

Formation of Virtual Organizations in Grids: A Game-Theoretic Approach

Thomas E. Carroll¹, Daniel Grosu²

*Dept. of Computer Science, Wayne State University
5143 Cass Avenue, Detroit, Michigan 48202 USA*

¹tec@cs.wayne.edu

²dgrosu@cs.wayne.edu

Abstract—Applications require the composition of resources to execute in a grid computing environment. The Grid Service Providers (GSPs), the owners of the computational resources, must form Virtual Organizations (VOs) to be able to provide the composite resource. We consider grids as self-organizing systems composed of autonomous, self-interested GSPs that will organize themselves into VOs with every GSP having the objective of maximizing its profit. Using game theory, we formulate the resource composition among GSPs as a coalition formation problem and propose a framework based on cooperation structures to model it. Using this framework, we propose a resource management system that supports the VO formation among GSPs in a grid computing system.

I. INTRODUCTION

Grid computing is *the* preferred computational platform of choice for collaborative, resource intensive applications which are typical in the domains of science and engineering [1]. There are two classes of participants in the grid, the users and the service providers. The users submit applications to be executed. The service providers, who are geographically distributed over the world, provide and operate resources to execute programs. Each provider has its own administrative domain and thus it is largely autonomous.

Applications require the composition of resources owned by several Grid Service Providers (GSPs). Service providers will form Virtual Organizations (VOs) [1] to provide the necessary resources on a *dynamic, per application basis*. The VO may dissolve as soon as the application has completed execution. Execution results in the service providers incurring costs (*e.g.*, power, administrators wages and time). If we assume that the service providers are *rational* (self-interested and welfare-maximizing), they will refuse to offer their resources unless they can recover their costs.

The rationality assumption permits analysis by economic models. Of particular interest are *coalitional games*, which are studied in game theory. Coalitional games model the interactions between groups of decision-makers. In this case the decision-makers are the service providers. The service providers form VOs in such a way that each provider maximizes its own profit. The VOs provide the composite resource needed to execute applications.

A VO is traditionally conceived for the sharing of resources, but it can also represent a business model [1]. In this work, a VO is a coalition of GSPs who desire to maximize their

individual profits and are largely indifferent about the global welfare.

In this paper we examine VO formation using models from coalitional game theory. We assume that a grid user submits a program and a specification consisting of a deadline and payment. A VO will form and execute the program. If the VO completes the program before the deadline, the VO will be paid; otherwise, the VO incurs the cost of execution. A GSP attempts to form a VO with other GSPs that maximizes its profit. But this is not simple as for n service providers, there are 2^n potential VOs in which a GSP can be part of. Further complicating the matter is that computing the worth of a coalition of GSPs requires determining a mapping that assigns application tasks to providers. This problem is known to be NP-hard. This process clearly takes an exponential amount of time and it makes completing the application before its deadline difficult. Consequently, a service provider cannot possibly compute the worth of all coalitions and thus has limited information to base its decision. It is likely that service providers will employ heuristics to expedite the process. These heuristics would be held private as they give providers significant economic advantages over the others. The advantage is only gained though, if others can be convinced to form the preferred coalition. This can be readily done by disclosing the coalition and its mapping.

Using the model that we developed, we propose a *resource management system*, the Virtual Organization Formation Manager (VOFM), to support VO formation among GSPs. The system is a middleware component which is supported by the grid implementation. The VOFM would be replicated across the grid, allowing users and GSPs to interact with local copies. The VOFM provides support for exchanging mappings and other information between GSPs to facilitate the VO formation process. Out of the set of resulting VOs, it selects the one that will execute the program.

A. Our Contributions

We model the Virtual Organization (VO) formation as a coalition formation problem. We use Myerson's cooperation structures [2] as the underlying coalition formation model. Coalition formation requires determining the value of various coalitions. GSPs will employ heuristics in determining these values. We design one such heuristic that can be used by the

GSPs. It is likely that GSPs will derive different heuristics that benefit themselves and consequently, they would be unwilling to share them. We examine this situation and show that GSPs can benefit from different heuristics if they share the results. Finally, we propose a resource management system that facilitates VO formation by providing the necessary structures for GSP information control and exchange.

B. Related Work

An important research topic in grid computing is the formation of virtual organizations. A *Virtual Organization* (VO) is an alliance among GSPs to collaborate and to pool their resources to compute large scale applications [1]. VO requirements for a grid architecture are discussed in [3]. Dynamic VO formation among autonomous agents and the management of such VOs are examined in [4]. VO formation in the CONOISE project is described in [5]. Furthermore, [6] describes the mechanisms for supporting dynamic VO formation and operation in CONOISE-G. VO formation requires resource discovery, which is a time consuming process. In [7], agents offering similar resources or composite resources form *communities* that are registered with the resource discovery service, thus, minimizing discovery costs. Coalitional game theory can be used to model VO formation among GSPs. *Coalitional game theory* examines the interactions between groups of decision makers. A good reference for coalitional game theory is [8]. One topic of interest is coalition formation, the partitioning of players into disjoint sets. There are many models of coalition formation (e.g., [9], [10]), but we are particularly interested in Myerson's cooperation structures [2]. Much research on coalition formation has been conducted in the multi-agent systems area for wide assortment of problems including distributed artificial intelligence [11] and service composition [12]. Further research has been conducted on task allocation and resource composition. Shehory and Kraus [13] proposed a distributed decision processing system in which the processing agents partition themselves into subsets with the goal of minimizing the ratio between coalition cost and coalition size. A taxonomy and classification of task allocation problems is given in [14]. For each class, a welfare-maximizing algorithm is proposed and its complexity analyzed. In [15], agents exchange idle time for the purpose of compositing resources to execute tasks. A simple non-game theoretic approach to coalitional formation in computational grids is proposed in [16]. Finally, machine learning techniques are employed to optimize coalition efficiency in [17]. Non-cooperative game theory was used to study the problem of scheduling tasks on parallel machines in [18], [19]. Several researchers have applied market-based approaches to resource allocation in grid computing [20], [21], [22], [23]. Grosu and Das [24] investigated the benefits of various types of auction-based allocation protocols. Grosu [25] proposed an architecture for strategyproof resource allocation in grids.

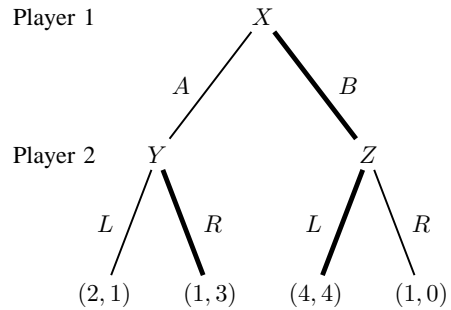


Fig. 1. A 2-player, 2-turn game with the actions that are part of the subgame perfect equilibria represented by thick lines.

C. Organization

This paper is structured as follows. In Section II we introduce the main concepts of coalitional game theory. In Section III we define the model that is employed throughout the paper. In Section IV we propose a framework and a model of coalition formation based on cooperation structures. In Section V we propose a resource management system to support VO formation in grid computing based on the proposed coalition formation framework. Finally, we draw conclusions and present future directions in Section VI.

II. COALITIONAL GAME THEORY

In this section we introduce the main concepts of coalitional game theory that are employed in this paper. *Coalitional game theory* studies the interactions between groups of *decision-makers* (the *players*).

A coalitional game comprises a set of players, N of cardinality n . Every subset S of N , where S is a coalition, has a *value* or *worth* given by the *characteristic function* $v(S)$. The value can be thought of as the profit obtained when the members of a coalition work as a group. We formally define a coalitional game as follows.

*Definition 1 (Coalitional game):*¹ A *coalitional game* (v, N) is characterized by the *player set* N and the *characteristic function* $v : S \subseteq N \rightarrow \mathbb{R}^+$ such that $v(\emptyset) = 0$.

An important problem in coalitional game theory is coalition formation. *Coalition formation* is the partitioning of the players into disjoint sets. The set of coalitions, $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$, is 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 $\bigcup_{S_i \in \mathcal{S}} S_i = N$.

There are several models of coalition formation. One model is *coalitional structures* [9]; another model is *cooperation structures* [26], [2], which we discuss in detail in Section IV. The coalition formation under cooperation structures is represented as an extensive-form game. In an *extensive-form game* (or simply *extensive game*), the players take turns playing the game.

An extensive game can be modeled as a topological tree where the interior nodes are *turns*, the edges are *actions*, and

¹Note that we use the relaxed definition of a coalitional game which does not require superadditivity. Superadditivity is the property that for two disjoint sets R and S of N , $v(R \cup S) \geq v(R) + v(S)$

the leafs are the *outcomes* of the game. The outcomes are represented as tuples; the i -th entry of an outcome is Player i 's profit. The move of the first player is depicted at the root of the tree. As an example, Figure 1 depicts a 2-player, 2-turn game, where Player 1 moves first, followed by Player 2. Player 2 can observe the action of Player 1 (thus, this game is of *perfect information*). To solve such games, we use the concept of subgame perfect equilibria. Nodes X , Y , and Z are *subgames* of the complete game depicted in Figure 1. A solution is a *subgame perfect equilibrium* when no player can do better by changing her action for any subgame, when all the other players' actions are unchanged. Player 2 must choose either L or R for each of the subgames Y and Z . For subgame Y , she chooses R and for Z , she chooses L which are the actions that maximize her profit. Player 1 knows which actions Player 2 will choose due to the rationality assumption. Therefore, she chooses action B which leads to a profit of 4 units instead of A which only achieves 1 unit of profit. We have identified the actions that are subgame perfect by thick, solid lines in Figure 1. The procedure we used to determine the subgame perfect equilibria is called *backward induction*. Let the *length* of the subgame be the largest number of actions needed to reach a leaf. By convention, the length of terminal subgames (nodes Y and Z in Figure 1) is one. Backward induction computes the equilibria for subgames of length ℓ to compute the equilibria of games of length $\ell + 1$. Backward induction gives a set of strategy profiles. A *strategy profile* gives the action taken for each and every subgame in an extensive game. Each strategy profile determines an outcome, but different strategy profiles need not determine unique outcomes.

In the next sections we present a model for grid computing comprising autonomous service providers and show how the service providers will form VOs to execute programs.

III. MODEL

A user has an application program $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$ comprising n independent tasks that she desires to have executed to completion before deadline d . A set of grid service providers, $\mathcal{G} = \{G_1, G_2, \dots, G_m\}$, provides resources for executing programs. We assume that each service provider is autonomous and it behaves rationally (self-interested and welfare-maximizing). Furthermore, we assume that each service provider G_j , for $j = 1, 2, \dots, m$, owns and operates a single machine. The machines owned by GSPs form a network of heterogeneous, unrelated machines. The cost function, $c : \mathcal{T} \times \mathcal{G} \rightarrow \mathbb{R}^+$, gives the cost of executing each task $T \in \mathcal{T}$ when assigned to each service provider $G \in \mathcal{G}$. Additionally, the function $t : \mathcal{T} \times \mathcal{G} \rightarrow \mathbb{R}^+$ gives the execution time of each task when assigned to each GSP. It is assumed that once a task is assigned to a GSP it is executed to completion on the machine owned by that GSP, *i.e.*, the task is neither preempted nor migrated. Techniques such as code profiling [27] and statistical prediction [28] can be used to estimate task execution times. The user submits the program, deadline d , cost function c , and execution time function t to the grid resource manager.

The user experiences an increase in welfare, ΔW , for having the program completed. She is willing to pay P , such that $P \leq \Delta W$, if the program is executed to completion by deadline d . If the program execution exceeds d , the user does not experience a welfare increase (*i.e.*, $\Delta W = 0$) and thus, is not willing to pay any amount (*i.e.*, $P = 0$) as she is rational.

As was stated above, GSPs incur cost for executing tasks. Service providers form VOs in order to have the necessary capacity to execute the program and more importantly, maximize their profits. The profit is simply defined as the difference between payment P and execution costs. In our model we represent VOs as coalitions of GSPs.² For each coalition S , where $S \subseteq \mathcal{G}$, there exists a mapping $\pi_S : \mathcal{T} \rightarrow S$, which assigns task $T \in \mathcal{T}$ to service provider $G \in S$. The costs incurred for executing the program \mathcal{T} on S under mapping π_S is given by

$$C(\mathcal{T}, S) = \sum_{T \in \mathcal{T}} \sum_{G \in S} \sigma_S(T, G) c(T, G), \quad (1)$$

where $\sigma_S(T, G)$ is an indicator such that

$$\sigma_S(T, G) = \begin{cases} 1 & \text{if } \pi_S(T) = G, \\ 0 & \text{if } \pi_S(T) \neq G. \end{cases} \quad (2)$$

The execution time of the program is given by its *makespan* (*i.e.*, completion time) as induced by the mapping π_S . The execution time is given by

$$E(\mathcal{T}, S) = \max_{G \in S} \sum_{T \in \mathcal{T}} \sigma_S(T, G) t(T, G). \quad (3)$$

Each GSP has the objective of determining the coalition in which it is a member and where it will receive the greatest profit. The money earned by the coalition is divided in some fashion among its members. Even if a coalition has low costs, it may be unattractive due to its low profit margins.

We formalize this scenario as a coalitional game (v, N) , where the player set N is the set of GSPs \mathcal{G} . The characteristic function v is then defined as

$$v(S) = \begin{cases} 0 & \text{if } |S| = 0 \text{ or } E(\mathcal{T}, S) > d, \\ P - C(\mathcal{T}, S) & \text{if } |S| > 0 \text{ and } E(\mathcal{T}, S) \leq d, \end{cases} \quad (4)$$

where $|S|$ is the cardinality of S . Note that $v(S)$ satisfies the constraint $v(\emptyset) = 0$.

Now the question is how the profit will be divided among the members of a coalition. Traditionally, the *Shapley value* [29], [30] would be employed, but computing the Shapley value requires iterating over every partition of a coalition, an exponential time endeavor. If we assume that all members remain in the coalition until the program is completed, the *equal sharing* of the profit among members seems justifiable. Equal sharing has been successfully used in other systems where tractability is critical (*e.g.*, [13]).

²Since coalitions have a well defined meaning in game theoretical modeling we will use the term coalition instead of VO when we describe the theoretical model of VO formation.

As we have stated above the welfare-maximizing GSP will determine its coalition by considering the profit it earns and not the coalition value. Therefore, a service provider G determines its preferred coalition S by solving:

$$\max_{(S)} \frac{P - C(\mathcal{T}, S)}{|S|} \quad (5)$$

subject to:

$$E(\mathcal{T}, S) \leq d \quad (6)$$

and

$$G \in S. \quad (7)$$

In the above, (5) is G 's profit maximizing objective subject to the constraints that the program is completed before the deadline (6), and that G is a member of the coalition (7).

In this section we discussed the motivations and presented the model for coalition formation. Next, we discuss the VO formation process.

IV. VIRTUAL ORGANIZATION FORMATION

In this section we investigate the virtual organization formation process considering the coalitional game model proposed in Section III. A GSP must choose which VO to form (or it can decide to form a VO consisting just of itself). Since the GSP is self-interested, it will choose to form a VO where it maximizes its profit. As in the previous section, we will use the term coalition when we refer to VO.

Computing the best coalition S requires determining the mapping π_S . Assume for the moment that determining the mapping is tractable. Service provider G computes its profit for each of the potential 2^{m-1} coalitions where it can be a member and then chooses the coalition S that maximizes its own profit, *i.e.*, G 's preferred coalition is $S = \arg \max_{S \subseteq \mathcal{G}} v(S)/|S|$. Note that there may exist coalitions such as S' with greater values than G 's choice, but these coalitions do not maximize G 's profit, *i.e.*, $v(S) < v(S')$ but $v(S)/|S| > v(S')/|S'|$. Determining a preferred coalition is further complicated by the fact that coalition formation is a non-cooperative process. The following example illustrates why. Let G determine its preferred coalition as S and let S contain GSP G' . Similarly, G' determines its preferred coalition as S' , but S' does *not* contain G as a member. The question is will G and G' be members of the same coalition or will they be members of different coalitions? These issues are resolved by the appropriate coalition formation model.

One model of coalition formation is *coalition structures* [9]. Another approach to coalition formation and the one we choose, is based on Myerson's *cooperation structure* (or *cooperation graph*) [2], [26]. In the cooperation structure model coalitions form by *building links* between pairs of GSPs. A *link* signifies that the service providers can carry on direct negotiations within a coalition. When instantiating a link, both GSPs must agree to the link or the link is not created. The link outcome is announced to all. Once a link is formed, it cannot be broken (the links are enforced by binding contracts). As

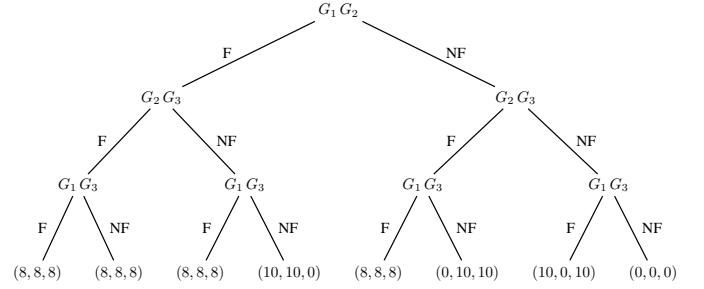


Fig. 2. A coalition formation game among three Grid service providers G_1 , G_2 and G_3 . F signifies that “the link is formed”, while NF signifies that “the link is not formed”.

the links are formed each GSP updates what it perceives to be its best coalition.

Before link building commences, an *order of rule* is announced that specifies the order in which links can be built. We are assuming a fully connected network, thus the order of rule contains the $n(n-1)/2$ possible links. Another consequence of the fully connected network is that each member of a coalition will have a link to every other member of that coalition [26].

We now provide a simple example of coalition formation. Let $\mathcal{G} = \{G_1, G_2, G_3\}$ and the resulting $v(S)$ be

$$v(S) = \begin{cases} 0 & \text{if } |S| \leq 1, \\ 20 & \text{if } |S| = 2, \\ 24 & \text{if } |S| = 3. \end{cases} \quad (8)$$

The order of rule is $\{G_1 G_2, G_2 G_3, G_1 G_3\}$. The link $G_1 G_2$ is decided first. We examine the case in which GSP G_1 and G_2 form a link, each obtaining 10 units of profit. If G_2 builds the link to G_3 , its profit will correspondingly reduce from 10 to 8 units. Thus, G_2 will reject forming this link. Similarly, G_1 will reject forming the link to G_3 . Thus the final coalitions are $S = \{S_1, S_2\}$, where $S_1 = \{G_1, G_2\}$ and $S_2 = \{G_3\}$. This example has another potential coalition set that is discussed in the following.

The service providers are assumed to be *farsighted* when deciding which coalition to form. A GSP decides on a link not on the present level of profit but on the profit of the coalition when formation completes. The formation can be modeled as an extensive game in which each “turn” is given by the order of rule. Solving for subgame perfection by backward induction determines what the final coalitions will appear as. The tree in Figure 2 represents the coalition formation game with $v(S)$ as given in (8) and the order of rule as $\{G_1 G_2, G_2 G_3, G_1 G_3\}$. In the figure F signifies “the link is formed” and NF signifies “the link is not formed”. The outcomes are represented as tuples, where the i -th position is G_i 's profit. The actions represent forming a link between pairs of GSPs. During a turn, the involved GSPs decide whether to form the link. Both GSPs must agree to the link to have it formed; if either GSP refuses, the link is not formed. There are four subgames with the action of forming a link between G_3 and G_1 . Beginning with the left most subgame, forming or not forming the link results in the same outcome of $(8, 8, 8)$. So the link may or

may not be formed. In the second subgame from the left, G_1 will reject building a link and thus earn a better outcome $((10, 10, 0))$ than if the link was formed $((8, 8, 8))$. In the third subgame, G_3 will reject the link request as there already exists a link between it and G_2 . In the last subgame, both G_1 and G_3 agree to the link. Now looking at the left subgame regarding the formation of a link between G_2 and G_3 , G_2 will choose not to form the link as it leads to the better outcome of $(10, 10, 0)$. In the right subgame, G_2 and G_3 agree to the link as it results in outcome $(0, 10, 10)$. But if the link is rejected, G_3 can still make the same profit by linking with G_1 . In the topmost subgame located at the root of the tree, G_1 and G_2 have the option of forming a link. If the link is not formed, G_1 can link with G_3 or G_2 will link with G_3 . There are six strategy profiles for the game, leading to two coalition sets of $\{\{G_1 G_2\}, \{G_3\}\}$ and $\{\{G_1 G_3\}, \{G_2\}\}$.

Up to this point we have assumed that computing an optimal mapping π_S in terms of cost is tractable. It is known, though, that the problem is NP-hard for $m \geq 2$ [31]. To determine the optimal coalition, a farsighted service provider would be required to evaluate an optimal mapping for all 2^m coalitions. This, by its very nature, would require exponential time. When a program is released, a substantial amount of time would be spent just on choosing a coalition, which certainly reduces the probability of successfully completing the program before its deadline. Traditionally, most research focused on minimizing the makespan of the program. The heuristic *Longest Process Time* (LPT) in which the tasks are ordered by non-increasing execution time and then scheduled using list scheduling has been shown to be a good approximation of the makespan in both identical [32] and uniform machine environments [33]. For unrelated machine environments, [34] provides an algorithm that is characterized by polynomial time complexity. Deadline scheduling heuristics such as *Earliest Deadline First* (EDF) appear to be inappropriate as they only consider the number of tasks completed before deadline and not the costs of the assignments. Deadline scheduling is a more difficult problem as it considers varying task release time and deadline, concepts that just complicate the working model.

We design a heuristic based on list scheduling that computes an approximate solution for the lowest cost mapping given a coalition S . The heuristic assumes that minimizing the total cost results in maximizing the service provider's profit. Service providers can use the following algorithm, MinCost, to compute a mapping for any coalition S that approximately minimizes cost.

Algorithm 1 (MinCost):

Input: program $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$;
coalition $S = \{G_i, \dots, G_j, \dots, G_k\}$;
cost function $c(T, G)$;
time function $t(T, G)$;
deadline d ;
payment P

Output: Map π_S

1. **for** $i = 1, 2, \dots, n$ **do**
2. Create a priority queue TQ_i ordered by

- non-increasing cost for
tuples $(T_i, G_j, c(T_i, G_j))$ for any i and j
3. Create an empty priority queue Q ordered by
non-increasing cost for tuples $(T_i, G_i, c(T_i, G_i))$
4. **for** $i = 1, 2, \dots, n$ **do**
5. $Q.enqueue \leftarrow TQ_i.dequeue$
6. **for** $i = 1, 2, \dots, n$ **do**
7. $d_i \leftarrow 0$
8. **while not** $Q.empty$ **do**
9. $(T_i, G_j, c(T_i, G_j)) \leftarrow Q.dequeue$
10. **if** $d_i + t(T_i, G_j) \leq d$ **then**
11. $\pi.assign(T_i) \leftarrow G_j$
12. **else if** $TQ_i.empty$ **then**
13. $\pi_{GAP} \leftarrow GAP(\mathcal{T}, S, c, t, d, P)$
14. **if** the cost of π_{GAP} exceeds P
or the makespan of π_{GAP} exceeds d **then**
15. **return** NOT FEASIBLE
16. **else**
17. **return** π_{GAP}
18. **else**
19. $Q.enqueue \leftarrow TQ_i.dequeue$
20. **return** π

In MinCost, the fully polynomial time (FPTAS) GAP [34] computes an $(1 + \epsilon)$ -approximate map, where $0 < \epsilon < 1$, for the GSPs in coalition S . If we give it P as the cost bound and d as the time bound, it computes a mapping π_S for coalition S that approximately satisfies the following constraints:

$$C(\mathcal{T}, S) \leq P \quad (9)$$

$$E(\mathcal{T}, S) \leq d \quad (10)$$

The resulting mapping will be within $(1 + \epsilon)$ of both the deadline and payment.

MinCost uses a simple heuristic to find a mapping π_S with the lowest cost. The heuristic functions as follows: of all the remaining tasks without assignment, task $T \in \mathcal{T}$ is assigned to $G \in S$ if it is the lowest cost assignment and only if the assignment does not result in G exceeding the deadline. For each task T_i , for $i = 1, 2, \dots, n$, there is a priority queue TQ_i that orders tuples $(T_i, G_j, c(T_i, G_j))$, for any i and j , by non-decreasing costs. The priority queue Q contains one tuple for each task ordered by non-decreasing cost. If the lowest cost task T_i in Q does not exceed the deadline for service provider G_j , then T_i is assigned to G_j . Otherwise, the tuple is discarded and then the next lowest cost tuple in TQ_i is dequeued and immediately inserted in Q . If we are unable to determine a mapping by using the list scheduling heuristic, we fallback to the GAP algorithm. GAP always obtains a feasible approximate mapping if such a mapping exists. All the priority queue operations require worst-case time $\Theta(|T||S| \log |S|)$.

A mapping π_S as determined by MinCost may have service providers without any tasks assigned to them. These GSPs are safely removed from the coalition as doing so does not require modifying the mapping or costs. The only impact is increased profits for the active members of the coalition.

Even if a task assignment heuristic is found that takes constant time, coalition formation would still require an expo-

nential amount of time due to the determination of the mappings for all possible coalitions and the necessary backward induction.

We present an example of coalition formation for a five-task program and three GSPs. The order of rule for this example is $\{G_1 G_2, G_2 G_3, G_1 G_3\}$. The costs and execution times are presented in Table I and Table II, respectively. We provide examples to further elucidate the information contained in the tables. From Table I, GSP G_2 incurs 5 units of cost if it executes either T_3 , T_4 , or T_5 and it incurs 2 units of cost for executing either T_1 or T_2 . If G_2 executes the entire program, it incurs cost $2 + 2 + 5 + 5 + 5 = 19$. GSP G_2 executes T_1 in 4 units of times as can be seen from Table II. If G_2 executes the entire program, then the program completes in time $4 + 2 + 4 + 2 + 2 = 14$. Assume that the user has specified a deadline $d = 10$ and a payment $P = 20$. Each GSP computes a mapping and a cost for the mapping for each coalition using MinCost. The mappings for each coalition are given in Table III. The profit per member for each coalition is presented in Table IV. Four strategies are the result of backward induction, but all strategies lead to the same outcome of coalitions $S_1 = \{G_1, G_3\}$ and $S_2 = \{G_2\}$. Service provider G_2 will not offer its resources as it cannot execute the program

before its deadline. Therefore, S_1 will execute the program.

It is likely that GSPs would have differing heuristics when determining coalitions. Some of the heuristics would produce better solutions than the one we offered. Before a link is formed, a party could convince the other by disclosing the structure of a coalition and its associated mapping. This process guides the coalition formation. The sharing of information complicates the formation process though as the GSPs have differing ideas of the preferred coalitions and the solutions given by backward induction may be inaccurate. But this does not hinder coalition formation.

The result of coalition formation is a set $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ of coalitions. Any of the coalitions in \mathcal{S} are able to finish the program by the deadline d and within the given cost constraint. Thus, each of them can potentially execute the program. To resolve this issue, we randomly choose one to execute the program. If we base the decision on further criteria (e.g., minimizing makespan, lowest cost), the GSPs would solve a program similarly to (5) but including additional objectives or constraints further imposed by the criteria.

V. VIRTUAL ORGANIZATION FORMATION FRAMEWORK

In this section we show how the coalition formation model presented above is used to form VOs in grid computing systems. Grid computing systems comprise geographically distributed machines, controlled and operated by independent, autonomous GSPs. We assume that these organizations exhibit welfare maximizing behavior. We propose a framework that brings together users and service providers and supports the VO formation process in order to execute user programs.

We are proposing a resource management system, the Virtual Organization Formation Manager (VOFM), to support coalition formation among GSPs. The VOFM is implementable as a middleware component and it would be supported by the grid implementation. Instances of the VOFM would be replicated across a grid system allowing users and GSPs to interact with a local copy.

The architecture of the VOFM is presented in Figure 3. VOFM manages several lists. VO formation begins by service providers notifying the VOFM of their readiness to perform work. The VOFM records the GSPs on the *GSP ready list*, $\mathcal{G} = \{G_1, G_2, \dots, G_m\}$. A user submits her program, $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$, and the associated program decomposition and execution profiles to the VOFM. Additionally, the user submits the amount she will pay, P , if the program is executed to completion before the deadline, d . The VOFM records \mathcal{T} , P , d and profiles on the *program pending list*, and notifies the GSPs in \mathcal{G} that a program is pending. Each service provider G_j , for $j = 1, 2, \dots, m$, retrieves the program details and estimates the costs $c(T_1, G_j), c(T_2, G_j), \dots, c(T_n, G_j)$ and execution times $t(T_1, G_j), t(T_2, G_j), \dots, t(T_n, G_j)$. These values are reported to the VOFM who records them on the *confirmed list* for program \mathcal{T} . A GSP need not estimate a cost or time for each task and a GSP may choose not to submit any values.

	G_1	G_2	G_3
T_1	3	2	5
T_2	3	2	1
T_3	3	5	3
T_4	2	5	3
T_5	2	5	4

TABLE I

COSTS FOR PROGRAM
 $\mathcal{T} = \{T_1, T_2, \dots, T_5\}$ AND
GSPs $\mathcal{G} = \{G_1, G_2, G_3\}$.

	G_1	G_2	G_3
T_1	2	4	5
T_2	2	2	5
T_3	4	4	5
T_4	4	2	3
T_5	4	2	3

TABLE II

EXECUTION TIMES FOR
PROGRAM $\mathcal{T} = \{T_1, T_2, \dots, T_5\}$
AND GSPs $\mathcal{G} = \{G_1, G_2, G_3\}$.

S	Mapping
$\{G_1\}$	NOT FEASIBLE
$\{G_2\}$	NOT FEASIBLE
$\{G_3\}$	NOT FEASIBLE
$\{G_1, G_2\}$	$T_4, T_5 \rightarrow G_1; T_1, T_2, T_3 \rightarrow G_2$
$\{G_1, G_3\}$	$T_1, T_4, T_5 \rightarrow G_1; T_2, T_3 \rightarrow G_3$
$\{G_2, G_3\}$	$T_1, T_4, T_5 \rightarrow G_2; T_2, T_3 \rightarrow G_3$
$\{G_1, G_2, G_3\}$	$T_4, T_5 \rightarrow G_1; T_1 \rightarrow G_2; T_2, T_3 \rightarrow G_3$

TABLE III

THE MAPPINGS AS DETERMINED BY MINCOST FOR EACH COALITION.

S	Profit
$\{G_1\}$	0
$\{G_2\}$	0
$\{G_3\}$	0
$\{G_1, G_2\}$	3.5
$\{G_1, G_3\}$	4.5
$\{G_2, G_3\}$	2
$\{G_1, G_2, G_3\}$	3.33

TABLE IV

PROFIT FOR EACH GSP IN A COALITION.

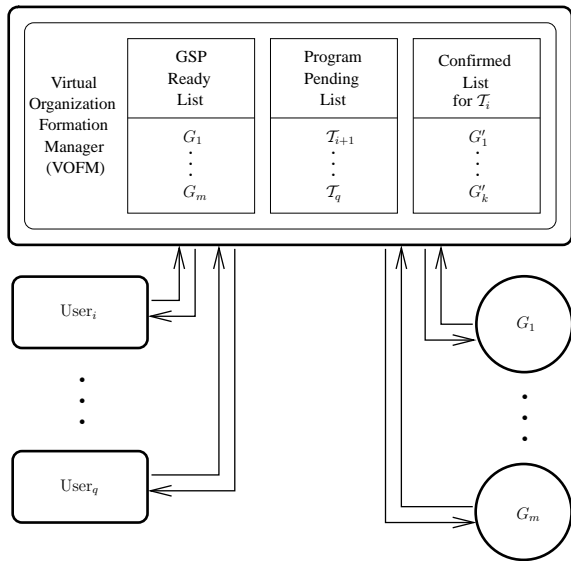


Fig. 3. The architecture of the Virtual Organization Formation Manager (VOFM).

Denote by $\mathcal{G}' = \{G'_1, G'_2, \dots, G'_k\}$ the set of GSPs that have committed. VO formation proceeds among the GSPs in \mathcal{G}' .

In order for a GSP G'_h , for $h = 1, 2, \dots, k$, to choose its preferred VO, it must determine the value of each of the 2^k possible coalitions. Determining the value requires the computation of a mapping. As we discussed earlier, these computations will require an exponential amount of time, and thus significantly reduce the available time for program execution. Additionally, these computations cost money that may not be necessarily recovered. Therefore, the service providers will be induced to employ heuristics. Good heuristics would increase profits and give GSPs economic advantages over their peers. An example of a heuristic that can be employed by GSPs is MinCost which was proposed in Section IV. Consequently, it would be unlikely that the GSPs would share algorithms. But it is to the advantage of a GSP to disclose its preferred VO (coalition) and its associated mapping. This permits the other members of the GSP's preferred VO to evaluate options and if welfare maximizing, come to the same conclusion. Fortunately, a given coalition and mapping are easily verified in linear time by other parties.

There are few strategies for a GSP to determine its preferred VO in a limited amount of time. A GSP may compute its preferred VO by examining all possible VOs consisting of fewer than k GSPs. Another strategy is to randomly generate VOs in which it is a member.

Each service provider transmits its preferred VO and mapping to the VOFM, which dispenses the information to the others. For large k , the number of coalitions with known values will be minute when compared against the total number of possible candidates. This is problematic as coalitional game theory at the very minimum assumes that the coalition values are computable. Thus, we must assign a value to these no-value coalitions. If the GSP's preferences are modeled as *risk*

averse (prefers the outcomes with payoffs that are certain) or *risk neutral* (prefers the expected payoff), it seems natural that they would prefer the known coalitions to the unknown and thus, give the unknown coalitions a value of zero.

The VOFM announces a randomly generated order of rule. At this point, the link formation commences. The GSPs, being rational, perform backward induction and determine the resulting strategy profiles. One by one, the VOFM iterates through the order of rule, providing the state of coalition formation to the link associated GSPs. It then awaits responses from each GSP. As we mentioned earlier, a link is formed only when both GSPs agree; otherwise, the link is not formed. After finishing, the VOFM knows the VO set (coalition set). The VOFM now needs to select the VO that will execute the program. All VOs with no value are removed from consideration. Out of the remaining VOs, the VOFM chooses one randomly. The selected VO executes the program. In the following we present the VO formation protocol which is employed in the VO formation process.

Protocol 1 (VO Formation):

Phase I: Initialization

1. Grid service providers, \mathcal{G} , register their readiness with the VOFM.
2. The user submits her program, \mathcal{T} , to the VOFM.
3. VOFM notifies the GSPs of a pending program.

Phase II: Commitment

1. A GSP G estimates the cost and time for the pending program \mathcal{T} .
2. GSP G transmits costs, $c(T_1, G), \dots, c(T_n, G)$, and time, $t(T_1, G), \dots, t(T_n, G)$, to the VOFM.

Phase III: VO Formation

1. The VOFM dispenses the costs and times to the committed GSPs, \mathcal{G}' .
2. The VOFM announces a randomly generated rule of order.
3. The GSPs perform backward induction, computing the strategy profiles.
4. The VOFM iterates through the order of rule. At each link, the VOFM gives the current state of the VO formation to the link-associated GSPs. The GSPs then announce an accept or reject. If both GSPs accept, the link is formed; otherwise, the link is not formed.
5. The VO set is determined.
6. The VOFM removes from consideration all VOs with zero value.
7. The VOFM randomly selects one of the remaining VO, S .
8. VO S executes the program.

The VOFM facilitates the execution of this protocol by maintaining the required data structures. Additionally, it provides a “meeting place” as it is unlikely that all GSPs would communicate directly with one another.

VI. CONCLUSION

In this article we proposed a model for Virtual Organization (VO) formation among autonomous Grid Service Providers (GSPs). We model VO formation using Myerson's cooperation

structures. A GSP determines the VO that it will form based on its profit. Additionally, the GSPs are assumed to be farsighted. They employ backward induction to investigate what the final coalitions (VOs) will appear as. Backward induction requires computing the value of all possible coalitions, which in itself, requires determining a task mapping. This process requires an exponential amount of time, thus GSPs will use heuristics to determine mappings and coalitions. The heuristics would be considered private to a given GSP, but fortunately, a mapping can be verified in linear time by the other GSPs. This allows them to decide in feasible time on whether to join or not the coalition. Using the above model, we proposed a resource management system, the Virtual Organization Formation Manager (VOFM), that supports VO formation among GSPs. VOFM is implemented as a middleware component with support from the grid implementation.

For future work, we plan to investigate by simulation the VO formation framework presented in this paper. We are interested in comparing the distributed approach presented in this paper to an approach that makes centralized scheduling decisions. In the framework, we assume that providers specify their costs truthfully. We would like to examine ways to extend the model in which providers misreport their costs. Finally, we plan to build a VOFM prototype.

REFERENCES

- [1] I. Foster and C. Kesselman, Eds., *The Grid 2: Blueprint for a New Computing Infrastructure*, 2nd ed. Morgan Kaufmann, 2003.
- [2] R. B. Myerson, "Graphs and cooperation in games," *Mathematics of Operation Research*, vol. 2, pp. 225–229, 1977.
- [3] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *Int. J. Supercomputer Applications*, vol. 15, no. 3, pp. 200–222, 2001.
- [4] I. Foster, N. R. Jennings, and C. Kesselman, "Brain meets brawn: Why grid and agents need each other," in *Proc. of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AA-MAS'04)*, 2004, pp. 8–15.
- [5] T. J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, A. Gray, and N. Fiddian, "CONOISE: Agent-based formation of virtual organizations," *Knowledge-Based Syst.*, vol. 17, pp. 103–111, 2004.
- [6] J. Patel, W. T. L. Teacy, N. R. Jennings, M. Luck, S. Chalmers, N. Oren, T. J. Norman, A. Preece, P. M. D. Gray, G. Shercliff, P. J. Stockreisser, J. Shao, W. A. Gray, N. J. Fiddian, and S. Thompson, "Agent-based virtual organisations for the grid," *Multiagent Grid Syst.*, vol. 1, no. 4, pp. 237–249, 2005.
- [7] A. Akram and R. Allan, "Organization of grid resources in communities," in *Proc. of the 4th international workshop on Middleware for grid computing (MGC'06)*, 2006, p. 20.
- [8] G. Owen, *Game Theory*, 3rd ed. New York, NY, USA: Academic Press, 1995.
- [9] R. J. Aumann and J. H. Dr eze, "Cooperative games with coalition structures," *Int. J. Game Theory*, vol. 3, no. 4, pp. 217–237, 1974.
- [10] R. B. Myerson, "Conference structures and fair allocation rules," *Int. J. Game Theory*, vol. 9, no. 3, pp. 169–182, Nov. 1978.
- [11] M. Klusch and O. Shehory, "Coalition formation among rational information agents," in *Proc. of the 7th European Workshop on Modelling Autonomous Agents in the Multi-Agent World (MAAMAW '96)*, Jan. 1996, pp. 204–217.
- [12] I. M uller, R. Kowalczyk, and P. Braun, "Towards agent-based coalition formation for service composition," in *Proc. of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT '06)*, Dec. 2006, pp. 73–80.
- [13] O. Shehory and S. Kraus, "Task allocation via coalition formation among autonomous agents," in *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI '94)*, 1995, pp. 655–661.
- [14] H. C. Lau and L. Zhang, "Task allocation via multi-agent coalition formation: taxonomy, algorithms, and complexity," in *Proc. of the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '03)*, 2003, pp. 346–350.
- [15] L. He and T. R. Ioerger, "Forming resource-sharing coalitions: a distributed resource allocation mechanism for self-interested agents in computational grids," in *Proc. of the 2005 ACM Symp. On Applied Computing (SAC '05)*, 2005, pp. 84–91.
- [16] H.-J. Zhang, Q.-H. Li, and Y.-L. Ruan, "Resource co-allocation via agent-based coalition formation in computational grids," in *Proc. of the 2nd International Conference on Machine Learning and Cybernetics (ICMLC '03)*, 2003, pp. 1936–1940.
- [17] H.-H. Lee and C.-H. Chen, "Multi-agent coalition formation for long-term task or mobile network," in *Proc. of the International Conference on Computational Intelligence for Modelling, Control, and Automation (CIMCA '06) and International Conference on Intelligent Agents Web Technologies and International Commerce (IAWTIC '06)*, 2006, pp. 52–57.
- [18] T. E. Carroll and D. Grosu, "A strategyproof mechanism for scheduling divisible loads in linear networks," in *Proc. of the 21st IEEE International Parallel and Distributed Processing Symp. (IPDPS '07)*, Mar. 2007.
- [19] N. Nisan and A. Ronen, "Algorithmic mechanism design," *Games and Economic Behaviour*, vol. 35, no. 1/2, pp. 166–196, Apr. 2001.
- [20] D. Abramson, R. Buyya, and J. Giddy, "A computational economy for grid computing and its implementation in the nimrod-G resource broker," *Future Gener. Comput. Syst.*, vol. 18, no. 8, pp. 1061–1074, Oct. 2002.
- [21] K. M. Chao, R. Anane, J. H. Chen, and R. Gatward, "Negotiating agents in a market-oriented grid," in *Proc. of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '02)*, 2002, p. 436.
- [22] J. Gomoluch and M. Schroeder, "Market-based resource allocation for grid computing: A model and simulation," in *Proc. of the 1st International Workshop on Middleware for Grid Computing (MGC '03)*, 2003, pp. 211–218.
- [23] R. Wolski, J. S. Plank, J. Brevik, and T. Bryan, "Analyzing market-based resource allocation strategies for the computational grid," *International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 258–281, Aug. 2001.
- [24] D. Grosu and A. Das, "Auctioning resources in grids: model and protocols," *Concurr. Comput.: Pract. Exper.*, vol. 18, no. 15, pp. 1909–1927, 2006.
- [25] D. Grosu, "AGORA: an architecture for strategyproof computing in grids," in *Proc. of the 3rd International Symp. On Parallel and Distributed Computing (ISPD'04)*, July 2004, pp. 217–224.
- [26] R. J. Aumann and R. B. Myerson, "Endogenous formation of links between players and of coalitions: an application of the shapley value," in *The Shapley Value: essays in honor of Lloyd S. Shapley*, A. E. Roth, Ed. Cambridge University Press, 1988, pp. 175–191.
- [27] J. Yang, A. Khokhar, S. Sheikh, and A. Ghafoor, "Estimating execution time for parallel tasks in heterogeneous processing (HP) environment," in *Proc. of the Heterogeneous Computing Workshop*, 1994, pp. 23–28.
- [28] M. A. Iverson, F.  zg uner, and L. Potter, "Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment," *IEEE Trans. Comput.*, vol. 48, no. 12, pp. 1374–1379, Dec. 1999.
- [29] L. S. Shapley, "On balanced sets and cores," *Naval Research Logistics Quarterly*, vol. 14, pp. 1589–1594, 1967.
- [30] —, "Cores and convex games," *International J. of Game Theory*, vol. 1, pp. 1–26, 1971.
- [31] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.
- [32] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM J. Applied Math.*, vol. 17, no. 2, pp. 416–429, Mar. 1969.
- [33] D. K. Friesen, "Tighter bounds for LPT scheduling on uniform processors," *SIAM J. Comput.*, vol. 16, no. 3, pp. 554–560, 1987.
- [34] P. Efraimidis and P. G. Spirakis, "Approximation schemes for scheduling and covering on unrelated machines," *Theory Comput. Sci.*, vol. 359, no. 1–3, pp. 400–417, Aug. 2006.