VCG-based Truthful Mechanisms for Social Task Allocation

Yingqian Zhang and Mathijs de Weerdt

Delft University of Technology, The Netherlands
{Yingqian.Zhang, M.M.deWeerdt}@tudelft.nl

Abstract. In many applications of the task allocation problem such as peer-to-peer and grid computing, and virtual organizations, the (social or business) relations between the participating agents play an important role, and thus they should be taken into account. Furthermore, in such applications, agents providing the resources usually act self-interested. This paper therefore studies the problem of finding truthful mechanisms for these kinds of social task allocation problems.
In this paper we give on the one hand an optimal mechanism and model the problem as an integer linear program (ILP), and on the other hand a polynomial-time approximation by splitting the problem into smaller sub-problems, each of which is solved optimally. We show that both mechanisms are truthful.
The optimal mechanism may take exponential time for some instances, and in theory, the quality of the approximation is not guaranteed. However, we show experimentally that for problem instances where the social network has the small-world property, the quality of the results for the approximation is quite good, due to the fact that the division into subproblems uses the locality of tasks in the social network.

1 Introduction

The task (and resource) allocation problem (TAP) has been extensively investigated in recent years [8, 17, 18]. With the increasing popularity of the Internet as a global platform for computation, many interesting applications of this problem have emerged. For example, in peer-to-peer and grid computing, resources and tasks are allocated to different agents. The difference from earlier problem settings is that these agents interact within a (partially) connected (social) network, just like in business applications, where preferential partner selection and interaction is very common. Therefore, we are interested in the study of the task allocation problem in agent social networks (STAP) [4].

In our previous work [4] we assumed agents to be cooperative. In this paper we study the STAP as a mechanism design problem, taking into account the incentives of agents. In the problem setting for this paper, agents receive a fair share of the utility of a task they contribute to. Consequently, it is more interesting for an agent to submit its resources to a task with a high utility. Our main goal now is to develop so-called truthful mechanisms in which agents cannot be better off by lying about the resources they have available.

We briefly describe the STAP as follows. A set of agents is connected by a partial (social) network. Each agent has a limited amount of resources of different types at its disposal. There also is a set of tasks to be done. Each task requires some resources, has

a fixed benefit, and is located at a certain agent, called a *manager*. We only allow neighboring agents to supply the resources for a task. These agents are called *contractors*. The problem is, given the declared resources of the contractors, to find out which tasks to execute, and which resources of which neighboring contractors to use for these tasks, such that the *total benefit* of the allocated tasks is maximized.

In contrast to some of the other task allocation and mechanism design problems, we assume in this problem setting that all resources have been paid for already, or that the costs for using them are negligible. This situation occurs in for example grid computing, but also in many companies, because the hardware and/or the employees need to be paid anyway. Also without costs, the social task allocation problem (STAP) is NP-complete [4].

The efficient task allocation can only be found if the information of the available resources among agents is correctly known, which requires agents to report their resources truthfully. However, in many situations—especially where multiple organizations or companies are involved—agents are often self-interested, and they may lie about their private information in order to maximize their utility. Therefore, incentivizing self-interested agents to report their private information truthfully is crucial to the performance of task allocation algorithms. Surprisingly, this issue has not yet gained much attention in the context of the TAP. Most of the existing work focuses on the development of computational models in cooperative settings. Only a few other works consider the influence of agents' selfish behavior in task allocation [6, 10, 15], among which Kraus et al. [6] study the case where the tasks need to be executed by self-interested agents under time constraints. They focus on (stable) strategies for distribuing revenues fairly among agents. Manisterski et al. [10] show impossibility results for completing tasks by non-cooperative agents, and Nisan and Ronen [15] propose a truthful task allocation mechanism, called *MinWork*, for the problem of minimizing the task completion time. In this paper, we address both the *computational part*—to develop an *efficient* allocation algorithm, and the *game theoretical aspect*—to define a *truthful* mechanism, of the task allocation problem in social networks.

The private information in STAP is the set of available resources of each contractor. We assume in this paper that only the contractor agents strategize about their private information. To avoid manipulation of the contractors, we aim to design a truthful mechanism by adding a payment function to reward agents for the use of their resources. Nisan and Ronen [15] showed that the truthfulness of agents can be guaranteed by so-called VCG-based mechanisms (see Definition 7) if the mechanism is able to compute the optimal solution. Since many interesting optimization problems are intractable, they showed an alternative way to achieve a truthful *VCG-based mechanism* by replacing the exact algorithm with an approximation [16]. However, they showed that for combinatorial auctions all reasonable VCG-based mechanisms are *not* truthful. Moreover, for a certain class of minimization problems (*cost minimization allocation problem*), any truthful VCG-based mechanism is either optimal or can lead to degenerate results, i.e. for any approximation there are instances in which the result can be arbitrarily far from the optimal solution. Their result suggests that for many NP-hard problems, developing good polynomial-time VCG-based truthful mechanisms is not a trivial problem.

Combinatorial auctions (CAs) have become the center of attention for studying the impact of using an approximation algorithm for the design of a truthful mechanism, e.g. by imposing additional restrictions on the problem domain [1, 7, 11]. For example, Lehmann et al. [7] restricted the preferences of each bidder agent to a single bundle of items, and then introduced efficient greedy mechanisms that are truthful. In CAs,

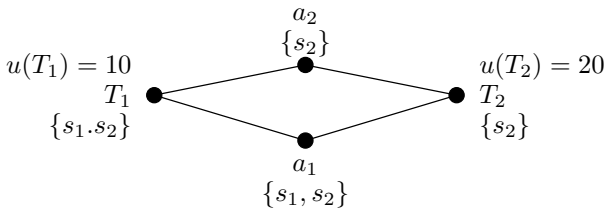

Fig. 1. For some allocation algorithms, agent a_2 will be better off by underreporting its resources.

the preference v of the bidders is assumed to be *monotone* (or called "free disposal" [2]), i.e., if a bundle of items S is a subset of another bundle S', then $v(S) \leq v(S')$. However, this is not the case in the problem under consideration in this paper. As a simple example, consider the case where there are two tasks T_1 and T_2. Task T_1 with the benefit of 10 requires resources s_1 and s_2 to complete. T_2 only needs resource s_2 yet with a higher benefit of 20. A contractor agent a_1 connects to both tasks, and it has required resources from both tasks. Thus it can be allocated to either T_1 or T_2. As agents' valuations relate to the tasks it is assigned to (it will be defined formally in Section 2.1), a_1's valuation could be 20 by declaring $\{s_2\}$ and 10 by declaring $\{s_1, s_2\}$ (see Figure 1). Clearly, its valuation is not monotonically increasing with the "better" declared type, and thus the domain we study is different from that of CAs. Since the characteristic of truthfulness depends strongly on the domains of the valuations of the agents [14], the existing results of truthfulness for CAs cannot directly be applied to the STAP.

In remainder of this paper, we first recapitulate the STAP introduced in [4], then define the mechanism design problem for STAP formally. Next, in Section 3 we propose an exact, truthful mechanism with an optimal allocation algorithm. We then introduce our main contribution, i.e., a polynomial-time truthful mechanism (in Section 4). Although the quality of this approximation cannot be guaranteed, in Section 5 we show experimentally that for artificially generated small-world networks the results of the approximation can be quite good, suggesting that instances of the STAP in practice having this small-world property can also be solved to a satisfactory level.

2 Social task allocation problems

In this section we define the social task allocation problem in a cooperative setting, followed by a mechanism design problem for self-interested agents.

2.1 Cooperative social task allocation

Let $\mathcal{A}$ denote a set of agents, which need resources to complete tasks. Let $R = \{r_1, \ldots, r_l\}$ denote the collection of the resource types available to the agents $\mathcal{A}$. Each agent $i \in \mathcal{A}$ controls a fixed amount of resources for each resource type in R, which is defined by a resource function: $s_i : R \to \mathbb{N}$. Finally, we assume agents are connected by a *social network*.

Definition 1 (Social network). *An agent social network $SN = (\mathcal{A}, AE)$ is an undirected graph, where vertices $\mathcal{A}$ are agents, and each edge $(i, j) \in AE$ indicates the existence of a social connection between agents i and j.*

Suppose a set of tasks $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ arrives at such an agent social network. Each task $t \in \mathcal{T}$ is then defined by a tuple $\langle U(t), req(t), loc(t)\rangle$, where $U(t)$ is the benefit gained if task t is accomplished, and the function $req(t) : R \to \mathbb{N}$ specifies the amount of resources required for the accomplishment of task t. Furthermore, a location function $loc : \mathcal{T} \to \mathcal{A}$ defines the locations (i.e., agents) of the tasks in the social network. An agent i that is the location of a task t, i.e. $loc(t) = i$, is called the *manager* of task t. The exact assignment of resources to tasks is defined by a *task allocation*.

Definition 2 (Task allocation). *Given a set of tasks $\mathcal{T} = \{t_1, \ldots, t_n\}$ and a set of agents $\mathcal{A}$ in a social network SN, a **task allocation** is a mapping $o : \mathcal{T} \times \mathcal{A} \times R \to \mathbb{N}$. A **valid** task allocation in SN must satisfy the following constraints:*

- *A task allocation must be correct. Each agent $i \in \mathcal{A}$ cannot use more than its available resources, i.e. for each $r \in R$, $\sum_{t \in \mathcal{T}} o(t, i, r) \leq s_i(r)$.*
- *A task allocation must be complete. For each task $t \in \mathcal{T}$, either all allocated agents' resources are sufficient, i.e. for each $r \in R$, $\sum_{i \in \mathcal{A}} o(t, i, r) \geq req(t)(r)$, or t is not allocated, i.e. $o(t, \cdot, \cdot) = 0$.*
- *A task allocation must be maximal, i.e., no more tasks can be allocated.*
- *A task allocation must obey the social relationships. Each task $t \in \mathcal{T}$ can only be allocated to agents that are (direct) neighbors of agent $loc(t)$ in the social network SN. Each such agent that can contribute to a task is called a **contractor**.*

The set of all valid task allocations is denoted by $\mathcal{O}$.

We write T_o to represent the tasks that are fully allocated in o. The benefit of o is then the sum of the benefits of each task in T_o, i.e., $U_o = \sum_{t \in T_o} U(t)$. Note that we do not include costs for resources, since we assume that the resources have already been paid for. Our only goal is to allocate these resources as efficiently as possible. We thus define the *efficient task allocation* as follows.

Definition 3 (Efficient task allocation). *We say a task allocation o^* is **efficient** if it is valid and U_{o^*} is maximized, i.e., $o^* = \arg\max_{o \in \mathcal{O}} U_o$.*

The social task allocation problem is then defined as follows.

Definition 4 (Social task allocation problem). *Given a set of agents $\mathcal{A}$ that are connected by a social network $SN = (\mathcal{A}, AE)$, a finite set of tasks $\mathcal{T}$ need to be allocated to the agents. The goal of this **social task allocation problem** (STAP) is to find the efficient task allocation o^*.*

2.2 Mechanism design for STAP

The efficient task allocation can only be found if the agents report their available resources truthfully. Since we cannot always rely on the agents to be honest, we treat the problem as a mechanism design problem so that every agent is incentivized to report its true resources, no matter what strategies other agents use.

We give a brief summary of the relevant mechanism design concepts below, but for a more elaborate introduction please see [14]. In a mechanism design setting, we provide a method that determines an outcome, i.e., a valid task allocation $o \in \mathcal{O}$, given the inputs (called strategies) from the contractor agents and the public knowledge. Typical assumption for a mechanism design problem is that some of the information is private, in our case, those of the contractor agents.

This private information or *type* s_i of a contractor agent is the description of the resources it has available, i.e., $s_i : R \to \mathbb{N}$. The set of all such functions is called its type space S. The type space of all m agents is defined by S^m. We use $\mathbf{s} = (s_1, \ldots, s_m) \in S^m$ to denote the *type profile* of the agents. We sometimes denote $\mathbf{s}$ by $(s_i, \mathbf{s}_{-i})$, where $\mathbf{s}_{-i}$ denotes the types of all contractors except i.

The set of possible inputs of an agent for the mechanism is called its strategy space. The revelation principle [12] says that for any coordination mechanism any equilibrium can also be achieved by a truthful direct-revelation mechanism. A direct revelation mechanism is one where the strategy space of the agents is exactly their type space. In our study we therefore define the strategy space $A = S$, and search for a mechanism that is truthful (see Definition 6).

Besides the strategies of the contractor agents, part of the input for the mechanism consists of public information. In our case, this is a social network and a set of tasks. We use Z to denote this public parameter space of the social task allocation problem. Each $z \in Z$ is a tuple $(SN, \mathcal{T})$.

When the mechanism receives inputs $\mathbf{a} = (a_1, \ldots, a_m) \in A$ (called a *strategy profile*), it selects an allocation $o = O(z, \mathbf{a})$ with some allocation algorithm O. In addition, the mechanism computes payments $(p_1(z, \mathbf{a}), \ldots, p_m(z, \mathbf{a}))$ for all contractor agents. The result for agent i, called its *utility*, is the sum of the *valuation* v_i that i gets from the resulting allocation o with its type a_i and the payment it receives from the mechanism:

$$u_i(\mathbf{a}) = v_i(a_i, o) + p_i(z, \mathbf{a}).$$

In the STAP, we define the valuation of agent i as its fair share of the utilities of the tasks it helped to fulfill. For this we define the *efficiency* e of a task t by dividing the utility of t by the total number of required resources for t: $e(t) = \frac{U(t)}{\sum_{r \in R} req(t)(r)}$. This efficiency of a task has no other relation to the efficiency of a task allocation (Definition 3 than that it expresses a heuristic for obtaining such an efficient task allocation. An agent then receives for each resource it is contributing a fair share (the efficiency) of the task it is contributed to.

$$v_i(a_i, o) = \sum_{t \in T_o} \sum_{r \in R} \min\{o(t, i, r), a_i(r)\} \cdot e(t). \tag{1}$$

The utility u_i is what agent i aims to maximize. The *social welfare* $W(o)$ of the system is then the sum of the valuations of the contractors in the allocation o, i.e., $W(o) =$

$\sum_{i=1}^{m} v_i(a_i, o)$. We use this to define the mechanism design problem for social task allocation formally.

Definition 5 (Mechanism design for STAP). *Given the parameter space Z, the type space S, and the strategy space A, the* mechanism design problem for STAP *is to find a mechanism $\mathcal{M} = (O, p)$ that consists of an* allocation function $O : Z \times A \to \mathcal{O}$*, and a* payment function $p_i : Z \times A \to \mathbb{R}$ *such that the selected output $o \in \mathcal{O}$ maximizes the total social welfare $W(o)$.*

A mechanism is *efficient* if it maximizes the social welfare. Such a mechanism design problem is called a *utilitarian mechanism design problem* [16].

We first show that the goal of this mechanism design problem is aligned with that of the STAP. We re-write the objective function of the STAP for this purpose using the definition of a valid allocation o as follows:

$$\begin{aligned} U(T_o) &= \textstyle\sum_{t \in T_o} U(t) = \sum_{t \in T_o} \sum_{r \in R} req(t)(r) \cdot e(t) \\ &= \textstyle\sum_{i \in \mathcal{A}} \sum_{t \in T_o} \sum_{r \in R} o(t, i, r) \cdot e(t) \\ &= \textstyle\sum_{i=1}^{m} v_i(a_i, o) = W(o) \end{aligned}$$

Thus, when an algorithm for STAP gives the optimal solution, it also outputs the optimal social welfare for the mechanism, if agents report their private information truthfully, i.e., if the mechanism is *truthful*.

Definition 6 (Truthful). *Given an output algorithm O, a mechanism is* truthful *if $A = S$, and for any parameter $z \in Z$, for any strategy profile $\mathbf{a} \in A^m$, for any agent i with type $s_i \in S$ it holds that*

$$\begin{aligned} & u_i(s_i, \mathbf{a}_{-i}) = v_i(s_i, O(z, s_i, \mathbf{a}_{-i})) + p_i(z, s_i, \mathbf{a}_{-i}) \\ \geq\ & u_i(a_i, \mathbf{a}_{-i}) = v_i(s_i, O(z, a_i, \mathbf{a}_{-i})) + p_i(z, a_i, \mathbf{a}_{-i}) \end{aligned}$$

Informally, agent i is always better off by revealing its true private type s_i to the mechanism, no matter what strategies other agents play. Truthful mechanisms can be achieved with carefully designed payment functions, such as VCG (Vickrey-Clarke-Groves) payments [19, 3, 5]. It has been shown that truthfulness can be guaranteed by a VCG payment if the mechanism is able to compute the optimal solution [16]. Furthermore, in some cases VCG-based mechanisms are the only possible truthful mechanisms [14].

Definition 7 (VCG mechanism [16]). *A mechanism $\mathcal{M} = (O, p)$ belongs to the VCG family if:*

1. *The allocation function $O(z, \mathbf{a})$ maximizes the total welfare according to $\mathbf{a}$, i.e. for all $\mathbf{a}$, $O(z, \mathbf{a}) \in \arg\max_{o \in O(z, \mathbf{a})} W(o)$.*
2. *The payment of agent i is calculated according to the VCG formula, i.e. $p_i(z, \mathbf{a}) = -v_i(a_i, O(z, \mathbf{a})) + W(O(z, \mathbf{a})) + h^{\mathbf{a}_{-i}}$, where $h^{\mathbf{a}_{-i}}$ is an arbitrary function of $\mathbf{a}_{-i}$.*

Nisan and Ronen [16] take into account the *computational complexity* while dealing with the incentive compatibility. A mechanism is an *exact* mechanism if its allocation algorithm maximizes the objective function. However, for many NP-hard problems, computation is intractable in order to maintain the truthfulness. Therefore, they introduced a class of *VCG-based* mechanism, where the optimal algorithm is replaced by a sub-optimal, polynomial-time algorithm. A *polynomial time mechanism* is one where both the allocation and the payment can be computed in polynomial time.

2.3 Hardness results

The following complexity results on STAP were proven in our previous work [4].

Theorem 1. *Given l resource types, a set of tasks $\mathcal{T}$, and a set of agents $\mathcal{A}$ with a social network SN, the problem of deciding whether a task allocation o with benefit more than K exists is* NP*-complete.*

Let the number of neighbors of each agent in the social network SN be bounded by Δ for $\Delta \geq 3$. Computing the efficient task allocation given such a network is NP*-complete. In addition, it is not approximable within Δ^{ε} for some $\varepsilon > 0$.*

In this paper we approach the mechanism design problem for STAP from two directions. First, we come up with a truthful mechanism that is guaranteed to be optimal, but can take exponential time. Then we propose an algorithm that takes polynomial time, but does not have any hard guarantees on the quality of the solutions. Later we investigate these disadvantages experimentally.

3 An exact mechanism for STAP

In this section, we first introduce an optimal allocation algorithm, and then a VCG payment scheme to incentivize agents to report their true types.

3.1 The optimal task allocation algorithm

The optimal task allocation algorithm should deal with the restrictions posed by the social network. For this NP-complete problem we used a straightforward translation to an integer linear programming (ILP) problem and the GNU Linear Programming Kit [9] to solve this problem. For the ILP formulation we introduce two types of variables: the binary variables $y_j \in \{0, 1\}$ for $1 \leq j \leq n$ describe whether or not task j is allocated, and the integer variables $\forall_{1 \leq j \leq n, 1 \leq i \leq m, 1 \leq k \leq l} x_{ijk}$ denote the amount of resources of type k agent i supplies to task j. The ILP formulation then looks as follows.

$$\texttt{Maximize} \quad \sum_{j=1}^{n} y_j \cdot U(t_j)$$

subject to having sufficient resources of each type for each chosen task from the neighboring agents, i.e.

$$\forall_{1 \leq j \leq n} \forall_{1 \leq k \leq l} \quad \sum_{\{i \in [1,m] | (i, loc(t_j)) \in AE\}} x_{ijk} \geq y_j \cdot req(t_j)(r_k),$$

and not using more resources than there are available, i.e.

$$\forall_{1 \leq i \leq m} \forall_{1 \leq k \leq l} \quad \sum_{j=1}^{n} x_{ijk} \leq rsc(i)(r_k).$$

This optimal algorithm (OPT) is exponential in the number of variables, i.e., the number of tasks, agents, and the resource types.

3.2 An exact truthful mechanism

Our mechanism is developed based on the VCG mechanism, which has some nice properties such as being *efficient* and *incentive compatible*.

Definition 8 (M_{OPT} **for** STAP**).**

- ***Task allocation algorithm*** OPT*: Let* $z = (SN, \mathcal{T})$ *be an instance of* STAP*. First the mechanism center announces a set of tasks* $\mathcal{T}$*—required resources (type and demand), utilities and locations—that need to be allocated to all contractor agents. Next the contractors declare their types* $\mathbf{a}$ *to the center. The center then finds the efficient allocation* $o = \mathsf{OPT}(z, \mathbf{a})$ *using the ILP translation.*
- ***Payment function*** p^{OPT}*: The payment of an agent is its marginal contribution to the society, i.e.,*

$$p_i^{\mathsf{OPT}}(z, a_i, \mathbf{a}_{-i}) = -v_i(a_i, o) + W(o) - W(o_{-i}) \tag{2}$$

where $o_{-i} = \mathsf{OPT}(z, \mathbf{a}_{-i})$ *is the efficient allocation computed by* OPT *without* i*'s participation.*

An agent has no incentive to not fully state its available resource types and amounts, since the utility of an agent is its marginal contribution to the society: suppose an agent states less resources, the total number of allocated tasks will be no more than when it fully states its available resources. Thus the marginal contribution of the agent to the social welfare would then be no more, since the resulting efficient allocation has lower utilities. Therefore the agent will derive a lower utility due to its incomplete report.[1]

Theorem 2. *The mechanism* $\mathcal{M}_{\mathsf{OPT}} = (\mathsf{OPT}, p^{\mathsf{OPT}})$ *is a truthful and efficient mechanism, where agents always receive non-negative utilities by participating in the game, i.e., agents are individually rational (IR). In addition, the mechanism gives no payment to agents that do not get any allocated tasks.*

Proof. Let s_i be the true type of agent i and a_i be any other type. Given a problem instance $z = (SN, \mathcal{T})$, let the resulting allocations be denoted by $o = \mathsf{OPT}(z, s_i, \mathbf{a}_{-i})$, and $\hat{o} = \mathsf{OPT}(z, a_i, \mathbf{a}_{-i})$, respectively, and let $o_{-i} = \mathsf{OPT}(z, \mathbf{a}_{-i})$ be the efficient allocation computed by OPT without i's participation.

First we prove truthfulness by showing that agent i receives always more or equal utility by declaring its true type s_i instead of a_i. This difference δ is calculated as follows (using Equation 2):

$$\begin{aligned}
\delta &= u_i(s_i, \mathbf{a}_{-i}) - u_i(a_i, \mathbf{a}_{-i}) \\
&= v_i(s_i, o) + p_i^{\mathsf{OPT}}(z, s_i, \mathbf{a}_{-i}) - (v_i(a_i, \hat{o}) + p_i^{\mathsf{OPT}}(z, a_i, \mathbf{a}_{-i})) \\
&= W(o) - W(o_{-i}) - (W(\hat{o}) - W(o_{-i})) \\
&= W(o) - W(\hat{o})
\end{aligned}$$

Since the optimal allocation will not get worse by adding more resources in the system, $W(o) - W(\hat{o}) \geq 0$. So, $\delta \geq 0$, i.e., agents obtain higher utilities by truthfully reporting

[1] In this paper we do not address the case that an agent overstates its resources. Such misreports can be easily avoided by a separate payment calculation and payment transfer stage.

their types. Since all agents declare their true resources to the mechanism, the allocation algorithm OPT outputs the optimal allocation.

We now show using a similar reasoning that agents are individually rational. Agent i's utility is computed by

$$\begin{aligned} u_i(s_i, a_{-i}) &= v_i(s_i, o) + p_i^{\mathsf{OPT}}(z, s_i, \mathbf{a}_{-i}) \\ &= W(o) - W(o_{-i}). \end{aligned}$$

When agent i joins, there are more resources available for the task allocation, thus, the system will always result in a better (or the same) allocation using the optimal algorithm OPT. Therefore, $W(o) \geq W(o_{-i})$ and thus $u_i(s_i, a_{-i}) \geq 0$, so agent i is guaranteed to receive non-negative utility when joining the allocation game.

If agent i is not in the resulting allocation o by OPT, we have $v_i(s_i, o) = 0$, and $W(o) = W(o_{-i})$. Therefore, we have $p_i^{\mathsf{OPT}}(z, a_i, \mathbf{a}_{-i}) = -v_i(a_i, o) + W(o) - W(o_{-i}) = 0$. That is, agent i receives no payment. □

We now show that $\mathcal{M}_{\mathsf{OPT}}$ belongs to the class VCG mechanisms, based on Definition 7. Since the allocation function OPT outputs the optimal solution, $\mathsf{OPT}(z, \mathbf{a})$ maximizes the total welfare. In addition, a VCG payment scheme should fit the form $p_i(z, \mathbf{a}) = -v_i(a_i, O(z, \mathbf{a})) + W(O(z, \mathbf{a})) + h^{\mathbf{a}_{-i}}$. Let $h^{\mathbf{a}_{-i}} = W(\mathsf{OPT}(z, \mathbf{a}_{-i}))$, then it follows from Equation 2 that P^{OPT} is a VCG payment function.

Theorem 2 showed that the mechanism is efficient, truthful, and individually rational. One may wonder how much the mechanism would pay the agents, and whether or not the mechanism is *budget balanced* (BB).

Definition 9 (Budget balanced). *Since there is no cost defined in our problem, a mechanism $\mathcal{M}$ is called* budget balanced *for social task allocation problem if the total payment to the contractors is 0, i.e., $\sum_{i=1}^{m} p_i = 0$.*

Unfortunately, a classic result from Myerson et al. [13] showed that it is impossible to achieve budget balance with efficiency and IR from a truthful mechanism. A budget deficit is inevitable in $\mathcal{M}_{\mathsf{OPT}}$. We now briefly analyze how much the mechanism pays out. The total payment to the agents is calculated by:

$$\sum_{i=1}^{m} p_i^{\mathsf{OPT}} = \sum_{i=1}^{m} (W(o) - W(o_{-i})) = mW(o) - \sum_{i}^{m} W(o_{-i}).$$

where m is the number of contractor agents and $W(o)$ is the maximal social welfare. When no agents are *critical*, in the sense that with the absence of any single agent i, the utility of the resulting allocation $W(o_{-i})$ still equals to that of the optimal one, $W(o)$, the total payment that the mechanism gives out is 0; the worst case is when $W(o_{-i}) = 0$ for all i, i.e., all agents hold "unique" resources such that any agent's absence will result in no allocated task at all. In that case the mechanism has to pay the agents in total m times of the social welfare in order to maintain their truthfulness. Therefore, we have:

$$0 \leq \sum_{i=1}^{m} p_i^{\mathsf{OPT}} \leq mW(o).$$

4 A truthful polynomial-time mechanism for STAP

The exact mechanism for STAP, $\mathcal{M}_{\mathsf{OPT}}$, is truthful and efficient. However, it takes exponential time to compute the allocation and the payment. Obviously, this is not feasible when the problem size is large. In this section, we develop a truthful polynomial-time mechanism by splitting the problem into sub-problems that each can be solved optimally in polynomial time.

Our main queston here is how to create sub-problems in such a way that the resulting mechanism is still truthful and the quality of the resulting solutions is still quite good. To achieve truthfulness, we should divide the problem in a way that does not depend on the declared types of the agents. We would like the algorithm to be computationally feasible. Since in most applications, the problem size tends to depend on the number of agents and the number of tasks (for a fixed number of resource types) the algorithm should be polynomial in at least m and n.

4.1 A cluster-based allocation algorithm

The idea of the cluster-based algorithm is to find a *partitioning* of the given social network into several disjoint subnetworks (or clusters) so that each subnetwork is small enough to be solved by the optimal algorithm in polynomial time.

Our approach is to cluster tasks based on the idea of *similarity*. This similarity-based heuristic comes from the consideration that tasks which connect to the same contractor agents should end up in the same cluster, because these interactions between tasks are the core of our problem. We would like to keep as many of these interactions in the sub-problems to maintain a certain level of quality.

Given a STAP with n tasks and m agents, we divide the graph so that each cluster C contains at most $\log n$ tasks and $\log m$ agents when n and m are large. For smaller numbers of tasks we limit the cluster size by a fixed number c. We start with the most efficient task, which has the highest utility for using resources. The tasks are divided in such a way that those in one cluster are more related to each other than to tasks outside of the cluster. The similarity of a task t to a cluster C is measured by the number of mutual contractor agents they are connected to (see also Algorithm 1): $\text{sim}(t, C) = \sum_{t' \in C} |\text{mutual}(t, t')|$, where

$$\text{mutual}(t, t') = \left\{ i \in \mathcal{A} \;\middle|\; \begin{array}{l} ((i, loc(t)) \in AE \text{ or } i = loc(t)) \text{ and} \\ ((i, loc(t')) \in AE \text{ or } i = loc(t')) \end{array} \right\}$$

After we find a cluster C with at most $\max\{c, \log n\}$ tasks, we limit the number of contractor agents in C by keeping only the agents which have most connections to the tasks in C. Each contractor agent can belong to at most one cluster. To distribute the contractor agents evenly over the $\frac{n}{\log n}$ clusters, we add exactly q contractor agents to each cluster, where $q = \min\left\{\log(m), \frac{m \cdot \max\{c, \log n\}}{n}\right\}$. In this way, the size of each cluster k is bounded by $\max\{c, \log n\} + q$. When the partitioning is completely done, we calculate the optimal solution by ILP described in Section 3 for each cluster separately.

Algorithm 1 Cluster-based algorithm CLS.

Input: a set of agents $\mathcal{A}$, tasks $\mathcal{T}$, a network $SN = (\mathcal{A}, AE)$ and a cluster size c.
Output: a task allocation and its value.

1. Sort all tasks $t \in \mathcal{T}$ in descending order of their efficiencies $e(t)$ in a list L.
2. Select the first task t from L, assign $C = \{t\}$, and then,
   - Compute for every other task t' in the network its similarity $\text{sim}(t', C)$ to the task(s) in C.
   - Add the task t' with largest similarity value to C: $C \leftarrow C \cup \{t'\}$.
   - Repeat the above two steps until $|C| = \max\{c, \log n\}$ or $L = \emptyset$.
3. Based on SN, include the q contractor agents with most connections to C.
4. Remove the tasks in C from L. Remove the agents in C from the network SN.
5. Compute the task allocation for C using the optimal algorithm OPT.
6. Repeat from step 2 to find another cluster until there is no more task left in this list.

Proposition 1. *Given a* STAP *with n tasks, m agents, l resource types, and the degree of the network bounded by Δ, the cluster-based algorithm* CLS *is polynomial in m, n, Δ, and exponential in l. When the number of resource types l is bounded by a constant,* CLS *is a polynomial-time algorithm.*

Proof. In this proof we show that the asymptotic run-time is polynomial; we therefore ignore the case that $\log n < c$ in the following. We first show that partitioning the network using the similarity-based heuristic takes polynomial time. Sorting the set of tasks takes $\mathrm{O}(n \log n)$. For a cluster C, computing the similarity of one task to C is bounded by $\mathrm{O}(\log n\Delta)$. This similarity is calculated for at most $\mathrm{O}(n)$ tasks. This is repeated for each task to be added to the cluster. Thus, determining one cluster takes $\mathrm{O}\left(n(\log n)^2\Delta\right)$. As there are in total at most $\mathrm{O}\left(\frac{n}{\log n}\right)$ clusters to be found, the computation time for the partition algorithm is: $\mathrm{O}\left(n \log n + n^2 \log n\Delta\right) = \mathrm{O}\left(n^2 \log n\Delta\right)$.

Recall that ILP runs exponentially in n, m, and l. Now in each cluster, the size of the input for the ILP is reduced by restricting the number of tasks and agents. More precisely, in each cluster, the number of variables in ILP is $\mathrm{O}(\log(n)\log(m)l)$, therefore, computing the solution for each cluster using ILP takes: $e^{\mathrm{O}(\log(n)\log(m)l)} = \mathrm{O}\left(m \cdot n \cdot 2^l\right)$.

When the number of resource types l is bounded by a constant, the total computation time of CLS is: $\mathrm{O}\left(n^2 \log n\Delta\right) + \mathrm{O}\left(\frac{n}{\log n}\right) \cdot \mathrm{O}(mn) = \mathrm{O}\left(n^2(\log n\Delta + \frac{m}{\log n})\right)$. □

The cluster-based algorithm can be used to define a truthful mechanism.

4.2 A polynomial-time truthful mechanism

We define a polynomial-time truthful mechanism as follows.

Definition 10 (A cluster-based mechanism $\mathcal{M}_{\mathsf{CLS}}$). *A cluster-based mechanism $\mathcal{M}_{\mathsf{CLS}} = (\mathsf{CLS}, P_{\mathsf{CLS}})$ works as follows.*

– *Let $z = (SN, \mathcal{T})$ be an instance of* STAP. *First, the mechanism partitions the agent network according to Algorithm 1, Steps $1-4$.*
– *After one cluster C_j is formed, the agents in this cluster are asked to submit their private types $\mathbf{a}$ to the mechanism. Based on the declared types, the mechanism uses the task allocation algorithm* CLS *to get the optimal allocation o^j of this cluster (Algorithm 1, Step 5).*
– *A payment function p^{CLS} calculates the payment to the agents in each cluster C_j based on the same payment function as used by the* OPT *mechanism (Definition 8), i.e., $p_i^{\mathsf{CLS}}(z, a_i, \mathbf{a}_{-i}) = -v_i(a_i, o^j) + W(o^j) - W(o^j_{-i})$.*

The mechanism calculates the allocation and the payment for each cluster. Agents are asked to submit their types after the clusters are formed. Therefore, agents' private types will not influence the partitioning process, i.e. they cannot manipulate in order to enter "good" clusters. When the mechanism divides the network into several clusters, it uses only the public information—the network structure and the information of the tasks, no private information of agents is involved in this stage, and each agent can only belong to one cluster. Indeed, we will show this is the key fact which ensures that the mechanism is truthful even when a sub-optimal allocation algorithm (CLS) is used.

Theorem 3. *The mechanism $\mathcal{M}_{\mathsf{CLS}} = (\mathsf{CLS}, p^{\mathsf{CLS}})$ is a polynomial-time truthful mechanism which is individual rational and gives no payment to agents that do not get any allocated tasks.*

Proof. The cluster-based algorithm (Algorithm 1) removes some edges between agents and tasks, and divides the network into disjoint clusters. Given a problem instance z, as a result of partitioning, the set of "allowable" allocations, given the type space of agents $\mathbf{a}$, is restricted to $\Omega = \{\omega_1, \ldots, \omega_k\}$ where $\omega_j = \{o^j_1, o^j_2, \ldots\}$ denotes the set of allowable allocations in cluster C_j. Note that Ω is a subset of the allowable outputs $\mathcal{O}$ in the exact mechanism: $\Omega \subseteq \mathcal{O}$. Each agent is only in one cluster after partitioning. We now prove that for each cluster, truth-telling is always in the best interests of all agents. For this we use a similar proof as for the exact mechanism (Theorem 2).

Given an agent i in the cluster C_j, we define the true type of i by s_i and any other type by a_i. Let $o^j = \mathsf{CLS}(z, s_i, \mathbf{a}_{-i})$, and $\hat{o}^j = \mathsf{CLS}(z, a_i, \mathbf{a}_{-i})$, $o^j, \hat{o}^j \in \omega_j$, respectively. The difference δ of the utility that i will receive by declaring s_i and a_i is:

$$\delta = u_i(s_i, a_{-i}) - u_i(a_i, a_{-i}) = W(o^j) - W(\hat{o}^j).$$

Since we use the optimal algorithm OPT to find the optimal allocation in every cluster, o^j is the best allocation over ω_j, i.e., $o^j = \arg\max_{o'^j \in \omega_j} W(o'^j)$. Therefore, $W(o^j) - W(\hat{o}^j) \geq 0$, i.e., agents are better off by report their types truthfully. The truthfulness result holds for all clusters. In addition, since the network is partitioned without the knowledge of the contractor agents' private information, the contractors cannot manipulate the mechanism in order to enter different clusters.

Similarly, we can show the mechanism is individually rational. The mechanism is a polynomial-time mechanism as both the allocation and the payment function can be computed in polynomial time when the number resource types is bounded by a constant (see Proposition 1). □

Nisan et al. [16] have showed that a class of VCG-based mechanisms where the output algorithms are *maximal in its range* (or MIR) are truthful. Informally speaking, an algorithm is maximal in its range if it optimizes the output over a—on forehand determined—set of allowable outputs. Our cluster-based mechanism $\mathcal{M}_{\mathsf{CLS}}$ actually belongs to this class of MIR. In addition, the following theorem says that if we desire a VCG-based truthful mechanism for social task allocation problem, this kind of MIR algorithm is all we can have.

Theorem 4. *For the social task allocation problem* STAP*, if a mechanism with VCG-based payment function is truthful, then its allocation algorithm (*ALG*) is maximal in its range.*

Proof. Let the type space of m agents be A^m. We define the set of allowable outputs of the mechanism using allocation algorithm ALG by $\mathcal{O} = \{\mathsf{ALG}(\mathbf{a}) | \mathbf{a} \in A^m\}$. The VCG-based payment function is defined as $p_i(\mathsf{ALG}(z, \mathbf{a})) = -v_i(\mathsf{ALG}(z, \mathbf{a})) + W(\mathsf{ALG}(z, \mathbf{a})) - h^{a_{-i}}$, where $h^{a_{-i}}$ is any function without agent i, and thus does not affect the mechanism's truthfulness. Let $h^{a_{-i}} = 0$. The utility of agent i is $u_i(s_i, a_{-i}) = v_i(s_i, \mathsf{ALG}(z, \mathbf{a})) + p_i(z, s_i, \mathbf{a}_{-i}) = W(\mathsf{ALG}(z, s_i, \mathbf{a}_{-i}))$, where s_i is the true type of i, which consists of all its available resources. Since the mechanism is truthful, we have for all i, for every other type $a_i \neq s_i$, that $u_i(a_i, \mathbf{a}_{-i}) \leq u_i(s_i, \mathbf{a}_{-i})$. Therefore, it holds that $W(\mathsf{ALG}(z, s_i, \mathbf{a}_{-i})) \geq W(\mathsf{ALG}(z, a_i, \mathbf{a}_{-i}))$. Thus, given any type in A^m, the algorithm ALG always maximizes the social welfare over the set of possible outputs $\mathcal{O}$. The algorithm is maximal in its range at A^m. □

According to Theorem 4, if a VCG-based mechanism is truthful, then the allocation algorithm must be the one which gives the optimal solution over a set of *restricted* allowable outputs. In the proposed cluster-based algorithm, such a restriction on outputs may bring some undesired performance loss: in some problem instances of STAP, the cluster algorithm may return an empty allocation, while the optimal algorithm allocates at least one task. As we cannot guarantee the performance theoretically, in the next section, we show the performance of this polynomial-time mechanism $\mathcal{M}_{\mathsf{CLS}}$ experimentally.

5 Experiments

In this section we investigate the performance of the cluster-based allocation algorithm for STAP on small-world networks. Small-world networks are networks where the neighbors of an agent often are neighbors of each other. This property occurs in many real-world (social) networks such as scientific collaborative networks and the Internet [20].

5.1 Experimental set-up

To get an idea of the performance of the cluster-based allocation algorithm for STAP we compare the resulting social welfare to the optimal algorithm. In all settings we generated a small-world network using the algorithm by Watts and Strogatz [20] with

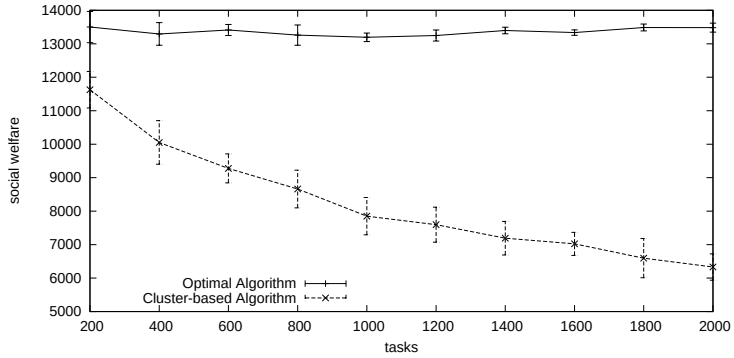

Fig. 2. The quality of the cluster-based algorithm decreases for larger number of tasks, given a fixed cluster size of 100 tasks, and a fixed number of agents of 150.

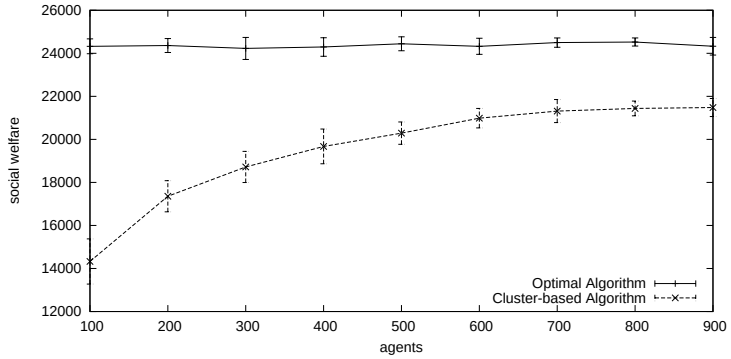

Fig. 3. The quality of the cluster-based algorithm increases for larger numbers of agents, given a fixed maximum cluster size of 100 tasks, and a fixed number of tasks of 600.

a rewiring probability of 0.05, and then generated tasks and their required resources, as well as the available resources for the agents using uniform distributions. Each specific run we did 20 times and calculated the average and the standard deviation over these 20 instances.

1. In the first experiment we investigated the influence of the number of tasks. For a small-world of 150 agents where each agent has on average 18 neighbors, we varied the number of tasks from 200 to 2000. In this setting each task requires 20 resources on average of 5 different types. The total number of resources available is exactly enough to fulfill all tasks. Note that because of the restrictions imposed by the network, not all these resources can be used. The result of this experiment is shown in Figure 2.
2. In the second experiment we varied the number of agents from 100 to 900 while keeping everything else the same for a problem with 600 tasks. The result of this experiment is shown in Figure 3.

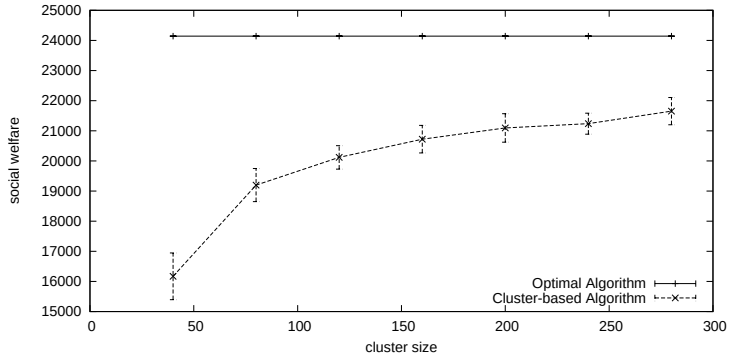

Fig. 4. The quality of the cluster-based algorithm increases for larger cluster sizes and a fixed number of tasks (600) and agents (400).

5.2 Results

From the first experiment we expected a drop in performance because the problem is splitted into more sub-problems. This result can be observed from the measurements in Figure 2 as well. The performance indeed decreases from 85% of the optimal down to about 50% for large instances with 2000 tasks.

More surprising was the result from the second experiment with the increasing number of agents. We expected to show that the quality of the cluster-based algorithm does not decrease if only the number of agents is increased. However, it turns out that for a fixed number of tasks, but an increasing number of (contractor) agents, the quality of the cluster-based algorithm significantly increases, to 88% of the optimal with 900 agents. This can be explained by the fact that because there are more agents, the average distance between tasks increases, and thus the sub-problems become more independent. The fact that the network has the small-world property increases this effect.

An increasing cluster size results in a smaller number of sub-problems that each is solved optimally. It can thus be expected that the quality of the algorithm increases with a larger cluster size. In a third experiment we confirmed this behavior (see Figure 4).

We also ran experiments with random and scale-free networks. As we expected, the cluster-based algorithm worked best on small-world networks. How to find good (community) structures and thus good task allocations in other types of networks we leave for future work.

6 Discussion and conclusion

Assigning services in a virtual organization or allocating resources in a peer-to-peer network can both be modeled as a social task allocation problem (STAP). In this paper we pose the problem of finding a truthful mechanism for STAP, i.e., to allocate resources to tasks in an environment where possible allocations are restricted by a network of relations between agents, and where the agents providing the resources are self-interested.

In this setting we assume that the tasks and the restrictions are given, and that the agents are asked for the resources they want to contribute.

We have presented a truthful optimal mechanism for this NP-complete problem, and we have showed how a truthful polynomial algorithm can be obtained by splitting the problem into smaller-sized sub-problems that can be solved optimally in polynomial time.

We have investigated the results of this algorithm for several problem instances based on a small-world network. We have showed that for a uniformly randomly generated range of such instances the heuristic achieved a quality of about 50-80% of the optimal algorithm. Since many real-world problems share this small-world property, this result is quite promising.

However, a polynomial-time algorithm with a guaranteed bound would be even more interesting. For the class of cost *minimization* allocation problems (CMAP) it has been proven that such a method does not exists [16] for a VCG-based truthful mechanism. This proof relies on the idea of constructing an instance for which the approximation gives a result that can be arbitrarily worse than the optimal solution.

The STAP is not a cost minimization problem. Our current research is to try to prove (or to disprove) that also for STAP no approximation can be found with a guaranteed bound on the quality. If this also holds for the STAP, the two approaches taken in this paper may be all that is possible. In that case, the next step of our research is to investigate the performance of the heuristic algorithm on real-world instances, and the influence of the small-world properties on the performance of the cluster-based heuristic.

The assumption that a task can only be assigned to its direct neighbors may impede the applicability of our approach to some problem domains. Our future work will introduce *mediator* agents into the network and study the consequent problems such as trust and incentives of the mediators.

Another ongoing research is to develop non-VCG-based polynomial-time truthful mechanisms for STAP. We are also interested in developing a mechanism for a setting where also the manager agents may lie about their types, and a mechanism which can be run distributedly. In addition, we would like to investigate the sufficient conditions of truthfulness in social task allocation domains.

Acknowledgments This work is supported by the Technology Foundation STW, applied science division of NWO, and the Ministry of Economic Affairs of the Netherlands.

References

1. M. Babaioff, R. Lavi, and E. Pavlov. Mechanism design for single-value domains. In M. M. Veloso and S. Kambhampati, editors, *Proc. of 20th Nat. Conf. on Artificial intelligence*, pages 241–247. AAAI, 2005.
2. L. Blumrosen and N. Nisan. Combinatorial auctions. In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, pages 209–242. Cambridge University Press, 2007.
3. E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1), September 1971.
4. M. de Weerdt, Y. Zhang, and T. B. Klos. Distributed task allocation in social networks. In *Proc. of 6th Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 17–24. ACM, 2007.

5. T. Groves. Incentives in teams. *Econometrica*, 41(4):617–31, July 1973.
6. S. Kraus, O. Shehory, and G. Taase. The advantages of compromising in coalition formation with incomplete information. In *Proc. of 3rd Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 588–595, Washington, DC, USA, 2004. IEEE Computer Society.
7. D. Lehmann, L. I. Oćallaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.
8. K. Lerman and O. Shehory. Coalition formation for large-scale electronic markets. In *Proc. of 4th Int. Conf. on Multi-Agent Systems*, pages 167–174. IEEE Computer Society, 2000.
9. A. Makhorin. GLPK. GNU Linear Programming Kit, 2004.
10. E. Manisterski, E. David, S. Kraus, and N. Jennings. Forming Efficient Agent Groups for Completing Complex Tasks. In H. Nakashima, M. P. Wellman, G. Weiss, and P. Stone, editors, *Proc. of 5th Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 257–264. ACM, 2006.
11. A. Mu'alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions: extended abstract. In *Proc. of 18th Nat. Conf. on Artificial intelligence*, pages 379–384, Menlo Park, CA, USA, 2002. AAAI.
12. R. Myerson. Incentive-compatibility and the bargaining problem. *Econometrica*, 47:61–73, 1979.
13. R. B. Myerson and M. A. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29(2):265–281, April 1983.
14. N. Nisan. Introduction to mechanism design (for computer scientists). In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, pages 209–242. Cambridge University Press, 2007.
15. N. Nisan and A. Ronen. Algorithmic mechanism design (extended abstract). In *Proc. of 31th ACM Symposium on Theory of Computing*, pages 129–140, New York, NY, USA, 1999. ACM.
16. N. Nisan and A. Ronen. Computationally feasible VCG mechanisms. In *Proc. of the 2nd ACM Conf. on Electronic commerce*, pages 242–252, New York, NY, USA, 2000. ACM.
17. P. V. Sander, D. Peleshchuk, and B. J. Grosz. A scalable, distributed algorithm for efficient task allocation. In *Proc. of 1st Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 1191–1198, New York, NY, USA, 2002. ACM.
18. O. Shehory and S. Kraus. Methods for Task Allocation via Agent Coalition Formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.
19. W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.
20. D. J. Watts and S. H. Strogatz. Collective dynamics of 'small world' networks. *Nature*, 393:440–442, 1998.