# Preventing Under-Reporting in Social Task Allocation

Mathijs de Weerdt and Yingqian Zhang

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

**Abstract.** In games where agents are asked to declare their available resources, they can also strategize over this declaration. Surprisingly, not in all such games a VCG payment can be applied to construct a truthful mechanism using an optimal algorithm, though such payments can prevent under-reporting of resources. We show this for the problem of allocating tasks in a social network (STAP).
Since STAP is NP-hard, we introduce an approximation algorithm as well. However for such an approximation, a VCG payment cannot prevent under-reporting anymore. Therefore we introduce an alternative payment function that motivates agents to fully declare their resources. We also demonstrate by experiments that the approximation algorithm works well in different types of social networks.

## 1 Introduction

In combinatorial auctions (CAs) agents try to buy sets (or bundles) of resources (or items) from an auctioneer preferably for less than what such a set is worth to them [1]. The values for these sets come for example from tasks that require these resources. In this paper we work towards a more general setting of such a resource allocation problem where the available resources have not just been trusted with a (central) auctioneer, but where these resources are owned by other agents.

In such a general resource allocation setting there are two classes of strategizing agents. On the one hand, there are agents that attach some value to specific sets of resources and construct bids based on this value (we call these the *managers*), and on the other hand there are agents that own resources and can decide to offer these resources to the others (we call these the *contractors*). In most current work on CAs [2] and resource allocation [3], agents are only allowed to strategize over the *value* of each bundle of resources. Here we consider the case where they may only strategize over the resources they have at their disposal.

As an application of the general resource allocation problem, consider a setting where a governmental institution would like to provide a platform to bring both suppliers and consumers together to form new contracts. In general, not all consumers (managers) want to buy services or resources from all suppliers (contractors), and not all contractors want to sell their resources to all managers.

We therefore include such a setting of preferred partnerships in our model as a graph where the agents are nodes, and an edge between two nodes indicates the existence of a social relation between the agents concerned, meaning that these agents are prepared to exchange resources. We call this problem, where the contractor agents can strategize over the resources they declare to their neighboring managers, the social task allocation problem (STAP) [4].

From the perspective of the general resource allocation problem introduced above, this problem can be seen as a variant of reverse CAs where it is publicly known how valuations depend on allocated resources, but the resources available to each contractor are private. As we will see in Section 3, this means that lying agents can incur infeasible solutions, and Vickrey-Clarke-Groves (VCG) mechanisms [5–7] are no longer truthful.

In this paper we study the strategies of the contractors in STAP for two different mechanisms: for an optimal algorithm with a VCG payment, and for a greedy approximation with an alternative payment function. First we start with a formal introduction of STAP and the mechanism design problem for STAP (Section 2). There we distinguish between two types of strategic behavior for the contractor agents: they can declare resources they do not have (over-reporting), or they can omit resources they do have (under-reporting). Next we show in Section 3 that even an optimal algorithm with a VCG payment cannot make truth-telling incentive compatible regarding both under- and over-reporting. Knowing that an optimal algorithm cannot deal with larger instances since STAP is NP-complete, we introduce an approximation and an alternative payment function in Section 4 with similar truth-telling properties as the VCG payment (i.e., only preventing under-reporting). Finally, we analyze the quality of the heuristic experimentally, and draw our conclusions in Sections 5 and 6, respectively.

## 2 Preliminaries

In STAP, we start from a *social network* $SN$ of a set of agents $\mathcal{A}$. This 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 $j$. Some of the agents have a set of resources at their disposal. These are the *contractor* agents. Resources come in different types. The set of $l$ resource types is $R$. The amount of resources of each type an agent $i \in \mathcal{A}$ has available is defined by the function $s_i : R \to \mathbb{N}$.

Some agents, called *managers*, have tasks they can perform to get some utility. The set of all $n$ tasks is denoted by $\mathcal{T} = \{t_1, \ldots, t_n\}$. Each task $t$ is defined by a tuple $\langle U(t), req(t), loc(t)\rangle$, where $U(t)$ is the utility gained if task $t$ is accomplished, $req(t) : R \to \mathbb{N}$ is a function that specifies the resources required for the accomplishment of task $t$, and $loc(t) : \mathcal{A}$ defines the manager of task $t$.

The exact assignment of how many resources of which type from which agent are assigned to which tasks is defined by a *task allocation*, which is a mapping $o : \mathcal{T} \times \mathcal{A} \times R \to \mathbb{N}$. A task allocation must obey the social relationships—each agent's resources can only be allocated to tasks that are (direct) neighbors of this agent in the social network $SN$. A valid task allocation must also be: (1) correct—each agent cannot use more than its available resources; and (2) complete—each task is either not allocated or it receives all required resources. The set of all valid task allocations for a specific problem setting is denoted by $\mathcal{O}$. We write $T_o$ to represent the tasks allocated in $o$. The utility of $o$ is the sum of the utilities of each task in $T_o$, i.e., $U(T_o) = \sum_{t \in T_o} U(t)$. Note that we do not include costs for resources; we assume they are already owned by the contractor agents. The goal of the *social task allocation problem* (STAP) is to find an *optimal* task allocation $o^*$, such that $o^*$ is valid and $U(T_{o^*})$ is maximal. STAP is NP-complete [4].

In this paper, we study the social task allocation problem in a *mechanism design* setting where the contractors can only strategize over the set of resources they declare. We give a brief summary of relevant mechanism design concepts below, but for a more elaborate introduction see e.g. [8]. 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 in this case the contractor agents, and possibly some additional true (often public) information. In our case this is the social network and a set of tasks. We use $Z$ to denote the space for this external information in STAP. Each $z \in Z$ is a tuple $(SN, \mathcal{T})$.

In the mechanism design setting we consider, each contractor is asked to declare its available resources, 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 agents except $i$. An agent can decide to declare another set of resources other than its true type. The set of all such choices is called its strategy space $A$. In our case $A = S$.

When the mechanism receives inputs $\mathbf{a} = (a_1, \ldots, a_m) \in A^m$ (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 $s_i$ and the payment it receives from the mechanism: $u_i(\mathbf{a}) = v_i(s_i, o) + p_i(z, \mathbf{a})$. This utility model is called *quasilinear*. This utility $u_i$ is what agent $i$ aims to maximize. In STAP, we define the valuation of a contractor 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 $t$: $e(t) = \frac{U(t)}{\sum_{r \in R} req(t)(r)}$. An agent then receives for each resource it is contributing the efficiency of the task it is allocated to, thus $v_i(a_i, o) = \sum_{t \in T_o} \sum_{r \in R} o(t, i, r) \cdot e(t)$. However, agent $i$ may not be able to fully contribute to a given allocation $o = O(z, \mathbf{a})$ because it is asked for resources it does not own. Therefore we define the valuation $v_i(s_i, o)$ that agent $i$ obtains based on its true type $s_i$ as

$$v_i(s_i, o) = \sum_{t \in T'_{o,i}} \sum_{r \in R} o(t, i, r) \cdot e(t), \tag{1}$$

where $T'_{o,i} = \{t \in T_o \mid \forall_r o(t,i,r) \leq s_i(r)\}$ is the set of allocated tasks that are feasible regarding agent $i$'s true type.

The *social welfare* $W(o)$ of the system is the sum of the valuations of the contractors in the allocation $o$, i.e., $W(o) = \sum_{i=1}^{m} v_i(s_i, o)$. We use this to define the mechanism design problem for STAP formally.

**Definition 1 (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 $p : Z \times A \to \mathbb{R}$ *such that the selected output $o \in \mathcal{O}$ maximizes the total social welfare $W(o)$.*

One of the most desirable properties of a mechanism is *truthfulness.*

**Definition 2 (Truthful).** *Given an output algorithm $O$, a mechanism is* truthful *if $A = S$, and for any parameter $z \in Z$, for any strategy profile $\mathbf{a_{-i}} \in A^{m-1}$, for any agent $i$ with type $s_i \in S$, and for any other type $a_i \in A$, 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, under a truthful mechanism, an agent $i$ is never worse off by revealing its true private type $s_i$ to the mechanism, no matter what strategies other agents play. In this paper, we study two types of lying by contractor agent $i$: (1) *under-reporting* its available resource types or amounts, i.e., $\exists_{r \in R} a_i(r) < s_i(r)$, denoted by $a_i < s_i$, and (2) *over-reporting* its available resource types or amounts, i.e., $\exists_{r \in R} a_i(r) > s_i(r)$, denoted by $a_i > s_i$.[1] We define *truthfulness with respect to under-reporting* and *truthfulness with respect to over-reporting* as follows.

**Definition 3.** *Given an output algorithm $O$, a mechanism is* truthful with respect to under-reporting (or with respect to over-reporting) *if $A = S$, and for any parameter $z \in Z$, for any strategy profile $\mathbf{a_{-i}} \in A^{m-1}$, for any agent $i$ with type $s_i \in S$ and for any other type $a_i \in A$ and $a_i < s_i$ (or $a_i > s_i$), it holds that*

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

**Proposition 1.** *If a mechanism for* STAP *is truthful then it is both truthful with respect to under-reporting as well as truthful with respect to over-reporting.*

*Proof.* This follows immediately from Definition 2 and Definition 3.

A mechanism is *individually rational* (IR) when an agent never receives negative utility by declaring its true type. We are looking for a mechanism that is IR, because otherwise agents have no incentive to take part at all.

It is well known that truthful mechanisms can be achieved with carefully designed payment functions, such as VCG payments [5–7]. Nisan et al. [9] also

[1] Note that agents can in principle also under-report some and over-report some other resources. We will discuss this mixed lying type at the end of Section 3.

showed that truthfulness is guaranteed by a VCG payment if the mechanism outputs the optimal solution. However, in the next section, we show that VCG mechanisms with an optimal algorithm can only achieve truthfulness with respect to under-reporting, but not with respect to over-reporting. Consequently, it is impossible to have a truthful VCG mechanism for $\mathsf{STAP}$.

## 3 An Exact VCG mechanism for STAP

In this section, we first introduce an optimal allocation algorithm for $\mathsf{STAP}$, and then a VCG mechanism to incentivize agents to report their true types with respect to under-reporting.

The optimal task allocation algorithm should deal with the restrictions posed by the social network. We translate this $\mathsf{NP}$-complete problem to an integer linear programming (ILP) problem and use the GNU Linear Programming Kit [10] 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: maxmize $\sum_{j=1}^{n} y_j \cdot U(t_j)$, subject to having sufficient resources of each type for each chosen task from the neighboring agents, and not using more resources than there are available, i.e.

$$\begin{array}{lll} \forall_{1\leq j\leq n}\forall_{1\leq k\leq l} & \sum_{\{i\in[1,m] \mid (i,loc(t_j))\in AE\}} x_{ijk} & \geq y_j \cdot req(t_j)(r_k), \text{ and} \\ \forall_{1\leq i\leq m}\forall_{1\leq k\leq l} & \sum_{j=1}^{n} x_{ijk} & \leq rsc(i)(r_k). \end{array}$$

This optimal algorithm ($\mathsf{OPT}$) is in the worst case exponential in the number of variables, i.e., the number of tasks, agents, and resource types.

Our mechanism is then developed using $\mathsf{OPT}$ and a VCG payment scheme as follows.

**Definition 4** (**$M_{\mathsf{OPT}}$ for $\mathsf{STAP}$**)**.** *Let $z = (SN, \mathcal{T})$ be given. The* ***task allocation mechanism*** *$M_{\mathsf{OPT}}$ is then defined as follows. First the mechanism center announces the set of tasks $\mathcal{T}$ 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.*

*For the VCG* ***payment function*** *$p^{\mathsf{OPT}}$ we follow Clarke's rule, taking an agent's marginal contribution to the society [8]: $p_i^{\mathsf{OPT}}(z, a_i, \mathbf{a}_{-i}) = -v_i(a_i, o) + W(o) - W(o_{-i})$, where $o_{-i} = \mathsf{OPT}(z, \mathbf{a}_{-i})$ is the efficient allocation computed by $\mathsf{OPT}$ without i's participation.*

**Proposition 2.** *The mechanism $\mathcal{M}_{\mathsf{OPT}} = (\mathsf{OPT}, p^{\mathsf{OPT}})$ is individually rational.*

*Proof.* When agent $i$ is truthful, its utility is computed by $u_i(s_i, \mathbf{a}_{-i}) = v_i(s_i, o) + p_i^{\mathsf{OPT}}(z, s_i, \mathbf{a}_{-i}) = W(o) - W(o_{-i})$. With agent $i$, the resulting allocation is never worse than that without $i$'s participation, because of the additional resources $i$ brings in. Therefore, $W(o) \geq W(o_{-i})$ and thus $u_i(s_i, \mathbf{a}_{-i}) \geq 0$, so $i$ is guaranteed to receive non-negative utility when declaring its true type. □

It seems that $\mathcal{M}^{\mathsf{OPT}}$ is both efficient and truthful, as its payment is VCG based and it uses an optimal allocation algorithm. Unfortunately, this is not always true. Before we show this, we discuss the relationship between the value $U(T_o)$ of the allocation $o$ computed by any allocation algorithm $O$ and the gained social welfare $W(o)$, since this relationship is important for the properties of the mechanism explained later. Consider the value of such an allocation $o$:

$$U(T_o) = \sum_{t \in T_o} U(t) = \sum_{t \in T_o} \sum_{r \in R} req(t)(r) \cdot e(t) = \sum_{i \in \mathcal{A}} \sum_{t \in T_o} \sum_{r \in R} o(t,i,r) \cdot e(t)$$

We distinguish two cases of a lying agent $i$: under-reporting and over-reporting.

- In the case of *under-reporting* ($a_i < s_i$), for each resource type $r \in R$, $o(t,i,r) \leq s_i(r)$, since the algorithm will not assign more resources than what $i$ declares. Therefore, by Eq. 1, we have $v_i(s_i, o) = \sum_{t \in T_o} \sum_{r \in R} o(t,i,r)e(t)$ since $T'_{o,i} = T_o$. Hence, $U(T_o) = \sum_{i=1}^{m} v_i(s_i, o) = W(o)$, i.e., the utility of the allocation is *equal* to the social welfare.
- In the case of *over-reporting* ($a_i > s_i$), the algorithm may use $i$'s non-existing but declared resources to allocate tasks. If so, the resulting allocation is *infeasible*, since agent $i$ cannot actually deliver these resources. Furthermore, since there exist a resource $r$ such that $o(t,i,r) > s_i(r)$, so by Eq. 1, we have $T'_{o,i} \subset T_o$. It follows that $U(T_o) \neq W(o)$, i.e. the utility of the allocation computed by the algorithm is *not* exactly the social welfare.

These results are used to show $M^{\mathsf{OPT}}$ is only efficient and truthful with respect to under-reporting, but not with respect to over-reporting.

**Theorem 1.** $\mathcal{M}_{\mathsf{OPT}} = (\mathsf{OPT}, p^{\mathsf{OPT}})$ *is efficient and truthful with respect to under-reporting.*

*Proof.* Let $s_i$ be the true type of agent $i$ and $a_i$ be any other type such that $a_i < s_i$. Given a problem instance $z$, 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 without $i$'s participation. We have shown that when $a_i < s_i$, $W(o) = U(T_o)$. Since $U(T_o)$ is maximal, then $W(o)$ is maximal. That is, the mechanism is efficient.

We now prove that agent $i$ never receives less utility by declaring its true type $s_i$ instead of $a_i$. The difference $\delta$ is calculated as follows:

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

Since the optimal allocation will not get worse by adding more resources in the system, $U(T_o) - U(T_{\hat{o}}) \geq 0$, thus, $W(o) - W(\hat{o}) \geq 0$ and then $\delta \geq 0$. □

Unfortunately, with $\mathcal{M}_{\mathsf{OPT}}$, agents do have an incentive to declare more resources than they actually have available.

**Theorem 2.** $\mathcal{M}_{\mathsf{OPT}} = (\mathsf{OPT}, p^{\mathsf{OPT}})$ *is not efficient and not truthful with respect to over-reporting.*

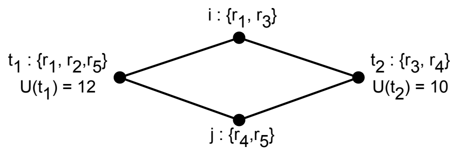

**Fig. 1.** Agent $i$ is better off by over-reporting when a VCG payment is used.

*Proof.* Consider the problem instance $z$ with tasks $t_1$ and $t_2$ with $U(t_1) = 12$, $U(t_2) = 10$ (see Figure 1). Task $t_1$ requires resources $\{r_1, r_2, r_5\}$; task $t_2$ requires $\{r_3, r_4\}$. Both tasks are connected to contractor agents $i$ and $j$, where $i$ has got resources $r_1$ and $r_3$, and $j$ owns $r_4, r_5$.

Suppose that agent $j$ is truthfully reporting its resource $r_4$. If $i$ also declares its type $\{r_1, r_3\}$ truthfully to the mechanism, the resulting optimal allocation $o_1 = \mathsf{OPT}(z, s_j, s_i)$ allocates $T_{o_1} = \{t_2\}$. So in this case, the utility that $i$ receives by declaring truthfully is: $u_i(s_j, s_i) = v_i(s_i, o_1) + p_i^{\mathsf{OPT}}(z, s_j, s_i) = W(o_1) = U(t_2) = 10$. Consider now the case where $i$ over-reports its resources, i.e., $a_i = \{r_1, r_2, r_3\}$. Then, both $t_1$ and $t_2$ are allocated, i.e. $T_{o_2} = \{t_1, t_2\}$. The utility of $i$ then becomes: $u_i(s_j, a_i) = v_i(s_i, o_2) + p_i^{\mathsf{OPT}}(z, s_j, a_i) = v_i(s_i, o_2) - v_i(a_i, o_2) + W(o_2) = \frac{10}{2} - (\frac{12}{3} \cdot 2 + \frac{10}{2}) + 22 = 14$. Since $u_i(s_j, a_i) > u_i(s_j, s_i)$, agent $i$ is better off by over-reporting its resources.

When $i$ over-declares, the output of $\mathsf{OPT}$ is not exactly maximizing the social welfare. It is not efficient with respect to over-reporting. □

The above example shows that $\mathcal{M}^{\mathsf{OPT}}$ is not truthful because an agent may declare a non-existing resource $r_2$ to improve its utility. In the example this results in an *infeasible* allocation, since $t_1$ cannot be executed successfully.[2] In general, it follows from Theorem 2 that no VCG mechanism for $\mathsf{STAP}$ can make the optimal algorithm truthful.

**Corollary 1.** *No VCG mechanism for* $\mathsf{STAP}$ *can make the optimal allocation algorithm* $\mathsf{OPT}$ *truthful.*

*Proof.* Consider the general description of a VCG payment for an agent $i$:

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

where $h^{\mathbf{a}_{-i}}$ is an arbitrary function that does not depend on $i$'s truthfulness. We can therefore assume that $h^{\mathbf{a}_{-i}} = 0$. Repeat the example in Theorem 2, we get the same result, i.e., agent $i$ is better off by over-reporting its available type. □

The results of [5–7, 9] show that VCG mechanisms are truthful if the mechanism selects the optimal one among all allowable (or feasible) outputs. The private information of a player in [9] is a set of *values* of its types. Therefore given the type space of the players, the set of feasible outputs is *known* to the

[2] Because this can be detected after the tasks have been executed, we may avoid over-reporting in some applications by transferring part of the payments afterwards.

mechanism. Here the private information of a contractor is its available resources. As a consequence, over-reporting may lead to an infeasible allocation. In other words, the mechanism has no knowledge about the set of feasible outputs. Thus, the "optimal" output is not optimizing the true social welfare. This is exactly why the VCG mechanism cannot guarantee truthfulness in STAP.

In principle, an agent can have a mixed type of lying, i.e., under-reporting some resources and over-reporting some others. However, if VCG mechanisms are not able to prevent "pure" over-reporting as shown in Theorem 2, they cannot prevent the mixed type of lying either. Therefore, in the remainder of this paper, we only study the case of under-reporting.

One of the disadvantages of VCG mechanisms is that they are not *budget-balanced* (BB), i.e. $\sum_{i=1}^{m} p_i = 0$. However, it is still interesting to study the total payment of the mechanism $\mathcal{M}_{\mathsf{OPT}}$.

**Proposition 3.** *The total payment of the mechanism* $\mathcal{M}_{\mathsf{OPT}} = (\mathsf{OPT}, p^{\mathsf{OPT}})$ *to the contractors is:* $-W(o) \leq \sum_{i=1}^{m} p_i^{\mathsf{OPT}} \leq (m-1)W(o)$, *where* $W(o)$ *is the optimal social welfare and* $m$ *is the number of contractor agents.*

## 4 A Greedy Mechanism for STAP

Often, we desire a polynomial-time mechanism where the allocation and the payments can both be computed in polynomial time. However, computing the efficient task allocation in a social network is NP-complete. Furthermore, when the number of neighbors of each agent is bounded by $\Delta$ for $\Delta \geq 3$, it is not approximable within $\Delta^{\varepsilon}$ for some $\varepsilon > 0$ (unless $\mathsf{P} = \mathsf{NP}$) [4]. This indicates that finding an approximation algorithm which has a non-trivial approximation ratio is difficult. Moreover, developing a payment function that makes such an approximation truthful is even more challenging. Still in this section, we work towards this by first introducing a greedy allocation approximation algorithm GTA. We show that VCG mechanisms for GTA cannot even make agents truthful with respect to under-reporting. We therefore propose a non-VCG mechanism that is truthful with respect to under-reporting.

### 4.1 A greedy allocation algorithm

The idea for the greedy allocation algorithm is based on a greedy approximation for 0-1 knapsack [11]: first sort all items on their relative value, and then try to insert them in this order. If an item is inserted, it is never removed again.

In the greedy allocation algorithm (GTA) for STAP we consider the tasks in order of efficiency (the ratio between utility and required resources). If a task is feasible, it is inserted, if not, it is removed from the current selection of tasks. Feasibility of a selection of tasks is checked by translating the problem to a (polynomially solvable) network flow instance (see Algorithm 1).

**Proposition 4.** *The greedy allocation algorithm (*GTA*) is a* $K|R|$*-approximation algorithm for* STAP*, where* $K$ *is the maximum number of resources of one type a task can require. The run time of* GTA *is* $\mathrm{O}(|R|n^2m)$.

**Algorithm 1** Greedy centralized allocation algorithm (GTA).

```
1. Sort all tasks from all managers in descending order of their efficiencies $e(t)$. Denote
   the sorted tasks by $t'_1, t'_2, \dots, t'_n$, and the current selection of tasks by $T' = \emptyset$.
2. For $i = 1, \dots, n$ do:
   (a) Test if the selection of tasks $T' \leftarrow T' \cup \{t'_i\}$ is feasible:
   (b) Create a network flow problem for each resource type $r \in R$ (separately):
       i. Create a source $s$ and a sink $s'$.
       ii. For each agent $j \in \mathcal{A}$, if $r \in a_j$ create an agent node and an edge from $s$
           to this node with capacity $a_j(r)$.
       iii. For each task $t \in T'$, if $r \in req(t)$ create a task node and an edge from
            this node to $s'$ with capacity $req(t)(r)$.
       iv. For each agent $j \in \mathcal{A}$ connect its node to all nodes of neighboring tasks,
           i.e., $\{t \in T' \mid (j, loc(t)) \in AE\}$. Give this connection unlimited capacity.
   (c) Solve the maximum flow problem for the created flow networks. If the total
       maximum flow in all networks is equal to $\sum_{t \in T'} \sum_{r \in R} req(t)(r)$, the current
       combination of tasks is feasible. Otherwise remove task $t'_i$ from $T'$.
3. Output the task set $T'$ and the current allocation.
```

So, in the worst case, GTA may return quite bad solutions. Therefore, in Section 5 we study the average performance of GTA by a set of experiments.

Unfortunately, GTA cannot be made truthful even with respect to under-reporting by using a VCG payment function, in contrast to OPT.

**Proposition 5.** *No VCG payment function can make the greedy task allocation algorithm* GTA *a truthful mechanism with respect to under-reporting.*

*Proof.* In this proof we show that for a specific instance the VCG payment cannot incentivize a contractor to declare all its available resources truthfully. First consider the general description of a VCG payment for any agent $i$ (see Equation 2), where without loss of generality we assume that $h^{\mathbf{a}_{-i}} = 0$. Thus, $p_i(z, \mathbf{a}) = -v_i(a_i, \mathsf{GTA}(z, \mathbf{a})) + W(\mathsf{GTA}(z, \mathbf{a}))$.

Consider a problem instance with tasks $t_1$, $t_2$ and $t_3$. Task $t_1$ requires resources $\{r_1, r_2, r_3\}$; task $t_2$ requires $\{r_2, r_4\}$; and $t_3$ requires $\{r_3, r_5\}$. All three tasks are connected to contractors $i$ and $j$, where $i$ has resources $\{r_1, r_4, r_5\}$, and $j$ owns $\{r_2, r_3\}$. Let the utilities of the tasks be $U(t_1) = 15$, $U(t_2) = 8$, and $U(t_3) = 8$. Thus the efficiencies are 5, 4, and 4, respectively. Suppose that agent $j$ is truthfully reporting its resources $\{r_2, r_3\}$. We now compare two situations. When $i$ also declares its type truthfully to the mechanism, i.e. $\{r_1, r_4, r_5\}$, then according to the greedy algorithm, the resulting allocation is $o_1 = \mathsf{GTA}(z, s_j, s_i)$ with $T_{o_1} = \{t_1\}$, because $t_1$ has the highest efficiency. The payment then is $p_i(z, s_j, s_i) = -v_i(s_i, o_1) + W(o_1)$. So in this case, the utility that $i$ receives by declaring truthfully is $u_i(s_j, s_i) = v_i(s_i, o_1) + p_i(z, s_j, s_i) = W(o_1) = 15$.

Consider now a case where $i$ under-reports ($a_i < s_i$) its resources, i.e., $\{r_4, r_5\}$. In this case $t_1$ cannot be allocated. The greedy algorithm then outputs the allocation $o_2 = \mathsf{GTA}(z, s_j, a_i)$ and $T_{o_2} = \{t_2, t_3\}$. The utility of $i$ then becomes $u_i(s_j, a_i) = v_i(s_i, o_2) - v_i(a_i, o_2) + W(o_2) = (4 + 4) - (4 + 4) + 16.$

**Algorithm 2** Greedy payment $p^{\mathsf{GTA}}$.

```
Inputs: a problem instance $z$, and the declared types $\mathbf{a}$.
For each agent $i$, let $b_i = 0$, and do
  1. For each resource type $r$ declared by agent $i$ in $a_i$, do
     (a) Sort tasks in $T_i^r$ in descending order of their efficiencies $e(t)$. Let $L$ denote this
         list of sorted tasks. Store the currently available resources of type $r$ of agent
         $i$: $k_{i,r} = a_i(r)$, and initialize the set of assigned tasks: $T_{i,r} = \emptyset$.
     (b) For each task $t \in L$, if $k_{i,r} \geq req(t)(r)$, then
          i. assign the amount $req(t)(r)$ of agent $i$'s resource $r$ to $t$,
         ii. update $i$'s available resource $r$: $k_{i,r} = k_{i,r} - req(t)(r)$; update the assigned
             task set: $T_{i,r} = T_{i,r} \cup \{t\}$.
     (c) For each task $t \in T_{i,r}$, if there exists no other agent $j$ such that $t \in T_j^r$ and
         $a_j(r) > 0$, update $b_i = b_i + e(t) \cdot req(t)(r)$.
  2. The payment to agent $i$ is calculated by: $p_i^{\mathsf{GTA}}(z, a_i, \mathbf{a}_{-i}) = -v_i(a_i, o) + b_i$.
```

Since $u_i(s_j, a_i) > u_i(s_j, s_i)$, agent $i$ is better off by under-reporting its available resources. This mechanism is not truthful with respect to under-reporting. □

## 4.2 A mechanism that is truthful with respect to under-reporting

Since we cannot make GTA truthful by a VCG payment, we propose a non-VCG payment function that pays agents for each resource that no other agent can provide. Consequently, agents will not benefit anymore from keeping essential resources from the mechanism to influence the selection of allocated tasks.

For this we introduce some notation and definitions. Given a strategy profile $\mathbf{a}$ and a set of tasks $\mathcal{T}$, let $T_i$ denote the set of agent $i$'s neighboring tasks to which it can contribute, and let $T_i^r$ denote the set of tasks of $i$'s neighbors to which agent $i$ can contribute a resource $r$, i.e., $req(t)(r) > 0$ and $a_i(r) > 0$. Clearly, $\bigcup_{r \in R} T_i^r = T_i$. The payment is based on the allocation of each resource $r$ of agent $i$ to the most efficient task $t$ such that the agent $i$'s valuation for such an allocation is as high as possible. However, we pay agent $i$ its contribution of resource $r$ to $t$ only if $r$ is unique for $t$, that is, no other agent $j$ connected to $t$ has declared $r$. This greedy payment is described in Algorithm 2.

We now define the greedy mechanism $\mathcal{M}_{\mathsf{GTA}} = (\mathsf{GTA}, p^{\mathsf{GTA}})$ which uses the greedy allocation algorithm GTA to determine the task allocation, and uses the greedy payment function $p^{\mathsf{GTA}}$ defined above (in Algorithm 2) to calculate the payments to each participating agent.

We first show in the following lemma that given an outcome $o$ based on an agent $i$'s declared type $a_i$ ($a_i \leq s_i$), $i$'s valuation based on its true type $s_i$ is equal to its valuation based on its declared type $a_i$.

**Lemma 1.** *Given any problem instance $z \in Z$, any algorithm $O$, strategy profile $\mathbf{a_{-i}} \in A^{m-1}$, and for any agent $i$ its declared type $a_i$, if $a_i \leq s_i$, it holds that $v_i(a_i, o) = v_i(s_i, o)$ where $o = O(z, \mathbf{a})$.*

**Theorem 3.** *The greedy mechanism $\mathcal{M}_{\mathsf{GTA}} = (\mathsf{GTA}, p^{\mathsf{GTA}})$ is truthful with respect to under-reporting, individually rational, and runs in polynomial-time.*

*Proof.* Let a problem instance $z$, the declared types of others $\mathbf{a}_{-i}$, and the declaration $a_i \leq s_i$ be given. From this, the outcome $o = \mathsf{GTA}(z, a_i, \mathbf{a}_{-i})$ can be calculated. The utility of an agent $i$ is then determined by $u_i(a_i, \mathbf{a}_{-i}) = v_i(s_i, o) + p_i^{\mathsf{GTA}}(z, a_i, \mathbf{a}_{-i}) = v_i(s_i, o) - v_i(a_i, o) + b_i$, where $b_i$ is computed based on the greedy payment given in Algorithm 2. We know from Lemma 1 that $v_i(s_i, o) = v_i(a_i, o)$ for $a_i \leq s_i$. Thus the utility of $i$ completely depends on the value of $b_i$, i.e. $u_i(a_i, \mathbf{a}_{-i}) = b_i$. According to the computation of $b_i$ in Algorithm 2, for each resource type $r$, the value that $i$ can get from the allocated tasks $T_{i,r}$ is maximal when $i$ declares its full amount of resources, because in this way, more highly efficient tasks can be allocated, no matter whether its resource $r$ is unique to tasks in $T_{i,r}$. Moreover, $b_i$ is maximized when $i$ declares every resource type $r$ that it has. In other words, each agent's utility is *monotonically increasing* with its declared resources.

An agent $i$'s utility for declaring its true type $s_i$ is $u_i(s_i, \mathbf{a}_{-i}) = b_i \geq 0$. So, it is rational for agent $i$ to participate.

In the greedy payment $p^{\mathsf{GTA}}$, sorting the tasks takes $\mathrm{O}(n \log(n))$, and determining the value and checking the unique for the resource of each contractor takes $\mathrm{O}(|R|nm)$. Hence the total payment computation is $\mathrm{O}(n \log(n) + |R|m^2 n)$. Since both $\mathsf{GTA}$ and $p^{\mathsf{GTA}}$ can be computed in polynomial time, the mechanism is a polynomial-time mechanism. □

This result (and its proof) can be generalized for any mechanism for $\mathsf{STAP}$ as long as we can make the agent's utility function monotonically increasing with the declared resources.

**Theorem 4.** *Given any problem instance $z \in Z$, any allocation algorithm $O$, for any agent $i$ with type $s_i \in S$ and for any other type $a_i \in A$ and $a_i \leq s_i$, a mechanism for $\mathsf{STAP}$ is truthful with respect to under-reporting if the payment to any agent $i$ is of the form $p_i^{MON} = -v_i(a_i, O(z, \mathbf{a})) + h(a_i)$, where $h(a_i)$ is any function which is monotonically increasing with the declaration $a_i$.*

**Proposition 6.** *The total payment of the mechanism $\mathcal{M}_{\mathsf{GTA}} = (\mathsf{GTA}, p^{\mathsf{GTA}})$ to the contractors is: $-W(o) \leq \sum_{i=1}^{m} p_i^{\mathsf{GTA}} \leq U(\mathcal{T})$, where $W(o)$ is the social welfare and $U(\mathcal{T})$ is the total utility of all tasks.*

## 5 Experiments

The worst-case performance guarantee for the greedy heuristic presented in the previous section is based on a specific case where one task with a high efficiency blocks all other tasks. In practice, not all tasks are connected to the same agents and the average performance will be much better. To see how much better, we investigated the performance of this mechanism experimentally.

A problem instance in these experiments is defined by the number of agents, tasks, and resource types; the requirements, the utility and the location of each task; the available resources at each agent and the relations between the agents. In our case these relations are defined by the type of social network. The types we

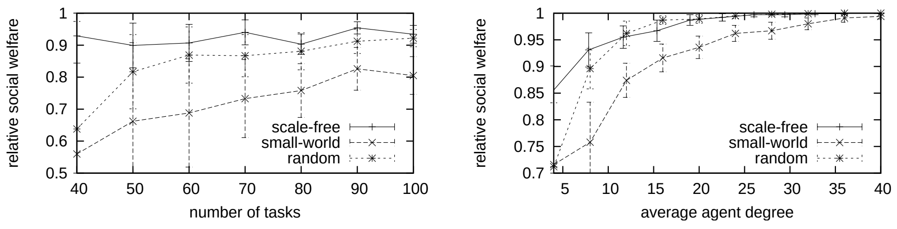

**Fig. 2.** These graphs show the relative social welfare of the greedy mechanism related to the number of tasks (left), and the degree of the networks (right), respectively.

consider in our experiments are: (i) random networks, where agents are connected completely at random until the desired degree has been reached, (ii) small-world networks [12], where most agents are connected locally (to the neighbors of their neighbors), but have a fixed "rewiring" probability of 0.05 to connected to any other agent, and (iii) scale-free networks, where agents have a higher probability to get connected to agents with more neighbors [13].

First we investigated each parameter separately using default settings for the other parameters such that the optimal solution can still be calculated in at most 10 minutes. These defaults were: 200 contractors, 100 tasks, 20 resource types, 20 resources per task on average, 10 resources on average per agent, a uniformly random distribution of resources and tasks to agents, and the value of a task drawn from a normal distribution around two times the number of resources with a standard deviation of one time this number (with a minimal value of 1). We kept the total number of resources required by the tasks and the total available resources in the network the same, and equal to each other. In a fully connected network this would mean that all tasks can be allocated. However, in the networks we consider they generally cannot. For each experiment with the parameter settings described above, we generated 20 instances of each of the three types of networks. In the plots we only showed the average and the standard deviation over these 20 instances for each setting.

When we varied the problem size by increasing the number of tasks from 40 to 100, we noticed (see the first graph in Figure 2) an increase of the quality of the greedy mechanism from about 0.55 up to about 0.8 in the small-world network setting, and about 0.9 in the random and scale-free setting. Interestingly, it seems that the greedy algorithm works better on structures like scale-free networks where some tasks (managers) are very well connected, while some others are (almost) not connected at all, than on more uniform structures like small-world networks, where each task has about the same number of connections.

Next we show the results for problem instances with a degree of the social network varying from 4 to 40. The greedy mechanism performed extremely well for networks with a large degree showing a relative social welfare of about 0.99. In such a setting most tasks can be allocated, because they end up in very well-connected nodes, having many alternative contractor agents.

A more important observation is that the performance over all these instances (about 200 per figure) was between 0.55 and 0.99, and on average around 0.85 of the optimal outcome. This is a much better result than the worst-case theoretical bound of $K|R|$ (in this case $K$ was 8) presented in Section 4.1.

## 6 Discussion and Conclusions

This paper started with a very general resource allocation problem where both the agents providing the resources (contractors) as well as the agents having some utility for bundles of resources (managers) can be strategizing. When the contractors are trusted, and only the managers are allowed to strategize over their value for each bundle, we end up with the well-studied CA problem. However, in this paper we consider a mechanism design setting which differs in an important way from most of the studied situations, such as CAs [2], single-parameter and single-value domains [14, 15], because the contractors are not strategizing about the valuation they declare, but about the resources they have available. When agents lie about their valuation, the output of a mechanism can be inefficient, but is always feasible. However, when agents lie about such things as their available resources, the mechanism can come up with an infeasible outcome.

We showed that in such a setting a VCG payment with an optimal algorithm cannot guarantee a truthful mechanism, but it can realize a mechanism that is truthful with respect to under-reporting. Moreover, since the problem is NP-hard, we can only expect to find optimal algorithms that run in time exponential in the size of the input.

In this paper we therefore proposed a polynomial-time approximation. We first showed that using this approximation, VCG mechanisms cannot be used to create similar truth-telling properties as for the optimal mechanism. However, we then proposed another payment function that actually could realize these properties. In general, we showed that this is because the utility of agents is monotonically increasing in the number of declared resources. Our experimental results show that the overall quality of the solutions using this mechanism is quite good. Since STAP is NP-complete, we conclude that this heuristic is more useful than the optimal mechanism in many practical situations with a moderate or larger problem size.

The fact that agents have an incentive to over-report is very severe, because it could lead to infeasible solutions. However, we believe that this consequence may also imply a solution, because lying agents can thus be detected and punished. This is an important topic in our current work.

Furthermore, we will consider the general resource allocation problem where not only the contractor agents, but also the managers can strategize. Our goal is to also incentivize the managers to truthfully declare their tasks and the attached utilities. It is relatively straightforward to see that a VCG payment can achieve this if an optimal algorithm is used. How to achieve this using an approximation algorithm is still an open question.

With the proposed greedy payment algorithm an agent may receive zero utility if its contribution (i.e. resource) is not unique to the task. Moreover, additional funding is required to incentivize the agents. To avoid these undesirable situations, we are interested in developing a dynamic payment scheme which adapts the payments based on (i) the specific instance, e.g. the resource distribution, and (ii) the generated social welfare, while maintaining the truthfulness property.

As a final remark, we believe that the results in this paper generalize to other (NP-hard) problem domains where a wrong input to the mechanism by the agents can lead to infeasible outcomes, such as in multiagent planning. We intend to show in the future that alternative payment functions such as the one proposed here can be applied to those settings as well.

**Acknowledgments**

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

## References

1. de Vries, S., Vohra, R.: Combinatorial Auctions: A Survey. INFORMS Journal on Computing **15**(3) (2003) 284–309
2. Blumrosen, L., Nisan, N.: Combinatorial auctions. In: Algorithmic Game Theory. Cambridge University Press (2007) 209–242
3. Chevaleyre, Y., Dunne, P.E., Endriss, U., Lang, J., Lemaitre, M., Maudet, N., Padget, J., Phelps, S., Rodriguez-Aguilar, J.A., Sousa, P.: Issues in multiagent resource allocation. Informatica **30** (2006) 3–31
4. de Weerdt, M., Zhang, Y., Klos, T.B.: Distributed task allocation in social networks. In: Proc. of 6th Int. Conf. on AAMAS, ACM (2007) 17–24
5. Vickrey, W.: Counterspeculation, auctions, and competitive sealed tenders. The Journal of Finance **16**(1) (1961) 8–37
6. Clarke, E.H.: Multipart pricing of public goods. Public Choice **11**(1) (1971)
7. Groves, T.: Incentives in teams. Econometrica **41**(4) (1973) 617–31
8. Nisan, N.: Introduction to mechanism design (for computer scientists). In: Algorithmic Game Theory. Cambridge University Press (2007) 209–242
9. Nisan, N., Ronen, A.: Algorithmic mechanism design (extended abstract). In: Proc. of 31th ACM Symposium on Theory of Computing, ACM (1999) 129–140
10. Makhorin, A.: GLPK. GNU Linear Programming Kit (2004)
11. Dantzig, G.: Discrete variable problems. Operations Research **5** (1957) 266–277
12. Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small world’ networks. Nature **393** (1998) 440–442
13. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439) (1999) 509–512
14. Archer, A., Tardos, E.: Truthful mechanisms for one-parameter agents. In: Proc. of 42nd IEEE Symposium on FOCS. (2001) 482–491
15. Babaioff, M., Lavi, R., Pavlov, E.: Mechanism design for single-value domains. In: Proc. of 20th Nat. Conf. on Artificial intelligence, AAAI (2005) 241–247