A Reputation-Based Mechanism for Dynamic Virtual Organization Formation in Grids

Lena Mashayekhy and Daniel Grosu
Department of Computer Science
Wayne State University
Detroit, MI 48202, USA
Email: {mlena, dgrosu}@wayne.edu

***Abstract*—In order to execute large scale applications programs in grids, several Grid Service Providers (GSPs) pool their resources together by forming Virtual Organizations (VOs). Forming such VOs is a challenging problem especially when the trust relationships among GSPs have to be considered. In this paper, we model the formation of VOs in grids by considering the trust and reputation of the participating GSPs. We design a mechanism for VO formation that enables the GSPs with high reputation to organize into a VO reducing the cost of execution and guaranteeing the maximum profit for the participating GSPs. Furthermore, the mechanism guarantees that the formed VO is stable, that is, the GSPs that are part of the VO do not have incentives to break away from it. We perform extensive simulation experiments using real workload traces to characterize the properties of the proposed mechanism. The results show that the mechanism produces stable VOs composed of GSPs with high reputation that obtain high individual profits.**

***Keywords*-grid computing; VO formation; reputation**

I. Introduction

Grid computing systems are composed of heterogeneous resources (CPUs, storage, etc.) owned by autonomous organizations. These systems provide essential resources for conducting cutting-edge science and engineering research. The resource management in such open distributed environments is a very complex problem. Efficient resource management in grids leads to efficient utilization of resources and faster execution of applications. One important aspect of resource management in grids is how Grid Service Providers (GSPs) pool their resources together to execute large scale applications. These GSPs collaborate and form Virtual Organizations (VOs). A key element in the formation of Virtual Organizations is the GSPs' reliability of executing the requested program. In some cases, a GSP agrees to provide some resources, but it fails to deliver the promised resources to a VO. As a result, the application program could not be executed by that VO. Selecting highly trusted GSPs to be part of the VO may avoid this problem. Therefore, considering the trust among GSPs based on their previous behaviour would avoid these problems in the VO formation. GSPs desire to build VOs with the most trusted GSPs and obtain high profits. In addition, if a GSP does not have any past interactions with another GSP, it can use its reputation in the network, that is, the reputation based on the opinions of the other GSPs that have direct trust to that GSP.

In this paper, we define trust as how likely is a GSP to provide the requested resources to another GSP. A GSP assigns a trust value to another GSP based on their past interactions. Trust is based on direct interactions of GSPs. However, some of the GSPs may not have had direct interactions in the past. A GSP that did not have past interactions with another GSP may use the reputation of that GSP to evaluate how likely it is to provide the requested resources. We define reputation as the opinion of other GSPs that have had direct interactions with both GSPs to evaluate how likely they are to provide the requested resources. Also, for a GSP, we define its global reputation as how likely is this GSP to provide the requested resources based on the opinion of all GSPs. We propose a mechanism for VO formation that takes into account the reputation of each of the GSPs that form the VO. We also describe a framework for calculating the global reputation for each GSP considering direct trust and reputation.

A. Related Work

There are several studies on how to measure reputation in different domains. These studies model different problems using graphs in which the weight associated with each edge represents the value of trust. One approach makes use of trust propagation to find the reputation [1]. If two nodes are not adjacent, each node can evaluate the value of trust to another one using an existing path between them based on the trust transitivity property. That is, if A trusts B and B trusts C, then A trusts C to some extent. Hang *et al.* [1] defined three operators, aggregation, concatenation, and selection, in order to improve the accuracy of trust propagation. They also considered the selection of the path with the highest propagated trust value in cases in which multiple paths exist between two participants. A social trust inference algorithm was designed by Kuter and Golbeck [2] to estimate the confidence in the trust. This algorithm is based on probabilistic reasoning where the confidence is calculated based on the preference similarity. Confidence and trust values are propagated over the entire network. Another approach proposed by Agrawal *et al.* [3] is based on the use of the network flow to find the reputation. Many reputation systems are designed based on graph centrality measures [4]. These studies focused on the centrality of nodes within a graph. The centrality of a node determines the reputation of the node among all nodes. Various centrality metrics were defined such as degree centrality, betweenness centrality, closeness centrality, and eigenvector centrality [5], [6], [7], [8].

Trust is one major concern when establishing sharing relationships among the GSPs in a grid system. Azzedin and Maheswaran [9] proposed a trust model for grid systems that considers the trust between GSPs and users. Their model assumed that the trust and the reputation decay with time. The amount of trust between two participants is a weighted sum of the direct trust and reputation. They used trust in three heuristic mapping algorithms: minimum completion time, Min-min, and Sufferage. The simulation results showed improvement in the overall quality of the schedules in terms of utilization and average completion time. However, the assumption of decaying trust and reputation with time limits the applications of this method in grids. This method converges to a state in which the formation of new VOs is not possible. GSPs form VOs and as a result would tend to just trust the members of their respective VOs. Lin and Huai [10] proposed a method in which a GSP decides to allocate resources based on the combination of the trust value and the bidding price of a requester. Their proposed method, QGrid, is based on Q-learning techniques that balance the relative importance of trust and price. QGrid is a distributed method for computing the reputation.

The combination of reputation and global trust was used to build a grid reputation management framework called GridEigenTrust [11], [12]. Reputation in GridEigenTrust is determined using the eigentrust algorithm proposed by Kamvar *et al.* [13], while the global trust is computed using the method proposed by Azzedin and Maheswaran [9]. In GridEigenTrust, a GSP selects trusted resources and GSPs to satisfy the requirements of the application based on a hierarchical process. Each organization has a set of entities: resources, GSPs, and users. A VO is a set of organizations or some parts of organizations (i.e., subset of entities in an organization). A hierarchy consists of entities, organizations, and VOs. A reputation is assigned to each entity. The reputation of the organization is computed based on the reputation of the entities that are part of the organization. Finally, the reputation of a VO is computed based on the reputations of its component organizations. However, the VO formation problem was not considered by the authors. To the best of our knowledge, our paper is the first to take into account the global trust of the GSPs in the VO formation process.

B. Our Contribution

We address the problem of VO formation in grids considering the trust relationships among GSPs. We design a mechanism that allows the GSPs to make their own decisions to participate in VOs. The mechanism provides a stable VO, that is, none of the GSPs has incentives to leave the VO and to collaborate with other GSPs outside the current VO. The mechanism determines the mapping of the tasks to each of the VOs that minimizes the cost of execution by using a branch-and-bound method. As a result, in each step of the mechanism the mapping provides the maximum individual payoffs for the participating GSPs. We analyze the properties of our proposed VO formation mechanism and perform extensive simulation experiments using real workload traces from the Parallel Workloads Archive [14]. The results show that the proposed mechanism determines a stable VO that not only guarantees the highest reputation among its participating GSPs, but also maximizes the individual payoffs of its members.

C. Organization

The rest of the paper is organized as follows. In Section II, we describe the system model, the VO trust model, and the VO formation framework. In Section III, we present the proposed mechanism and characterize its properties. In Section IV, we evaluate the mechanism by extensive simulation experiments. In Section V, we summarize our results and present possible directions for future research.

II. VO Formation Framework

In this section, we describe the model of the system, the VO trust model, and the VO formation framework.

A. System Model

We first describe the system model which considers that a user wants to execute a large-scale application program $\mathcal{T}$ consisting of n independent tasks $\{T_1, T_2, \ldots, T_n\}$ on the available set of grid service providers (GSPs) by a given deadline d. Application programs consisting of several independent tasks are representative for a wide range of problems in science and engineering [15], [16], [17]. Each task $T \in \mathcal{T}$ composing the application program is characterized by its workload $w(T)$, which can be defined as the amount of floating-point operations required to execute the task. Executing the application program $\mathcal{T}$ requires a large number of resources which cannot be provided by a single GSP. Thus, several GSPs pool their resources together to execute the application. We consider that a set of m GSPs, $\mathcal{G} = \{G_1, G_2, \ldots, G_m\}$, are available and are willing to provide resources for executing programs. Here, we assume that the GSPs are driven by incentives in the sense that they will execute a task only if they make some profit out of it. More specifically, the GSPs are assumed to be self-interested and welfare-maximizing entities. Each service provider $G \in \mathcal{G}$ owns several computational resources which are abstracted as a single machine with speed $s(G)$. The speed $s(G)$ gives the number of floating-point operations per second that can be executed by GSP G. Therefore, the execution time of task T at GSP G is given by the execution time function $t : \mathcal{T} \times \mathcal{G} \rightarrow \mathbb{R}^+$, where $t(T, G) = \frac{w(T)}{s(G)}$. We also assume that once a task is assigned to a GSP, the task is neither preempted nor migrated.

A GSP incurs cost for executing a task. The cost incurred by GSP $G \in \mathcal{G}$ when executing task $T \in \mathcal{T}$ is given by the cost function, $c : \mathcal{T} \times \mathcal{G} \rightarrow \mathbb{R}^+$. Furthermore we assume that a GSP has zero fixed costs and its variable costs are given by the function c. A user is willing to pay a price P less than her available budget B if the program is executed to completion by deadline d. If the program execution exceeds d, the user is not willing to pay any amount that is, $P = 0$.

Since a single GSP does not have the required resources for executing the program, GSPs form VOs in order to have the necessary resources to execute the program and more importantly, to maximize their profits. The profit is simply defined as the difference between the payment received by a GSP and its execution costs. If the profit is negative (*i.e.*, a loss), the GSP will choose not to participate.

B. VO Trust Model

GSPs desire to form VOs with the most trusted GSPs. A GSP assigns a trust value to another GSP based on their interactions. We model the trust relationship among GSPs as a weighted directed graph $(\mathcal{G}, E)$, where $\mathcal{G}$ is a set of GSPs that represents the vertices in the graph and E is a set of edges (i, j). The weight u_{ij} associated with edge (i, j) represents the amount of trust that G_i assigns to G_j, where $G_i, G_j \in \mathcal{G}$. The weight u_{ij} is the strength of the trust relationship from G_i to G_j which is based on past interactions among them. Trust can be an asymmetric relationship. If $u_{ij} = 0$ then G_i distrusts G_j completely. This can happen if they did not have any interactions in the past or G_j did not provide the requested resources to G_i in past interactions. Direct trust is based on past interactions between two GSPs, but if the two GSPs did not have any interactions in the past, they can rely on the observations of the other GSPs.

To form a VO, G_i should be able to select G_j based not only on their direct trust u_{ij} but also on the trust the other GSPs have on G_j. As a result, we need to consider the reputation of GSPs rather than their direct trust. To do so, we need to define a metric that characterizes the reputation of each GSP. This metric measures how likely is the GSP to provide the requested resources based on all other GSPs' opinions.

Direct trust can be used for local ratings. That means that a GSP rates another GSP based on their direct trust. To assign a single rating to a GSP, i.e., the local trust value, the normalization of the direct trust is used in such a way that the values of local trust are between 0 and 1. We define $a_{ij} \in [0, 1]$ to be the *normalized trust* that G_i assigns to G_j. In addition, for each $i = 1, \cdots, m$, $\sum_{j=1}^{m} a_{ij} = 1$, where m is the number of GSPs. Each GSP computes the normalized trust values by dividing the local trust u_{ij} by the sum of the local trust values assigned to all its neighbor GSPs as follows:

$$a_{ij} = \frac{u_{ij}}{\sum_{k \in N_i} u_{ik}} \tag{1}$$

where, $N_i = \{G_j | \exists (i, j) \in E\}$ is the set of G_i's neighbors.

In the following we describe a procedure for determining the global reputation of each GSP within a given set $\mathcal{G}$ of GSPs. This procedure is called the *power method* [18]. We denote by $x^q_{G_i \to G_j}$ the trust G_i assigns to G_j based on the opinion of q GSPs. The procedure start by determining $x^0_{G_i \to G_j}$ the local trust G_i assigns to G_j as follows:

$$x^0_{G_i \to G_j} = a_{ij} \tag{2}$$

Let A be the matrix of normalized trust of the graph $(\mathcal{G}, E)$, where $a_{ij} \in [0, 1]$ represents the *normalized trust values*. To find the reputation between GSPs G_i and G_j, G_i uses its neighbors opinions about G_j by weighting their opinions using the trust G_i places on them:

$$x^1_{G_i \to G_j} = \sum_{G_k \in \mathcal{G}} (a_{kj})^T \cdot x^0_{G_i \to G_k} \tag{3}$$

This method aggregates the local trust values of all GSPs and computes the reputation of GSPs using the transitive property of the trust. As a result, the power method 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. This procedure can be done for the neighbors of neighbors, and so on. This improves the accuracy of trust propagation.

$$x^q_{G_i \to G_j} = \sum_{G_k \in \mathcal{G}} (a_{kj})^T \cdot x^{q-1}_{G_i \to G_k} \tag{4}$$

Let $\mathbf{x}^q_{G_i}$ denote the vector that contains all the reputation scores that G_i assigns to the other GSPs using q GSPs. In other words, the length of a path from G_i to other GSPs in the graph is q. Using the matrix notation, equation (4) for all GSPs G_i, $i = 1, \ldots, m$ can be written as follows:

$$\mathbf{x}^q_{G_i} = (A^T) \cdot \mathbf{x}^{q-1}_{G_i} \tag{5}$$

If q is large, G_i will assign a reputation score to each GSP considering the opinion of all GSPs. In addition, if all other GSPs do the same to find the reputation scores of all GSPs, they will find the same reputation scores as in $\mathbf{x}^q_{G_i}$. As a result, $\mathbf{x}^q_{G_i}$ converges to the *global reputation* of the GSPs (i.e., the global reputation vector $\mathbf{x}$). This vector is the left principal eigenvector of A, that is, it satisfies:

$$\lambda \mathbf{x} = (A^T) \cdot \mathbf{x} \tag{6}$$

where λ is the eigenvalue of A. As a result, the procedure determines the global reputation of each GSP. By using this method, we convert the trust values between each pair of GSPs into a global reputation for each GSP. The i-th component x_i of the eigenvector $\mathbf{x}$ then gives the global reputation score of G_i. Using this method, a GSP has high reputation to the extent that the GSP is connected to others who have high reputation [19], [20]. Here, the eigenvector $\mathbf{x}$ determines the *centrality* of the GSPs based on their reputation.

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

$$\bar{x}(\mathcal{G}) = \frac{1}{|\mathcal{G}|} \sum_{i: G_i \in \mathcal{G}} x_i \tag{7}$$

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

C. VO Formation Model

We model the VO formation problem as a coalitional game. A *coalitional game* [21] is defined by the pair $(\mathcal{G}, v)$, where $\mathcal{G}$ is the set of players (in our case GSPs) and v is a real-valued function called the *characteristic function*, defined on $\mathcal{C} \subseteq \mathcal{G}$ such that $v : \mathcal{C} \rightarrow \mathbb{R}^+$ and $v(\emptyset) = 0$. In our model, the players are the GSPs that form VOs which are coalitions of GSPs. In this work, we use the terms VO and coalition interchangeably.

Each subset $\mathcal{C} \subseteq \mathcal{G}$ is a *coalition*. If all the players form a coalition, it is called the *grand coalition*. A coalition has a *value* given by the characteristic function $v(\mathcal{C})$ representing the profit obtained when the members of a coalition work as a group. To maximize the value of a VO, a GSP prefers to join a VO with higher value and members of a VO prefer to join with GSPs that have higher reputation scores. As a result, the formation of a VO not only depends on profit, but it also depends on how much trust the GSPs that are part of the VO have on each other. The reputation of GSPs in a VO means how much reputation each GSP has based on the opinions of all GSPs in that VO. The trust graph among the GSPs in a VO has an impact on the formation of the VO. We define a subgraph $(\mathcal{C}, \mathcal{E})$ of $(\mathcal{G}, E)$, where $\mathcal{C}$ is the set of GSPs in the VO and $\mathcal{E}$ is a set of edges among GSPs in $\mathcal{C}$. We denote by $A_{\mathcal{C}}$ the matrix containing the trust values of the GSPs in $\mathcal{C}$. We define the reputation in a VO $\mathcal{C}$ using the subgraph $\mathcal{S}$. We model the VO formation based on reputation as a coalition formation problem. *Coalition formation* [22] is the partitioning of the players into disjoint sets. A coalition structure $\mathcal{CS} = \{S_1, S_2, \ldots, S_h\}$ forms a partition such that each player is a member of exactly one coalition, *i.e.*, $S_i \cap S_j = \emptyset$ for all i and j where $i \neq j$ and $\bigcup_{S_i \in \mathcal{CS}} S_i = \mathcal{G}$. In our proposed VO formation game only one of the coalitions in the coalition structure is selected to execute the application program, thus, the formation of the rest of the coalitions is not important. The reason for that is the rest of the GSPs which are not in the final coalition can participate again in another coalition formation process for executing another application program.

For each VO composed of GSPs from $\mathcal{G}$, there exists a mapping $\pi_{\mathcal{C}} : \mathcal{T} \rightarrow \mathcal{C}$, which assigns task $T \in \mathcal{T}$ to GSP $G \in \mathcal{C}$. To make sure that a VO is able to execute the program $\mathcal{T}$, we need to find a mapping of all the tasks on the members of the VO in such a way that the mapping satisfies all constraints. This problem is known as the task assignment problem.

The task assignment problem finds a mapping of the n tasks of the application to k GSPs in VO $\mathcal{C}$ where $k = |\mathcal{C}|$. We consider the following decision variables:

$$\sigma_{\mathcal{C}}(T, G) = \begin{cases} 1 & \text{if } \pi_{\mathcal{C}}(T) = G, \\ 0 & \text{if } \pi_{\mathcal{C}}(T) \neq G. \end{cases} \tag{8}$$

We formulate the task assignment problem as an integer program (IP) as follows:

$$\text{Minimize } C(\mathcal{T}, \mathcal{C}) = \sum_{T \in \mathcal{T}} \sum_{G \in \mathcal{C}} \sigma_{\mathcal{C}}(T, G) c(T, G), \tag{9}$$

Subject to:

$$\sum_{T \in \mathcal{T}} \sum_{G \in \mathcal{C}} \sigma_{\mathcal{C}}(T, G) c(T, G) \leq P, \ (\forall G \in \mathcal{C} \text{ and } \forall T \in \mathcal{T}), \tag{10}$$

$$\sum_{T \in \mathcal{T}} \sigma_{\mathcal{C}}(T, G) t(T, G) \leq d, \ (\forall G \in \mathcal{C}), \tag{11}$$

$$\sum_{G \in \mathcal{C}} \sigma_{\mathcal{C}}(T, G) = 1, \ (\forall T \in \mathcal{T}), \tag{12}$$

$$\sum_{T \in \mathcal{T}} \sigma_{\mathcal{C}}(T, G) \geq 1, \ (\forall G \in \mathcal{C}), \tag{13}$$

$$\sigma_{\mathcal{C}}(T, G) \in \{0, 1\}, \ (\forall G \in \mathcal{C} \text{ and } \forall T \in \mathcal{T}). \tag{14}$$

The objective function (9) represents the costs incurred for executing the program $\mathcal{T}$ on $\mathcal{C}$ under the mapping. Constraints (10) ensure that the sum of the cost of execution the program $\mathcal{T}$ on $\mathcal{C}$ under the mapping is less than or equal to the payment. Constraints (11) ensure that each GSP can execute its assigned tasks by the deadline. Constraints (12) guarantee that each task $T \in \mathcal{T}$ is assigned to exactly one GSP. Constraints (13) ensure that each GSP $G \in \mathcal{C}$ is assigned at least one task. Constraints (14) represent the integrality requirements for the decision variables.

We define the following characteristic function for our proposed VO formation game:

$$v(\mathcal{C}) = \begin{cases} 0 & \text{if } |\mathcal{C}| = 0 \text{ or IP is not feasible,} \\ P - C(\mathcal{T}, S) & \text{if } |\mathcal{C}| > 0 \text{ and IP is feasible,} \end{cases} \tag{15}$$

where $|\mathcal{C}|$ is the cardinality of $\mathcal{C}$, and that $v(\mathcal{C})$ satisfies the constraint $v(\emptyset) = 0$.

The trust among GSPs in a VO is an important factor in the formation of a VO. The objective is to find the VO $\mathcal{C}$ such that its members have the highest average global reputation (as defined in Section II-B) and the VO provides the maximum individual profit for its members.

There are different ways to divide the profit earned by coalition $\mathcal{C}$ among its members. Traditionally, the *Shapley value* [23] would be employed, but computing the Shapley value requires iterating over every partition of a coalition, an exponential time endeavor. 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 successfully used as an allocation rule in other systems where tractability is critical (*e.g.*, [24]). For this reason we adopt here the equal sharing of the profit as the payoff division rule. A VO divides the profit equally among its members.

Due to their welfare-maximizing behavior, the GSPs prefer to form a low profit coalition if their profit divisions are higher than those obtained by participating in a high profit coalition. Also, the GSPs will prefer a VO with the highest average

reputation for its members. Therefore, a GSP G determines its preferred VO $\mathcal{C}$, where $G \in \mathcal{C}$, by solving:

$$\max_{(\mathcal{C})} \frac{P - C(\mathcal{T}, \mathcal{C})}{|\mathcal{C}|} \tag{16}$$

and

$$\max_{(\mathcal{C})} \frac{\sum_{i:G_i \in \mathcal{C}} x_i}{|\mathcal{C}|} \tag{17}$$

where x_i is the reputation score of G_i as computed by the power method presented in Section II-B.

This is a bicriteria optimization problem in which the GSP goal is to maximize the profit share that it obtains from the VO and at the same time maximize the average global trust it has within the VO. Minimizing the cost $C(\mathcal{T}, \mathcal{C})$ by solving the IP problem implicitly maximizes the profit, $P - C(\mathcal{T}, \mathcal{C})$, earned by a VO. That means, a VO finds the maximum profit, then the profit is divided among participating GSPs. As a result, a GSP prefers a VO that provides the highest profit among all possible VOs.

Since a GSP has to solve a bicriteria optimization problem (defined by equations (16) and (17)) we will asses the optimality of the solutions using the concept of *Pareto optimality*. In our case Pareto optimality refers to the set of solutions of the bicriteria problem defined above that are not dominated by other solutions in both criteria. Here the two criteria are the individual payoff and the average reputation. Thus, a solution $\mathcal{C}$ yielding an individual payoff of π and an average reputation of $\bar{x}$ is Pareto optimal if there is no other solution (in our case VO) $\mathcal{C}'$ with both a higher payoff π' and a higher average reputation $\bar{x}'$ (i.e., $\pi' \geq \pi$ and $\bar{x}' \geq \bar{x}$). 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.

The *payoff* or the *share* of GSP G part of coalition $\mathcal{C}$, denoted by $\psi_G(\mathcal{C})$ is given by

$$\psi_G(\mathcal{C}) = \frac{P - C(\mathcal{T}, \mathcal{C})}{|\mathcal{C}|}. \tag{18}$$

Thus, the payoff vector $\psi(\mathcal{G}) = (\psi_{G_1}(\mathcal{G}), \cdots, \psi_{G_m}(\mathcal{G}))$ gives the payoff divisions of the grand coalition. A solution concept for coalitional games is a payoff vector that allocates the payoff among the players in some fair way. The primary concern for any coalitional game is stability. One of the solution concepts used to asses the stability of coalitions is the *core*. In order to define the core we need to introduce first the concept of imputation. An *imputation* is a payoff vector such that $\psi_G(\mathcal{G}) \geq v(G)$ for all GSPs $G \in \mathcal{G}$, and $\sum_{G \in \mathcal{G}} \psi_G(\mathcal{G}) = v(\mathcal{G})$. The first condition says that by forming the grand coalition the profit obtained by each member G participating in the grand coalition is not less than the one obtained when acting alone. The second condition says that the entire profit of the grand coalition should be divided among its members. The *core* is a set of imputations such that $\sum_{G \in S} \psi_G(\mathcal{G}) \geq v(S), \forall S \subseteq \mathcal{G}$, i.e., for all coalitions, the payoff of any coalition is not greater than the sum of payoffs of its members in the grand coalition. The core contains payoff vectors that make the players want to form the grand coalition. The existence of a payoff vector in the core shows that the grand coalition is stable. Therefore, a payoff division is in the core if no player has an incentive to leave the grand coalition to join another coalition in order to obtain higher profit. In our previous work [25], we showed that the core of the VO formation game $(\mathcal{G}, v)$ can be empty. If the grand coalition does not form, independent and disjoint coalitions would form.

III. VO Formation Mechanism

In this section, we describe our proposed trust-based mechanism for VO formation and characterize its properties.

A. Trust-based VO Formation Mechanism (TVOF)

The proposed trust-based VO formation mechanism (TVOF) is given in Algorithm 1. The mechanism is executed by a trusted party that also facilitates the communication among VOs/GSPs.

TVOF uses a set $\mathcal{L}$ containing feasible VOs. We initialize $\mathcal{C}$ with the set of all GSPs, that is, all GSPs form a VO initially. In every iteration, TVOF executes a branch-and-bound method (IP-B&B) to solve the integer programming model given in equations (9) to (14) in order to find an optimal allocation for the application program $\mathcal{T}$ on $\mathcal{C}$. If the IP-B&B finds a feasible mapping that assigns all tasks $T \in \mathcal{T}$ to $\mathcal{C}$ satisfying the deadline (all constraints), $\mathcal{C}$ is added to $\mathcal{L}$. Then, TVOF calculates the reputation values of all GSPs in the VO using the power method (described in Section III) by calling Algorithm 2. The power method is one of many eigenvalue algorithms that can be used to find the largest eigenvector of the trust matrix, $A_{\mathcal{C}}$ representing the trust relationships among the GSPs in $\mathcal{C}$. Algorithm 2 starts by assigning the same reputation score to all GSPs in the VO $\mathcal{C}$. Then, it recomputes the reputation scores of each GSP as the weighted sum of the scores of all GSPs in a GSP's neighborhood. The algorithm repeats these steps until the average relative error between $\mathbf{x}^{q+1}$ and $\mathbf{x}^q$ is smaller than the given threshold ϵ. That means $\mathbf{x}^q$ does not change significantly any more and it represents the global reputation of the GSPs in $\mathcal{C}$. The algorithm returns the eigenvector representing the reputation of GSPs participating in the VO.

In every iteration, TVOF selects a GSP, G, with the lowest reputation in the VO $\mathcal{C}$ using $\mathbf{x}$. Then, TVOF removes it from the VO. If more than one GSPs have the same lowest reputation, the mechanism chooses one of them randomly. This changes the graph $(\mathcal{C}, \mathcal{E})$ by removing not only G, but also all edges with direct trust to G. This is a greedy choice for a VO since in each step a VO removes a GSP with the lowest reputation.

The mechanism recalculates the reputation scores for all remaining GSPs in the VO in every iteration. The recalculation step is necessary since the reputation of the VO members should be based on their opinion about the participating GSPs in the VO. The opinion of the GSP with the lowest reputation

Algorithm 1 Trusted VO Formation Mechanism (TVOF)

$\mathcal{L} = \emptyset$
$\mathcal{C} = \mathcal{G}$
repeat
  $flag \leftarrow TRUE$
  Map program $\mathcal{T}$ on $\mathcal{C}$ using IP-B&B
  if FEASIBLE **then**
    $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{C}$
    $flag \leftarrow FALSE$
  end if
  $\mathbf{x}$ = REPUTATION$(\mathcal{C}, \mathcal{E})$
  Find a GSP G with the lowest reputation in $\mathbf{x}$
  $\mathcal{C} = \mathcal{C} \setminus G$
until $flag$
Find $k = \arg\max_{\mathcal{C}_i \in \mathcal{L}} \{v(\mathcal{C}_i)/|\mathcal{C}_i|\}$
Map and execute program $\mathcal{T}$ on VO $\mathcal{C}_k$

Algorithm 2 REPUTATION$(\mathcal{C}, \mathcal{E})$

Input: Trust graph: $(\mathcal{C}, \mathcal{E})$
$A_{\mathcal{C}}$ = adjacency matrix of $(\mathcal{C}, \mathcal{E})$
$x_G^0 \leftarrow \frac{1}{|\mathcal{C}|}$ for all $G_i \in \mathcal{C}$
repeat
  $\mathbf{x}^{q+1} \leftarrow A_{\mathcal{C}}^T \mathbf{x}^q$
  $\delta \leftarrow ||\mathbf{x}^{q+1} - \mathbf{x}^q||$
until $\delta < \epsilon$
return $\mathbf{x}^{q+1}$

should not affect the value of the reputation of the other GSPs. This recalculation affects GSPs' reputations in the entire VO. As a result, TVOF considers only the opinions of the participating GSPs when calculating the global reputation, that is, the opinions of GSPs outside the VO do not have any effect on the reputation of the VO's members.

These iterations continue until TVOF finds a VO that could not execute the program. At the end, TVOF chooses a VO $\mathcal{C}_k$ from $\mathcal{L}$ that yields the highest individual payoff for its members. The program is executed by VO $\mathcal{C}_k$, a trusted subset of $\mathcal{G}$. This VO contains members with high reputation and yields the highest individual payoff for them.

B. Stability and efficiency

Since only one VO forms and executes the program, the formation of other VOs with GSPs outside of the selected VO is not important. We will be interested in characterizing the stability of the VO obtained by TVOF. In order to do this we will formally define the stability concept that will be used in this paper. We define the *VO preference relation* $\succeq_i$ for each member G_i. This allows G_i to compare two VOs and to indicate its preference to be a part of one of them. $A \succeq_i B$ implies that G_i prefers to be a member of VO A than to be a member of VO B, or at least it prefers both VOs equally. In addition, $A \succ_i B$ indicates that G_i strictly prefers to be a member of A than a member of B. We define a new concept of stability similar to the stability of a coalition structure defined in the context of the hedonic games [26]. We call this notion *individual stability* and we define it as follows.

Definition 1 (Individual stability): A coalition $\mathcal{C}$ is *individually stable* if there is no member $G_i \in \mathcal{C}$ such that $\mathcal{C} \setminus \{G_i\} \succeq_j \mathcal{C}$ for all $j \in \mathcal{C}$.
In other words, a VO $\mathcal{C}$ is individually stable if no GSP $G \in \mathcal{C}$ can leave $\mathcal{C}$ without making at least one GSP $G' \in \mathcal{C}$ unhappy.

Theorem 1: TVOF produces VOs that are individually stable.

Proof: (Sketch) We consider two cases to show that TVOF forms an individually stable VO. First, if G is the GSP that has the lowest reputation among all GSPs in the VO $\mathcal{C}$, then TVOF checks this case by removing G. Thus, we have two cases where either the VO is not feasible or the individual profit of GSPs is not as much as the individual profit of GSPs in $\mathcal{C}$. As a result, leaving G as part of the VO makes other GSPs in $\mathcal{C}$ unhappy. Second, if G is not the GSP that has the lowest reputation among all GSPs in the VO $\mathcal{C}$, then removing G decreases the total reputation of GSPs in $\mathcal{C}$, thus the participating GSPs would be unhappy because of G leaving the VO. As a result, the formed VO is individually stable. ■

Another important characteristics of the solution produced by TVOF is the optimality of the VO in terms of both profit and reputation. We now show that the VO produced by TVOF is a Pareto optimal solution for the VO formation problem.

Theorem 2: TVOF produces a Pareto optimal solution to the VO formation problem.

Proof: (Sketch) The set $\mathcal{L}$ in the description of TVOF contains the feasible VOs formed by TVOF. Based on the definition of Pareto optimality in Section II-C, we need to show that the resulting VO $\mathcal{C} \in \mathcal{L}$ from TVOF is not dominated by other VOs in both its individual payoff, π, and its average reputation, $\bar{x}$. Since in each step of the mechanism, TVOF removes a GSP with the lowest reputation, the high reputable GSPs are always in the VO. As a result, the GSPs outside the VO are not able to form a VO with higher average reputation than $\bar{x}$. That means, there is no VO $\mathcal{C}' \notin \mathcal{L}$ where $\bar{x}' \geq \bar{x}$. However, there may be other VOs $\mathcal{C}' \in \mathcal{L}$ that have higher average reputation. Those VOs do not have a higher individual payoff than π, since TVOF selects $\mathcal{C}$ which has the highest individual payoff among all VOs in $\mathcal{L}$. As a result, the VO formed by TVOF is a Pareto optimal solution. ■

IV. Experimental Results

We perform a set of simulation experiments which allows us to investigate how effective the proposed trust-based VO formation mechanism is in producing stable VOs.

A. Experimental Setup

We consider 16 GSPs which is a reasonable estimation of the number of GSPs in real grids. The number of GSPs is small since each GSP is a provider and not a single machine. We use real workloads from the Parallel Workloads Archive [27], [14] to drive our simulation experiments. More specifically we use the logs from the Atlas cluster at Lawrence Livermore National Laboratory (LLNL). This log consists of traces (collected from November 2006 to Jun 2007) that contain a good range of job sizes from 8 to 8832. We used the cleaned log LLNL-Atlas-2006-2.1-cln.swf which has 43,778 jobs. We selected 21,915

TABLE I: Simulation Parameters

Param.	Description	Value(s)
m	Number of GSPs	16
n	Number of tasks	$[8, 8832]$
s	GSP's speeds ($m \times 1$ vector)	4.91 $\times[16, 128]$ GFLOPS
w	Tasks' workload ($n \times 1$ vector)	$[17676, 1682922.14]$ GFLOP
t	Execution time matrix ($m \times n$)	$\frac{w}{s}$ seconds
c	Cost matrix ($m \times n$)	$[1, \phi_b \times \phi_r]$
d	Deadline	$[0.3, 2.0]$ $\times$ Runtime $\times$ $n/1000$ seconds
P	Payment	$[0.2, 0.4] \times max_c \times n$ units
ϕ_b	Maximum baseline value	100
ϕ_r	Maximum row multiplier	10
Runtime	Runtime of a job from Parallel Workloads Archive	≥ 7200 seconds
max_c	Maximum amount of cost	$\phi_b \times \phi_r$

jobs that completed successfully out of the total jobs of the log. About 13% of the total completed jobs are large jobs having runtimes greater than 7200 seconds.

The Atlas cluster [28] contains 1152 nodes, each with 8 processors which makes 9,216 processors in total. Each processor is an AMD dual-core Opteron with a clock speed of 2.4 GHz. The theoretical system peak performance of the Atlas cluster is 44.24 TFLOPS (Tera FLoating-point OPerations per Second). As a result, the peak performance of each processor is 4.91 GFLOPS (GigaFLOPS).

We selected six different application program sizes from the Atlas log, ranging from 256 to 8192 tasks. For each program, the number of allocated processors the job uses gives the number of tasks, and the average CPU time used in seconds gives the the average runtime of a task. We used the peak performance of a processor to convert the runtime to workload for each task. We generated the values of the other parameters based on the extracted data from the Atlas log. The parameters and their values are listed in Table I. The values for deadline and payment were generated in such a way that there exists a feasible solution in each experiment.

Each task has a workload expressed in Giga Floating-point Operation (GFLOP). To generate a workload, we extract the runtime of a job (in seconds) from the Parallel Workloads Archives, and multiply that by the performance (GFLOPS) of a processor in the Atlas system. This number gives the maximum amount of giga floating-point operations for a task. We assume that the workload of each task is in $[0.5, 1.0]$ of the maximum GFLOP of the job. The workload vector, w, contains the workload of each task of the application program.

The speed vector s is generated relative to the Atlas system. Each GSP has a speed chosen within the range $4.91 \times [16, 128]$ GFLOPS. This is due to the fact that each GSP can have several processors capable of performing 4.91 GFLOPS. The reason that we chose this range is that the number of processors in the Atlas is 9,216. If all 16 GSPs have the highest performance of 128×4.91, we would have 2048 processors that is 22.2 percent of the power of the Atlas system. As a result the deadline is generated at most 16 times larger than the runtime to make sure there is a feasible solution for the task allocation.

Based on the speed vector and the workload vector, the execution time of each task T_j on each GSP G_i is obtained. The execution time matrix is consistent if GSP G_i that executes any task T_j faster than GSP G_k, executes all tasks faster than GSP G_k [29]. The generated time matrix is consistent due to the fact that for every task $T_j \in \mathcal{T}$, $w(T_j)$ is fixed for all GSPs $G_i \in \mathcal{G}$, thus, for any task T_j if $t(T_j, G_i) < t(T_j, G_k)$ is true, then we have $s(G_i) > s(G_k)$ which means G_i is faster than G_k. As a result, $t(T_q, G_k) > t(T_q, G_i)$ is satisfied for all tasks $T_q \in \mathcal{T}$.

Each cost matrix c is generated using the method described by Braun *et al.* [29]. First, a baseline vector of size n is generated where each element is a random uniform number within $[1, \phi_b]$. Then, the rows of the cost matrix are generated based on the baseline vector. Each element j in row i of the matrix, $c(i, j)$, is generated by the element i of the baseline vector multiplied by a uniform random number within $[1, \phi_r]$, a row multiplier. Therefore, one row requires m different row multipliers. As a result, each element in the cost matrix is within the range $[1, \phi_b \times \phi_r]$.

We consider that the costs of GSPs are unrelated to each other, i.e., if $s(G_i) > s(G_k)$, for any task T_j, either $c(T_j, G_i) \leq c(T_j, G_k)$ or $c(T_j, G_k) \leq c(T_j, G_i)$ is true. This is due to GSPs policies. However, we consider that the costs are related to the workload of the tasks, i.e., for two tasks T_j and T_q where $w(T_j) > w(T_q)$, we have $c(T_j, G_i) > c(T_q, G_i)$ for all $G_i \in \mathcal{G}$. A task with the smallest workload has the cheapest cost on all GSPs.

We use the Erdös-Rényi model to generate random trust graphs connecting the GSPs [30]. An Erdös-Rényi graph (m, p) is a graph constructed by connecting nodes randomly where the graph has m nodes. The probability of having an edge in the graph is p for any pair of nodes, and it is independent from every other edge. That means, all graphs with m nodes and e edges have equal probability. Based on the parameter $p \in [0, 1]$, the graph can be sparse or complete. In these experiments, $m = 16$ is the number of GSPs, and $p = 0.1$. We use the ILOG Concert Technology APIs in C++ to solve the IP problem by CPLEX solver provided by IBM ILOG CPLEX Optimization Studio for Academics Initiative [31].

B. Analysis of Results

We compare the performance of our trust-based VO formation mechanism (TVOF) with that of another mechanism, Random VO Formation (RVOF). The RVOF mechanism is the same as TVOF, but instead of removing a GSP with the lowest reputation score from a VO, RVOF removes a GSP without considering its reputation score. As a result, in each step a GSP is removed from a VO randomly. Both mechanisms use the branch-and-bound method to find the mapping of the tasks to GSPs in a VO. This allows us to focus on the VO formation and not on the choice of the mapping algorithms. We performed a series of ten experiments for each case, and we represented the average of the obtained results.

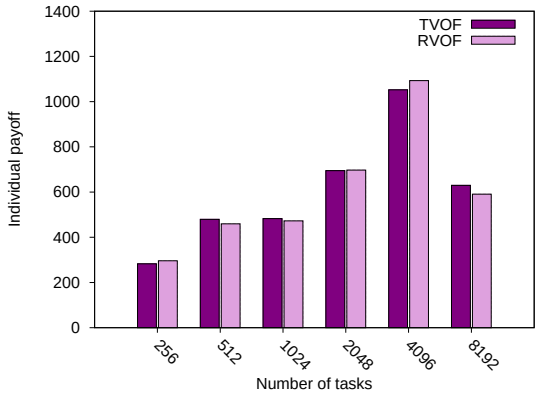

Fig. 1: GSP's Individual Payoff

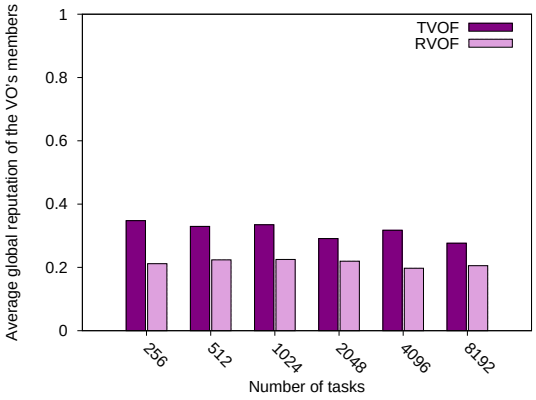

Fig. 3: GSP's Average Reputation

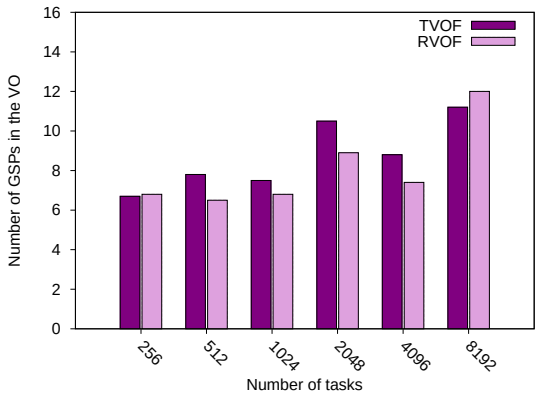

Fig. 2: Size of Final VO

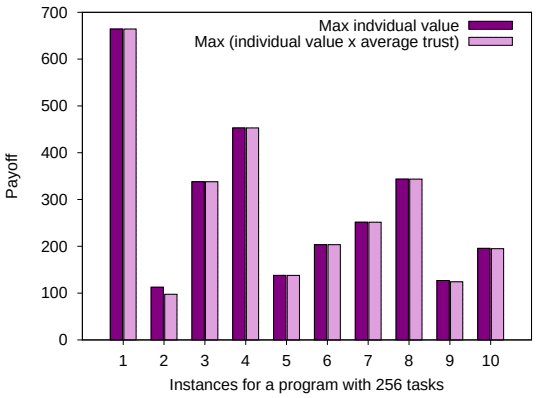

Fig. 4: GSP Individual payoffs obtained by TVOF

In Fig. 1, we show the performance of TVOF and RVOF, in terms of the individual GSP's payoffs in the final VO, as a function of the number of tasks. Both mechanisms select a VO with the highest individual payoff as the final VO to execute the application program. The figure shows that on average both mechanisms lead to the same amount of payoff for the GSPs participating in the final VO. This is due to the fact that both TVOF and RVOF select the VO that yields the highest individual payoff for the GSPs. In addition to selecting the VO yelding the highest individual profit, TVOF also guarantees that the selected VO is composed of GSPs with the highest average global reputation.

In Fig. 2, we show the size of the final VO obtained by TVOF and RVOF. This figure shows that as the number of tasks increases the size of the VO obtained by TVOF increases. This means that the more tasks the more GSPs pool their resources to form a VO in order to execute the program. The VOs formed by TVOF do not necessary have smaller size than the VOs obtained by RVOF.

In Fig. 3, we show the average global reputation of the GSPs in the final VO obtained by TVOF and RVOF. This figure shows that TVOF forms VOs composed of more reputable GSPs. The average global reputation of the members of the VOs produced by TVOF is higher in all cases than the average reputation of the members of the VOs obtained by RVOF. In addition, TVOF tries to keep the average reputation scores the same for all VOs. By considering the individual payoff in Fig. 1, the results show that TVOF not only provides the highest average reputation for GSPs in the VO, but also it provides reasonable individual payoff for them.

Fig. 4 shows the individual payoff of GSPs participating in the VO obtained by TVOF. We select 10 different programs with 256 tasks. TVOF selects the VO that provides the highest individual payoff. In addition, this figure shows the individual payoff of GSPs in the VO with the highest product of individual payoff and average reputation among the VOs in the list $\mathcal{L}$ maintained by TVOF. That means, we find a VO that has the highest product of individual payoff and average reputation among all VOs in $\mathcal{L}$. For that VO, we show the individual payoff for its members. Comparing these two cases, the results show that in most cases, TVOF not only finds the VO with the highest individual payoff, but also the obtained VO has the highest average reputation score. As a result, the TVOF mechanism provides the Pareto optimal VO.

The Pareto optimality of the achieved results can also be seen from Fig. 5 and Fig. 6. In these figures, we show the formation of VOs for two programs A and B consisting of 256 tasks. Both figures show the results of all iterations of

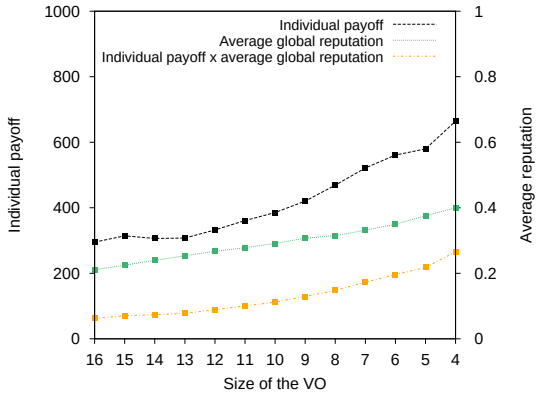

Fig. 5: Program A: Results of TVOF iterations

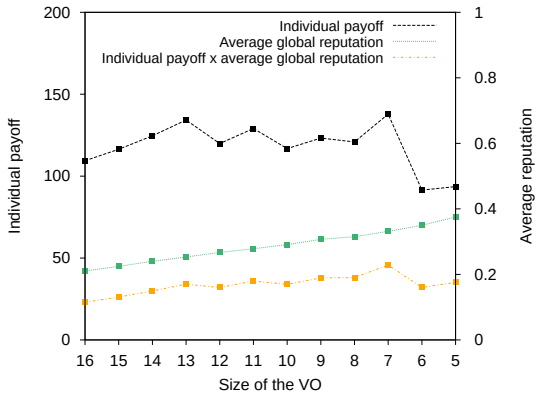

Fig. 6: Program B: Results of TVOF iterations

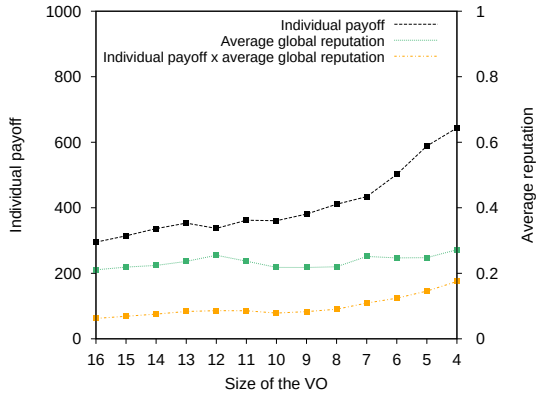

Fig. 7: Program A: Results of RVOF iterations

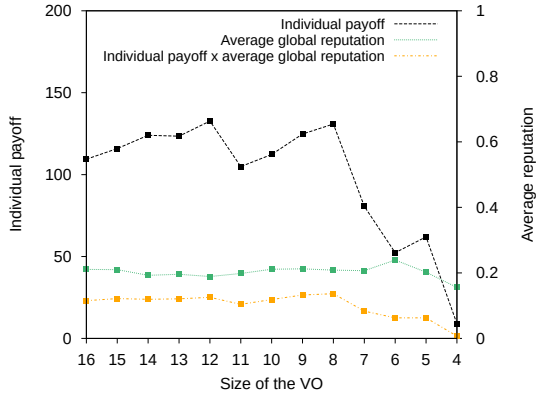

Fig. 8: Program B: Results of RVOF iterations

TVOF mechanism. The VO formed by all 16 GSPs has the lowest individual payoff for them (left vertical axis) and also has the lowest average global reputation (right vertical axis). By reducing the size of a VO and removing a GSP with the lowest reputation score, the average global reputation values increase. In the case of program A, shown in Fig. 5, a VO with 4 GSPs is the final VO obtained by TVOF that provides the highest individual payoff and the highest average global reputation. In the case of program B, shown in Fig. 6, a VO with 7 GSPs is the final VO obtained by TVOF that provides the highest individual payoff. The final VO does not have the highest average global reputation, but it provides the highest product of individual payoff and average global reputation.

We should mention again that TVOF selects the final VO with the highest individual payoff, but this VO is the one that also has the highest average reputation. This is due to the fact that in each step, TVOF removes a GSP with the lowest reputation among all GSPs in a VO.

We show the formation of VOs using RVOF for the same programs A and B in Fig. 7 and Fig. 8, respectively. Both figures show the results of all iterations. The average global reputation changes in all iterations, but it does not increase since GSPs are removed randomly. Selecting a VO with the highest individual payoff does not provide the highest product of individual payoff and average global reputation. Comparison of Fig. 5 and Fig. 7 for program A, and Fig. 6 and Fig. 8 for program B, shows the effectiveness of our proposed TVOF mechanism.

Fig. 9 shows the execution time of TVOF and RVOF. The TVOF's execution time is reasonable given that the application program would require several hours to execute. The reason for getting higher execution times for 4096 and 8192 tasks is that finding the mapping takes more time.

From the above results, we conclude that the proposed VO formation mechanism is able to form stable VOs that ensure the program is completed before its deadline and provide the highest individual payoff for the GSPs.

V. Conclusion

We proposed a novel mechanism for VO formation in grids considering the trust relationships among grid service providers. In the proposed mechanism, GSPs cooperate to form VOs with high reputation GSPs in order to execute application programs. We modeled the problem as a coalitional game and derived a centralized VO formation mechanism. To find the optimal configuration of all the tasks on participating GSPs in a VO, we used an branch-and-bound method. We showed that our proposed mechanism produces a stable

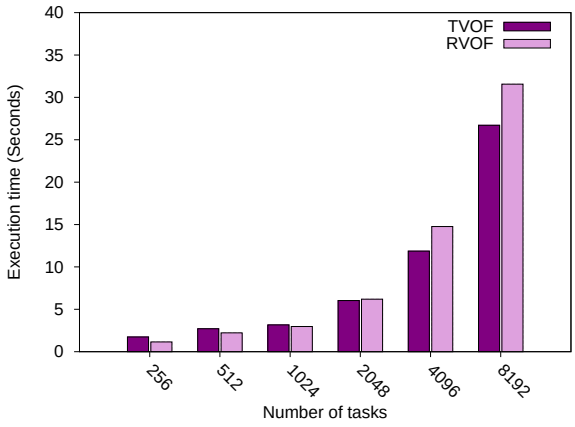

Fig. 9: Execution Time

coalition in terms of reputation and individual payoff. We performed extensive experiments with the data extracted from real workload traces to investigate its properties. Experimental results showed that the VO obtained by TVOF maximizes the reputation of the participating GSPs. In addition, most of the time TVOF determines the final VO with the highest individual payoffs for its members. The mechanism's execution time is reasonable given that applications programs would require several hours to execute. We believe that this research will encourage grid service providers to adopt trust-based VO formation mechanisms and use them to pool their resources together in order to execute application programs. In future work, we would like to consider the task dependencies in our VO formation model and design new mechanisms for VO formation.

ACKNOWLEDGMENT

This research was supported in part by NSF grants DGE-0654014 and CNS-1116787.

REFERENCES

[1] C. Hang, Y. Wang, and M. Singh, "Operators for propagating trust and their evaluation in social networks," in *Proc. of the 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 2009, pp. 1025–1032.

[2] U. Kuter and J. Golbeck, "Sunny: A new algorithm for trust inference in social networks using probabilistic confidence models," in *Proc. of the National Conference on Artificial Intelligence*, vol. 22, no. 2, 2007, p. 1377.

[3] 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.

[4] K. Avrachenkov, D. Nemirovsky, and K. Pham, "A survey on distributed approaches to graph based reputation measures," in *Proc. of the 2nd International Conference on Performance Evaluation Methodologies and Tools*, 2007, p. 82.

[5] R. Hanneman and M. Riddle, *Introduction to social network methods*. University of California Riverside, 2005.

[6] L. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1979.

[7] N. Friedkin, "Theoretical foundations for centrality measures," *American Journal of Sociology*, pp. 1478–1504, 1991.

[8] S. Borgatti and M. Everett, "A graph-theoretic perspective on centrality," *Social Networks*, vol. 28, no. 4, pp. 466–484, 2006.

[9] F. Azzedin and M. Maheswaran, "Integrating trust into grid resource management systems," in *Proc. of the 31st International Conference on Parallel Processing*, 2002, pp. 47–54.

[10] L. Lin and J. Huai, "QGrid: an adaptive trust aware resource management framework," *IEEE Systems Journal*, vol. 3, no. 1, pp. 78–90, 2009.

[11] G. von Laszewski, B. Alunkal, and I. Veljkovic, "Toward reputable grids," *Scalable Computing: Practice and Experience*, vol. 6, no. 3, 2005.

[12] B. Alunkal, I. Veljkovic, G. Von Laszewski, and K. Amin, "Reputation-based grid resource selection," *Proceedings of the Workshop on Adaptive Grid Middleware*, 2003.

[13] S. Kamvar, M. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proc. of the 12th International Conference on World Wide Web*, 2003, pp. 640–651.

[14] Parallel workloads archive. [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/

[15] W. Cirne, D. Paranhos, L. Costa, E. Santos-Neto, F. Brasileiro, J. Sauvé, F. Silva, C. Barros, and C. Silveira, "Running bag-of-tasks applications on computational grids: The mygrid approach," in *Proc. of the 32nd International Conference on Parallel Processing*, 2003, pp. 407–416.

[16] C. Weng and X. Lu, "Heuristic scheduling for bag-of-tasks applications in combination with QoS in the computational grid," *Future Generation Computer Systems*, vol. 21, no. 2, pp. 271–280, 2005.

[17] F. da Silva, S. Carvalho, and E. Hruschka, "A scheduling algorithm for running bag-of-tasks data mining applications on the grid," in *Proc. of the 10th International Euro-Par Conference*, 2004, pp. 254–262.

[18] G. Golub and C. Van Loan, *Matrix computations*. Johns Hopkins Univ. Press, 1996, vol. 3.

[19] P. Bonacich and P. Lloyd, "Eigenvector-like measures of centrality for asymmetric relations," *Social Networks*, vol. 23, no. 3, pp. 191–201, 2001.

[20] P. Bonacich, "Some unique properties of eigenvector centrality," *Social Networks*, vol. 29, no. 4, pp. 555–564, 2007.

[21] G. Owen, *Game Theory*, 3rd ed. New York, NY, USA: Academic Press, 1995.

[22] K. Apt and A. Witzel, "A generic approach to coalition formation," *International Game Theory Review*, vol. 11, no. 3, pp. 347–367, 2009.

[23] L. Shapley, "A Value for n-person Games," in *Contributions to the Theory of Games*, H. Kuhn and A. Tucker, Eds. Princeton University Press, 1953, vol. II, pp. 307–317.

[24] O. Shehory and S. Kraus, "Task allocation via coalition formation among autonomous agents," in *Proc. of the International Joint Conference on Artificial Intelligence*, vol. 14, 1995, pp. 655–661.

[25] L. Mashayekhy and D. Grosu, "A merge-and-split mechanism for dynamic virtual organization formation in grids," in *Proc. of the 30th IEEE International Performance Computing and Communications Conference*, 2011, pp. 1–8.

[26] A. Bogomolnaia and M. Jackson, "The stability of hedonic coalition structures," *Games and Economic Behavior*, vol. 38, no. 2, pp. 201–230, 2002.

[27] S. Chapin, W. Cirne, D. Feitelson, J. Jones, S. Leutenegger, U. Schwiegelshohn, W. Smith, and D. Talby, "Benchmarks and standards for the evaluation of parallel job schedulers," in *Proc. of the 5th Workshop on Job Scheduling Strategies for Parallel Processing*, 1999, pp. 67–90.

[28] Atlas. [Online]. Available: https://www.llnl.gov/news/newsreleases/2007/NR-07-04-05.html

[29] T. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen *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.

[30] P. Erdös and A. Rényi, "On random graphs, I," *Publicationes Mathematicae (Debrecen)*, vol. 6, pp. 290–297, 1959.

[31] IBM ILOG CPLEX Optimization Studio for Academics Initiative. [Online]. Available: http://www01.ibm.com/software/websphere/products/optimization/academic-initiative/