathcal{C}$'s neighborhood. Cloud-REP function repeats these steps (lines 6-9) until the average relative error between $\boldsymbol{\rho}^{q+1}$ and $\boldsymbol{\rho}^q$ is smaller than a given threshold ϵ. In other word, $\boldsymbol{\rho}^q$ does not change significantly any more, and it represents the global reputation of the cloud providers in $\mathcal{F}$. Finally, Cloud-REP function returns the eigenvector that represents the reputation of cloud providers in $\mathcal{F}$.

We now investigate the properties of our proposed mechanism. We are interested in characterizing the rationality, stability, and Pareto optimality of the federation obtained by TCFF.

Theorem 1. TCFF produces federations satisfying the individual rationality.

Proof: TCFF selects the final federation from a set $\mathcal{L}$ by finding a federation $\mathcal{F}^*$ with the maximum individual payoff $v(\mathcal{F}^*)/|\mathcal{F}^*|$. For all cloud providers in $\mathcal{F}^*$, we must have: $\psi_{\mathcal{C}}(\mathcal{F}^*) \geq 0$. Note that set $\mathcal{L}$ contains feasible solutions to the IP-ASSIGN. As a result, we have $v(\mathcal{F}^*) \geq 0$, and thus, for all $\mathcal{C} \in \mathcal{F}^*$, $\psi_{\mathcal{C}}(\mathcal{F}^*) = \frac{v(\mathcal{F}^*)}{|\mathcal{F}^*|} \geq 0$. This proves the individual rationality property. □

Theorem 2. TCFF produces cloud federations that are individually stable.

Proof: To prove that TCFF forms an individually stable federation, we consider two cases that show that by removing a cloud provider from the formed federation, at least one of the participating cloud providers in the federation becomes unhappy.

i) Cloud provider $\mathcal{C}$ has the lowest reputation among all cloud providers in the federation $\mathcal{F}$. Note that TCFF has already checked this case when finalizing the federation $\mathcal{F}$.

TABLE 1: The program settings.

	cost				time			
	J_1	J_2	J_3	J_4	J_1	J_2	J_3	J_4
$\mathcal{C}_1$	3	4	6	4	3	6	9	6
$\mathcal{C}_2$	4	6	9	6	2	4	6	4
$\mathcal{C}_3$	8	10	20	10	1.5	3	4.5	3
$\mathcal{C}_4$	6	9	15	9	1	2	3	2

As a result, there are two possibilities, either the cloud federation is not feasible, or the individual profit of cloud providers in $\mathcal{F} \setminus \mathcal{C}$ is not as much as the individual profit of cloud providers in $\mathcal{F}$. As a result, removing $\mathcal{C}$ from the final federation $\mathcal{F}$ makes other cloud providers in $\mathcal{F}$ unhappy.

ii) Cloud provider $\mathcal{C}$ does not have the lowest reputation among all cloud providers in the federation $\mathcal{F}$. In this case, removing $\mathcal{C}$ decreases the total reputation of cloud providers participating in $\mathcal{F}$, which makes these cloud providers unhappy.

As a result, the formed federation is individually stable. □

We now investigate another important characteristic of the solution produced by TCFF, the optimality of the cloud federation in terms of both profit and reputation. We prove that the formed federation by TCFF is a Pareto optimal solution for the trust-aware cloud federation formation problem.

Theorem 3. TCFF produces a Pareto optimal solution to the trust-aware cloud federation formation problem.

Proof: We note that the set $\mathcal{L}$ (in Algorithm 1) contains the feasible federations formed by TCFF. To prove the Pareto optimality (defined in Section 4.1) of the solution, we need to show that the final federation $\mathcal{F} \in \mathcal{L}$ obtained by TCFF is not dominated by any other federation in both individual payoff and average reputation. We consider that the individual payoff of cloud providers in the final federation $\mathcal{F}$ is ψ, and their average reputation is $\bar{\rho}$.

The highest reputable cloud providers are always in the formed federation $\mathcal{F}$. This is due to the fact that in each iteration, TCFF removes a cloud provider with the lowest reputation. As a result, the cloud providers outside the federation $\mathcal{F}$ are not able to form a federation with higher average reputation than $\bar{\rho}$. That means, there is no federation $\mathcal{F}' \notin \mathcal{L}$, where $\bar{\rho}' \geq \bar{\rho}$. However, there may be other federations $\mathcal{F}' \in \mathcal{L}$ that have higher average reputation. Those federations do not have a higher individual payoff than ψ. This is due to the fact that TCFF selects $\mathcal{F}$ which has the highest individual payoff among all federations in $\mathcal{L}$. As a result, the federation formed by TCFF is a Pareto optimal solution. □

4.4 Example: TCFF Execution

To show how TCFF mechanism works, we consider an example with four cloud providers, each with a capacity of 5 VMs, where their trust graph is shown in Fig. 1. We assume that a user submits a four-job program $\mathcal{J} = \{J_1, J_2, J_3, J_4\}$ with a deadline $d = 10$ seconds and a payment $P = 50$. In Table 1, we give the cost of executing each job on each cloud provider, and the execution time of each job on each cloud provider. As an example, $\mathcal{C}_1$ needs 6 seconds to execute J_2, incurring a cost of 4 units. The normalized trust matrix is:

$$W = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.29 & 0 & 0.71 & 0 \\ 0 & 0.63 & 0 & 0.37 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

TABLE 2: The average reputation for each federation.

$\mathcal{F}$	$\frac{v(\mathcal{F})}{\lvert\mathcal{F}\rvert}$	Average reputation
$\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$	5.5	0.445435
$\{\mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$	6.33	0.57723
$\{\mathcal{C}_2, \mathcal{C}_3\}$	8.5	0.707107
$\{\mathcal{C}_3\}$	0	1

Using the Cloud-REP procedure described in Algorithm 2 the global reputations of cloud providers are $\rho_{\mathcal{C}_1} = 0.178174, \rho_{\mathcal{C}_2} = 0.62361, \rho_{\mathcal{C}_3} = 0.712697$ and $\rho_{\mathcal{C}_4} = 0.267261$. The individual payoff, $\frac{v(\mathcal{F})}{|\mathcal{F}|}$, and the average reputation are given in Table 2. If all cloud providers form a grand federation to execute the program, J_1, J_2, J_3, and J_4 are assigned to $\mathcal{C}_4$, $\mathcal{C}_2$, $\mathcal{C}_1$, and $\mathcal{C}_3$, respectively. The value of the grand federation is $v(\mathcal{I}) = 50 - 28 = 22$. The individual payoff of each cloud provider is 5.5, and the average global reputation is $\bar{\rho}(\mathcal{I}) = 0.4454$. Based on the eigenvector values, $\mathcal{C}_1$ has the lowest reputation among all cloud providers and it is removed from the grand federation. Federation $\{\mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$ based on opinion of all its members finds the reputation values $\rho_{\mathcal{C}_2} = 0.36076, \rho_{\mathcal{C}_3} = 1.15446$ and $\rho_{\mathcal{C}_4} = 0.216461$. In this federation, the individual value of the participating cloud providers is 6.33. In this step, $\mathcal{C}_4$ is removed from the federation since it has the lowest reputation value in the federation. Then, TCFF calculates the reputation of $\mathcal{C}_2$ and $\mathcal{C}_3$ in federation $\{\mathcal{C}_2, \mathcal{C}_3\}$. In this step, $\rho_{\mathcal{C}_2} = \rho_{\mathcal{C}_3} = 0.707107$. One of them is selected randomly to be removed. Cloud provider $\mathcal{C}_3$ could not execute the entire program, and as a result, its value is zero. Among these cloud federations, $\{\mathcal{C}_2, \mathcal{C}_3\}$ is selected to execute the program since it provides the highest individual payoff, 8.5, for $\mathcal{C}_2$ and $\mathcal{C}_3$. The average reputation of cloud federations shows that this federation also has the highest average reputation.

5 Experimental Results

We perform a set of experiments to investigate the effectiveness of the proposed trust-based federation formation mechanism in producing stable cloud federations.

5.1 Experimental Setup

For our experiments, we consider that 16 cloud providers are participating, each with a capacity of 8192 VM instances. Similar sizes for the cloud federation were considered in [36], [37]. We consider six different application program sizes ranging from 256 to 8192 jobs. The execution time of jobs were generated from uniform distributions having the average execution time extracted from performing extensive job profiling on a Hadoop cluster of 64 processors for the TeraSort (a Reduce-heavy program) program from

the HiBench benchmark workloads [38]. HiBench is a comprehensive benchmark suite for Hadoop provided by Intel to characterize the performance of MapReduce-based data analysis running in clouds and data centers. The cluster is composed of four Intel nodes, and it has a total of 80GB memory, 64 processors, 4TB of storage, and network speed of 1Gbps. We ran and profiled several TeraSort programs, and for each program, we collected their start time and finish time.

We consider a cost model for the cloud providers such that the cost functions of any two cloud providers are not related to each other. However, the cost of executing a job on a cloud provider is related to the amount of resources it requires. Meaning that, for any cloud provider $\mathcal{C}$ and two jobs J_j and J_q, if J_j requires more resources, then $c(J_j, \mathcal{C}) > c(J_q, \mathcal{C})$. As a result, a job with fewer required resources has the cheapest cost on all cloud providers. We generated the cost matrices based on the method described by Braun *et al.* [39]. We first generated a baseline vector of size 16, where each element is a random uniform number within $[1, \phi_b]$, where ϕ_b is the baseline value. We then generated the rows of the cost matrix based on the baseline vector. Each element j in row i of the matrix, $c(i, j)$, was generated by the element i of the baseline vector multiplied by a uniform random number within $[1, \phi_r]$, a row multiplier. We set the maximum value for ϕ_b and ϕ_r as 100 and 10, respectively. The choice of the deadline and payment is at the latitude of the users, we selected the values of deadlines and payments specifically to obtain feasible solutions in our experiments.

To analyze the performance of our proposed mechanism, we use two different types of random graph models for the trust graph: the Erdös-Rényi model [40], and the Barabási-Albert model [41]. These are the two major types of random graphs models that can be used to generate a wide range of graph topologies. Considering this wide range of topologies in our experiments allows us to investigate how the underlying trust graph affects the formation of federations. An Erdös-Rényi graph (m, p) is a m-node graph constructed by connecting nodes randomly, where $p \in [0, 1]$ is the probability of having an edge for any pair of nodes in the graph independent from every other edge. Based on p, the graph can be sparse or complete. In Erdös-Rényi model, all graphs with m nodes and e edges have equal probability. In our experiments, we set $m = 16$ as the number of cloud providers, and $p = 0.1$. A Barabási-Albert graph is a random scale-free graph that uses power-law (or scale-free) degree distributions. In such graphs, a node with a high degree is more likely to be connected with a new node. A Barabási-Albert graph has two parameters (m, n), where m is the number of nodes and n is the number of edges to attach from a new node to existing nodes. In our experiments, $m = 16$ is the number of cloud providers, and $n = 1$. The weights representing the direct trust assigned to the edges of the graphs were drawn from the uniform distribution on the interval $[0, 10]$. We use the ILOG Concert Technology APIs in C++ to solve IP-ASSIGN by CPLEX solver provided by IBM ILOG CPLEX Optimization Studio for Academics Initiative.

5.2 Analysis of Results

We compare the performance of our proposed trust-aware cloud federation formation mechanism (TCFF) with that of two other mechanisms: Profit-Aware Cloud Federation Formation (PCFF) and Random Cloud Federation Formation (RCFF).

The PCFF mechanism has the same structure as TCFF. However, PCFF removes a cloud provider from a federation without considering its reputation score. Meaning that, in each iteration a cloud provider is removed randomly from a federation. PCFF selects a federation with the highest individual payoff for its participating cloud providers. As a result, PCFF can be considered as a mechanism that solves the single-objective problem of maximizing the individual profit. Note that the comparison of TCFF with PCFF shows the importance of selecting the lowest reputation cloud providers for removal from the considered federations. We consider PCFF as one of the baseline algorithms to investigate and quantify the effect of neglecting trust when making a federation formation decision.

The RCFF mechanism chooses cloud providers randomly to form a federation. However, if the federation is not capable of running the job, the mechanism searches for another federation, until it finds a feasible federation. RCFF does not have any preference over the two objectives. However, it shows the formation of a random feasible federation. All mechanisms use the branch-and-bound method to solve the job assignment problem, IP-ASSIGN, in finding an assignment of the jobs to cloud providers in a federation. This allows us to focus on the federation formation and not on the choice of job assignment algorithms. We consider RCFF to investigate how well the proposed mechanism behaves against a mechanism which does not have preferences over the two objectives.

We performed a series of ten experiments for each case, and we represented the average of the obtained results. We present the results obtained by the mechanisms in terms of individual profit, size of the final federation, and trust of the participating cloud providers. We analyze the performance of the mechanisms for each of the two types of random graph models considered for the underlying trust graph, the Erdös-Rényi model and the Barabási-Albert model. In addition, we present several case studies to investigate the properties of our proposed mechanism, TCFF, in more details.

5.2.1 Erdös-Rényi Type Trust Graph

We first analyze the performance of TCFF, PCFF, and RCFF while considering the Erdös-Rényi trust graph among the cloud providers. Fig. 2 shows the normalized individual cloud provider's payoffs in the final federations obtained by the three mechanisms as a function of the number of jobs that are part of the program. Here, the payoffs are normalized relative to the highest individual payoff obtained by using TCFF. The results show that on average TCFF and PCFF mechanisms lead to almost the same amount of payoff for the cloud providers participating in the final federation. This is due to the fact that both TCFF and PCFF mechanisms select a federation that yields the highest individual payoff for the cloud providers as the final federation to execute

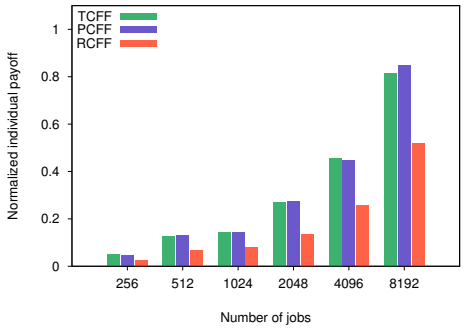

Fig. 2: Cloud provider's individual payoff (Erdös-Rényi type trust graph).

Fig. 3: Size of final federation (Erdös-Rényi type trust graph).

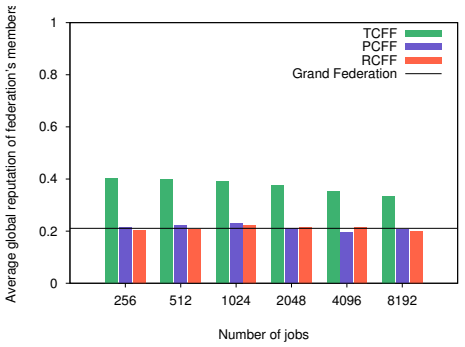

Fig. 4: Cloud provider's average reputation (Erdös-Rényi type trust graph).

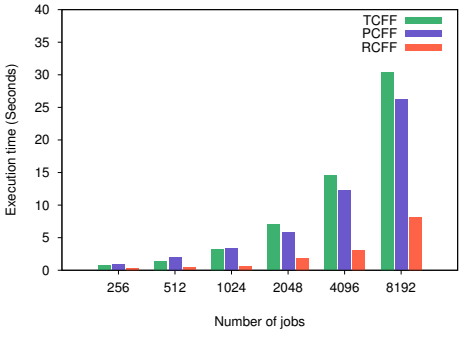

Fig. 5: Execution time of the mechanims (Erdös-Rényi type trust graph).

the application program. In addition, the results show that RCFF finds the lowest individual payoff for the cloud providers participating in the final federation among all the mechanisms in all the cases.

The number of participating cloud providers in the final federations obtained by TCFF, PCFF, and RCFF is shown in Fig. 3. The results show that the more the number of jobs, the more cloud providers participate in the final federation obtained by TCFF. This is due to the fact that more cloud providers need to pool their computing resources to form a federation in order to execute a larger program. In addition, the results show that the formed federations by TCFF do not necessarily have smaller sizes than those obtained by PCFF. However, RCFF always finds larger federations than those obtained by TCFF and PCFF. This is due to the fact that RCFF only guarantees finding a feasible federation.

Fig. 4 shows the average global reputation of the cloud providers in the final federation obtained by TCFF, PCFF, and RCFF. In this figure, we show as a horizontal line, the average global reputation in the grand federation. The obtained results show that TCFF forms federations composed of more reputable cloud providers. The average global reputation of the cloud providers in the final federations obtained by TCFF is significantly higher than the average reputation of the cloud providers in the federations obtained by PCFF, RCFF, and the grand federation, in all cases. Furthermore, by considering the individual payoff in Fig. 2 and the results of this figure, TCFF not only forms federations with the highest average reputation, but also with high individual payoff for their members. The results show the significance of the selection of cloud providers by our proposed mechanism, TCFF, which considers reputation. PCFF is a greedy mechanism that maximizes only the individual payoff without considering the reputation. As a result, PCFF obtains cloud federations with lower average reputation cloud providers than the ones obtained by TCFF.

In Fig. 5, we show the execution time of the three mechanisms, TCFF, PCFF, and RCFF, in computing the final federations. All mechanisms are very fast in finding the final federations. RCFF is the fastest since it only checks the feasibility of randomly formed federations. Note that the execution time of TCFF is reasonable given that the application program would require several hours to execute. The execution times for programs with 4096 and 8192 jobs are higher since finding job assignments for those programs takes more time.

5.2.2 *Barabási-Albert Type Trust Graph*

We analyze the performance of TCFF, PCFF, and RCFF while considering the Barabási-Albert trust graph among the cloud providers. Fig. 6 shows the normalized individual cloud provider's payoffs in the final federations obtained by the three mechanisms as a function of the number of jobs. The results show that TCFF selects the final federation yielding the highest individual profit for its participating cloud providers. In addition, RCFF finds the lowest individual profit among all the mechanisms.

The sizes of the final federations obtained by TCFF, PCFF, and RCFF are shown in Fig. 7 One of the TCFF's objective is to maximize the individual profit of the participating cloud providers. As a result, it favors the formation of smaller federations. Furthermore, RCFF finds larger fed-

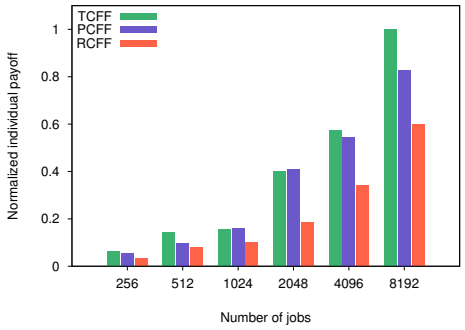

Fig. 6: Cloud provider's individual payoff (Barabási-Albert type trust graph).

Fig. 7: Size of final federation (Barabási-Albert type trust graph).

erations to guarantee that the federation is able to execute the program.

Fig. 8 shows the average global reputation of the cloud providers participating in the final federations produced by TCFF, PCFF, and RCFF. In addition, we show as a horizontal line, the average global reputation in the grand federation. The obtained results show that TCFF forms federations composed of more reputable cloud providers. The average global reputation of the cloud providers in the final federations obtained by TCFF is significantly higher in all cases than the average reputation of the cloud providers in the federations obtained by PCFF, RCFF, and the grand federation.

Fig. 9 shows the execution time of TCFF, PCFF, and RCFF in computing the final federations. All mechanisms are very fast in finding the final federations.

Comparing the results for the two types of trust graphs, we can observe that TCFF obtains a higher individual payoff and a slightly higher average global reputation for the Barabási-Albert type trust graph, than the ones obtained for the Erdös-Rényi type trust graph. This happens because the scale-free graph is more robust to the removal of a cloud provider. Thus, the trust relationships are not affected as much by the removal of a cloud provider and TCFF can still form federations with high individual payoff for the cloud providers. For both types of trust graphs, TCFF obtains the highest individual payoff and average global reputation among all the mechanisms in all cases.

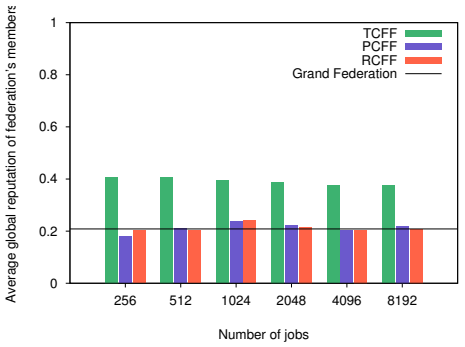

Fig. 8: Cloud provider's average reputation (Barabási-Albert type trust graph).

Fig. 9: Execution time of the mechanims (Barabási-Albert type trust graph).

Fig. 10: Cloud provider individual payoffs obtained by TCFF

5.2.3 *Case Studies*

To analyze the properties of TCFF in more details, we provide the results of several case studies. First, we consider all the programs with 512 jobs and analyze the operation of TCFF. Second, we choose two programs out of all the programs with 512 jobs and discuss the operation of the mechanisms in more details for each of them. We consider the Erdös-Rényi type trust graph for all these case studies.

We now investigate the Pareto optimality of the results obtained by TCFF. Fig. 10 shows the normalized individual payoff of the cloud providers participating in the final federation produced by TCFF. In addition, we show the normalized individual payoff of cloud providers in the federation that has the highest product of normalized individual payoff and average reputation among all federations in the list $\mathcal{L}$ maintained by TCFF (bars denoted by L-MAX in the figure).

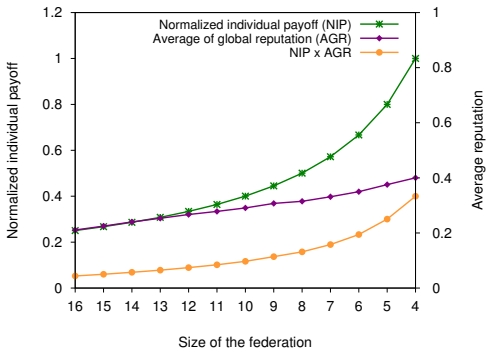

Fig. 11: Program A: Results of TCFF iterations

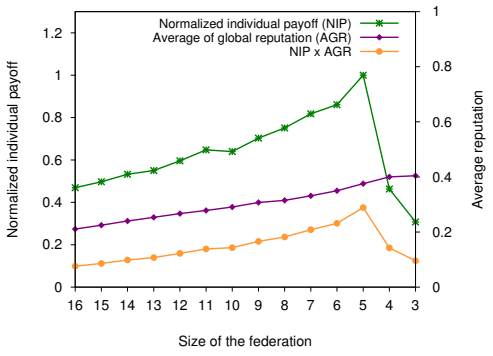

Fig. 12: Program B: Results of TCFF iterations

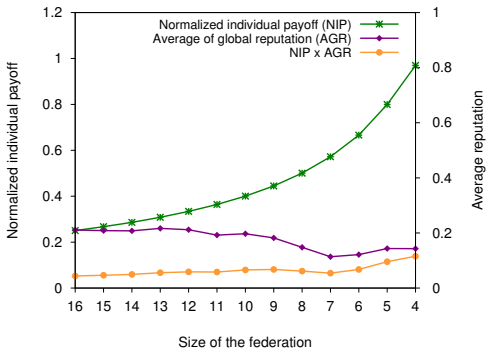

Fig. 13: Program A: Results of PCFF iterations

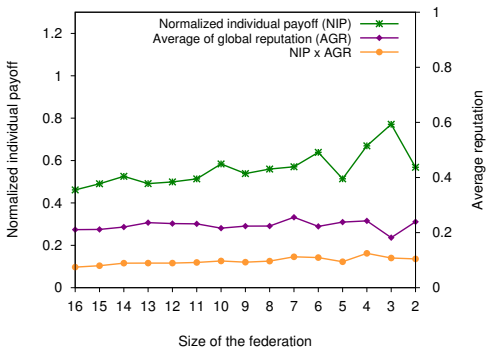

Fig. 14: Program B: Results of PCFF iterations

In doing so, we find a federation in $\mathcal{L}$ that has the highest product of normalized individual payoff and average reputation, and we present its normalized individual payoff. This figure shows that in most cases, TCFF not only finds the federation with the highest individual payoff, but also the obtained federation has the highest average reputation score leading to a Pareto optimal federation.

Figs. 11-15 show how TCFF, PCFF, and RCFF mechanisms perform for two specific programs A and B consisting of 512 jobs. Figs. 11-14 show the results of each iteration of the TCFF and PCFF mechanisms, while Fig. 15 shows only the results for the final federation obtained by RCFF (the final federation is the only feasible solution explored by RCFF). In these figures, the left vertical axis represents the normalized individual payoff of the participating cloud providers, while the right vertical axis represents the average global reputation.

We can also investigate the Pareto optimality of the obtained federations from Fig. 11 and Fig. 12. Both figures show that the federation formed by all 16 cloud providers leads to low individual payoffs for them (left vertical axis) and also to the lowest average global reputation (right vertical axis). The results show that by removing a cloud provider with the lowest reputation score (and thus reducing the size of a federation), the average global reputation values increase. Fig. 11 shows that in the case of program A, a federation with 4 cloud providers is the final federation formed by TCFF that provides the highest normalized individual payoff and the highest average global reputation of 0.4. Fig. 12 shows that in the case of program B, a federation with 5 cloud providers is the final federation formed by TCFF that provides the highest normalized individual payoff and an average global reputation of 0.37. It is worth noting that the final federation (shown in Fig. 12) does not have the highest average global reputation, but it gives the highest product of normalized individual payoff and average global reputation. In addition, the results show that the normalized individual payoff of the cloud providers decreases when the federation size reduces from 5 to 4. This is due to the fact that in each iteration, TCFF removes a cloud provider with the lowest reputation. A cloud provider with the lowest reputation may have a lower cost compared to other cloud providers in the federation. As a result, removing such a cloud provider increases the total cost of the program execution which in turn leads to decrease in the normalized individual payoff of the remaining cloud providers.

The formation of federations obtained by PCFF for the same programs A and B is shown in Fig. 13 and Fig. 14, respectively. Both figures show that the average global reputation changes in all iterations, but it does not increase. This is due to the fact that cloud providers are removed randomly. PCFF for program A selects a federation with 4 cloud providers as the final federation with normalized individual payoff of 0.96 and average global reputation of 0.14. PCFF for program B selects a federation with 3 cloud provider as the final federation with normalized individual payoff of 0.77 and average global reputation of 0.18. The results show that selecting a federation with the highest normalized individual payoff does not provide the highest product of normalized individual payoff and average global reputation. The objective of PCFF is to maximize the individual payoff without considering the reputation of the cloud providers. As shown in Fig 13 and Fig. 14, the average

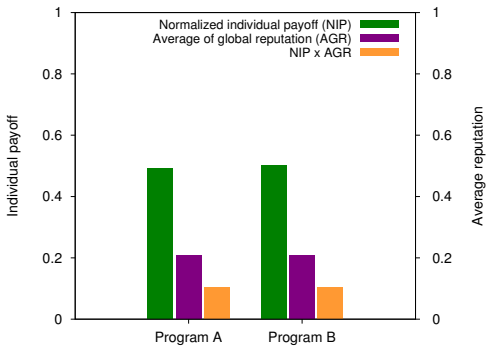

Fig. 15: Program A and B: Results of RCFF

global reputation does not exhibit any trend as the size of the federation decreases. The key observation is that no federation obtained by PCFF has both the average reputation and the individual payoff higher than those obtained by the federation selected by TCFF.

Fig. 15 shows the individual payoff and the average reputation for the federation determined by RCFF considering the same programs, A and B. For program A, RCFF selects a federation with 13 cloud providers as the final federation with normalized individual payoff of 0.49 and average global reputation of 0.20. Note that the individual payoffs are normalized for each application irrespectively. For program B, RCFF selects a federation with 13 cloud providers as the final federation with normalized individual payoff of 0.50 and average global reputation of 0.20. The federations obtained by RCFF, for both programs, do not have both higher average reputation and higher individual payoff than those selected by TCFF.

From the above results, we conclude that our proposed trust-aware cloud federation formation mechanism, TCFF, is able to form stable and Pareto optimal federations guaranteeing highest profit and reliability while ensuring service level agreement.

6 Conclusion

We proposed a novel mechanism for cloud federation formation considering the trust relationships among cloud providers when making decisions. Trust relationships are key factors influencing the formation of cloud federations. In some cases, a cloud provider agrees to provide some resources, but it fails to deliver the promised resources to a federation. As a result, the application program could not be executed by that federation. In our proposed mechanism, TCFF, cloud providers cooperate to form federations with high reputation cloud providers in order to execute large-scale Map-heavy/Reduce-heavy programs. We proposed a trust-aware cloud federation formation game to model the problem, and we derived our cloud federation formation mechanism, TCFF, to solve this bicriteria optimization problem. We performed extensive experiments to investigate the properties of TCFF. Experimental results showed that the formed federation obtained by TCFF maximizes the reputation of its participating cloud providers. Moreover, the formed federation produced by TCFF has the highest individual payoffs for its participating cloud providers in most of the times. Our proposed mechanism determines a stable federation with Pareto optimal solutions in terms of reputation and individual payoff. The execution time of TCFF is reasonable given that large-scale applications programs would require several hours to execute, making the mechanism suitable for deployment on clouds. We believe that this research will encourage cloud service providers to adopt trust-based federation formation mechanisms and use them to pool their resources together in order to execute large-scale application programs. Like any other new technology, reaching the critical mass so that enough number of providers participate in the mechanism can be one challenge. This can be overcame by the incentives derived from using the mechanism. Moreover, currently cloud providers do not offer VMs that are compatible with the VMs offered by other cloud providers, not allowing the users to change cloud providers smoothly. Creating standardized services among all providers that are willing to provide federated resources is also another challenge. In future work, we would like to study data privacy in cloud federation formation by designing new matching techniques to avoid co-locating VMs that could potentially be prone to information leakage. Furthermore, we plan to design cloud federation formation mechanisms that take into account the geographical distribution of cloud providers when making federation formation decisions.

Acknowledgment

This research was supported in part by NSF grant CNS-1755913.

References

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *Proc. of the 6th USENIX Symposium on Operating System Design and Implementation*, 2004, pp. 137–150.

[2] L. Mashayekhy, M. Nejad, D. Grosu, Q. Zhang, and W. Shi, "Energy-aware scheduling of mapreduce jobs for big data applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2720–2733, 2015.

[3] Y. Wang, J. Tan, W. Yu, L. Zhang, X. Meng, and X. Li, "Preemptive reducetask scheduling for fair and fast job completion," in *Proc. of the 10th Intl. Conf. on Autonomic Computing*, 2013, pp. 279–289.

[4] L. Mashayekhy, M. Nejad, and D. Grosu, "A PTAS mechanism for provisioning and allocation of heterogeneous cloud resources," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 9, pp. 2386–2399, 2015.

[5] L. Mashayekhy, M. M. Nejad, and D. Grosu, "Physical machine resource management in clouds: A mechanism design approach," *IEEE Trans. on Cloud Computing*, vol. 3, no. 3, pp. 247–260, 2015.

[6] L. Mashayekhy, M. M. Nejad, D. Grosu, and A. V. Vasilakos, "An online mechanism for resource allocation and pricing in clouds," *IEEE Transactions on Computers*, vol. 65, no. 4, pp. 1172–1184, 2016.

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

[8] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, and M. Kunze, "Cloud federation," in *Proc. of the 2nd Intl. Conf. on on Cloud Computing, Grids, and Virtualization*, 2011, pp. 32–38.

[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] L. Mashayekhy, M. M. Nejad, and D. Grosu, "Cloud federations in the sky: Formation game and mechanism," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 14–27, 2015.

[11] N. Samaan, "A novel economic sharing model in a federation of selfish cloud providers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 12–21, 2014.

[12] D. Bruneo, "A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 560–569, 2014.
[13] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *Proc. of the 2nd USENIX Conf. on Hot Topics in Cloud Computing*, 2010, pp. 1–10.
[14] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous environments," in *Proc. of the 8th USENIX Conf. on Operating Systems Design and Implementation*, 2008, pp. 29–42.
[15] H. Chang, M. S. Kodialam, R. R. Kompella, T. V. Lakshman, M. Lee, and S. Mukherjee, "Scheduling in mapreduce-like systems for fast completion time," in *Proc. of the 30th IEEE International Conference on Computer Communications*, 2011, pp. 3074–3082.
[16] J. Zhan, L. Wang, X. Li, W. Shi, C. Weng, W. Zhang, and X. Zang, "Cost-aware cooperative resource provisioning for heterogeneous workloads in data centers," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2155–2168, 2013.
[17] C. Hang, Y. Wang, and M. Singh, "Operators for propagating trust and their evaluation in social networks," in *Proc. 8th Int. Conf. Autonomous Agents and Multiagent Syst. Vol. 2*, 2009, pp. 1025–1032.
[18] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *Proc. of the 13th Intl. Conf. on World Wide Web*, 2004, pp. 403–412.
[19] D. Agrawal, H. Chivers, J. Clark, C. Jutla, and J. McDermid, "A proposal for trust management in coalition environments," IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 2008.
[20] K. Avrachenkov, D. Nemirovsky, and K. Pham, "A survey on distributed approaches to graph based reputation measures," in *Proc. of the 2nd Intl. Conf. on Performance Evaluation Methodologies and Tools*, 2007, p. 82.
[21] M. Rahman, R. Ranjan, and R. Buyya, "Reputation-based dependable scheduling of workflow applications in peer-to-peer grids," *Computer Networks*, vol. 54, no. 18, pp. 3341–3359, 2010.
[22] L. Mashayekhy and D. Grosu, "A reputation-based mechanism for dynamic virtual organization formation in grids," in *Proc. 41st IEEE Intl. Conf. on Parallel Processing*, 2012, pp. 108–117.
[23] R. K. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B. S. Lee, "Trustcloud: A framework for accountability and trust in cloud computing," in *Proc. of the IEEE World Congress on Services*, 2011, pp. 584–588.
[24] M. M. Hassan, M. Abdullah-Al-Wadud, A. Almogren, S. Rahman, A. Alelaiwi, A. Alamri, M. Hamid *et al.*, "QoS and trust-aware coalition formation game in data-intensive cloud federations," *Concurrency and Computation: Practice and Experience*, 2015.
[25] R. Shaikh and M. Sasikumar, "Trust model for measuring security strength of cloud computing service," *Procedia Computer Science*, vol. 45, pp. 380–389, 2015.
[26] K. Gokulnath and R. Uthariaraj, "Game theory based trust model for cloud environment," *The Scientific World Journal*, vol. 2015, 2015.
[27] F. Messina, G. Pappalardo, D. Rosaci, C. Santoro, and G. M. Sarné, "A trust model for competitive cloud federations," in *Proc. of the 8th IEEE Intl. Conf. on Complex, Intelligent and Software Intensive Systems*, 2014, pp. 469–474.
[28] U. Premarathne, I. Khalil, Z. Tari, and A. Zomaya, "Cloud-based utility service framework fortrust negotiations using federated identitymanagement," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, 2015.
[29] L. Mashayekhy, M. M. Nejad, and D. Grosu, "A framework for data protection in cloud federations," in *Proc. of the 43rd International Conference on Parallel Processing*, 2014, pp. 283–290.
[30] G. Golub and C. Van Loan, *Matrix computations*. Johns Hopkins Univ. Press, 1996, vol. 3.
[31] P. Bonacich, "Some unique properties of eigenvector centrality," *Social Networks*, vol. 29, no. 4, pp. 555–564, 2007.
[32] L. S. Shapley, "Cores of convex games," *International Journal of Game Theory*, vol. 1, no. 1, pp. 11–26, 1971.
[33] V. Conitzer and T. Sandholm, "Computing shapley values, manipulating value division schemes, and checking core membership in multi-issue domains," in *Proc. of the 19th National Conference on Artifical Intelligence*, 2004, pp. 219–225.
[34] T. Javidi, "Cooperative and non-cooperative resource sharing in networks: A delay perspective," *IEEE Transactions on Automatic Control*, vol. 53, no. 9, pp. 2134–2142, 2008.
[35] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*.
[36] J. Xu and B. Palanisamy, "Cost-aware resource management for federated clouds using resource sharing contracts," in *Proc. of the IEEE 10th Intl. Conf. on Cloud Computing*, 2017, pp. 238–245.
[37] T. Gouasmi, W. Louati, and A. H. Kacem, "Geo-distributed bigdata processing for maximizing profit in federated clouds environment," in *Proc. of the IEEE 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, 2018, pp. 85–92.
[38] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, "The HiBench benchmark suite: Characterization of the MapReduce-based data analysis," in *Proc. of the IEEE 26th Conf. on Data Engineering Workshops*, 2010, pp. 41–51.
[39] T. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran *et al.*, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *J. of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810–837, 2001.
[40] P. Erdös and A. Rényi, "On random graphs, I," *Publicationes Mathematicae (Debrecen)*, vol. 6, pp. 290–297, 1959.
[41] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

BIOGRAPHIES

Lena Mashayekhy is an assistant professor in the Department of Computer and Information Sciences at the University of Delaware. She received her PhD degree in computer science from Wayne State University in 2015. Her research interests include edge and cloud computing, data-intensive computing, distributed systems and parallel algorithms, algorithmic game theory, and electric vehicles. Her doctoral dissertation received the 2016 IEEE TCSC Outstanding PhD Dissertation Award. She is also a recipient of the 2017 IEEE TCSC Award for Excellence in Scalable Computing for Early Career Researchers. She has published more than thirty peer-reviewed papers in venues such as IEEE Transactions on Cloud Computing and IEEE Transactions on Parallel and Distributed Systems. She is a member of the IEEE, ACM, and INFORMS.

Mark M. Nejad is an assistant professor in the Department of Civil and Environmental Engineering at the University of Delaware. His research interests include network optimization and control, cloud computing, game theory, and connected vehicles. He received the 2016 Institute of Industrial and Systems Engineers (IISE) Pritsker Best Doctoral Dissertation Award. He has published more than thirty peer-reviewed papers in venues such as IEEE Transactions on Cloud Computing and IEEE Transactions on Computers. He is a member of the IEEE, IISE, and 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 computing, approximation algorithms, computer security, and topics at the border of computer science, game theory and economics. He has published more than one hundred 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 such as ICDCS, CLOUD, ICPP and NetEcon. He is a senior member of the ACM, the IEEE, and the IEEE Computer Society.