Algorithmic Mechanism Design

Noam Nisan*
Institute of Computer Science, Hebrew University of Jerusalem, Givat Ram 91904, Israel
and
School of Computer Science, IDC, Herzliya
Email:noam@cs.huji.ac.il

Amir Ronen†
Institute of Computer Science, Hebrew University of Jerusalem, Givat Ram 91904, Israel
Email:amiry@cs.huji.ac.il

*This research was supported by grants from the Israeli ministry of Science and the Israeli academy of sciences.
†This research was supported by grants from the Israeli ministry of Science and the Israeli academy of sciences.

Algorithmic Mechanism Design

contact:

Amir Ronen

Institute of Computer Science, Hebrew University of Jerusalem, Givat Ram 91904, Israel

Email:amiry@cs.huji.ac.il

Abstract

We consider algorithmic problems in a distributed setting where the participants cannot be assumed to follow the algorithm but rather their own self-interest. As such participants, termed agents, are capable of manipulating the algorithm, the algorithm designer should ensure in advance that the agents' interests are best served by behaving correctly.

Following notions from the field of mechanism design, we suggest a framework for studying such algorithms. In this model the algorithmic solution is adorned with payments to the participants and is termed a mechanism. The payments should be carefully chosen as to motivate all participants to act as the algorithm designer wishes. We apply the standard tools of mechanism design to algorithmic problems and in particular to the shortest path problem.

Our main technical contribution concerns the study of a representative problem, task scheduling, for which the standard tools do not suffice. We present several theorems regarding this problem including an approximation mechanism, lower bounds and a randomized mechanism. We also suggest and motivate extensions to the basic model and prove improved upper bounds in the extended model. Many open problems are suggested as well.

Journal of Economic Literature classification numbers: C60, C72, D61, D70, D80.

1 Introduction

1.1 Motivation

A large part of research in computer science is concerned with protocols and algorithms for interconnected collections of computers. The designer of such an algorithm or protocol always makes an implicit assumption that the participating computers will act as instructed – except, perhaps, for the faulty or malicious ones.

With the emergence of the Internet as *the* platform of computation, this assumption can no longer be taken for granted. Computers on the Internet belong to different persons or organizations and will likely do what is most beneficial to their owners. We cannot simply expect each computer on the Internet to faithfully follow the designed protocols or algorithms. It is more reasonable to expect that each computer will try to manipulate it for its owners' benefit. Such an algorithm or protocol must therefore be designed in advance for this kind of behavior! Let us sketch two example applications we have in mind:

Load balancing

The aggregate power of all computers on the Internet is huge. In a "dream world" this aggregate power will be optimally allocated online among all connected processors. One could imagine CPU-intensive jobs automatically migrating to CPU-servers, caching automatically done by computers with free disk space, etc. Access to data, communication lines and even physical attachments (such as printers) could all be allocated across the Internet. This is clearly a difficult optimization problem even within tightly linked systems, and is addressed, in various forms and with varying degrees of success, by all distributed operating systems. The same type of allocation over the Internet requires handling an additional problem: the resources belong to different parties who may not allow others to freely use them. The algorithms and protocols may, thus, need to provide some motivation for these owners to "play along".

Routing

When one computer wishes to send information to another, the data usually gets routed through various intermediate routers. So far this has been done voluntarily, probably due to the low marginal cost of forwarding a packet. However, when communication of larger amounts of data becomes common (e.g. video), and bandwidth needs to be reserved under various quality of service (QoS) protocols, this altruistic behavior of the routers may no longer hold. If so, we will have to design protocols specifically taking the routers' self-interest into account.

1.2 This Work

In this paper we propose a formal model for studying algorithms that assume that the participants all act according to their own self-interest. We adopt a rationality-based approach, using notions from game theory and micro-economics, and in particular from the field of mechanism design. We assume that each participant has a well defined **utility function**[1] that represents its preference over the possible outputs of the algorithm, and we assume that participants act as to rationally optimize their utility. We term such rational and selfish participants **agents**[2]. The solutions we consider contain both an algorithmic ingredient (obtaining the intended results), and a payment ingredient that motivates the agents. We term such a solution a **mechanism**[3].

Our contributions in this work are as follows:

1. We present a formal model for studying optimization problems. The model is based on the field of mechanism design[4]. A problem in this model has, in addition to the output specification, a description of the agents' utilities. The mechanism has, in addition to the algorithm producing the desired output, payments to the participating agents. An exposition of applying several classic notions from mechanism design in our model appears in Nisan (1999).

2. We observe that the known techniques from mechanism design provide solutions for several basic optimization problems, and in particular for the shortest path problem, where each edge may belong to a different agent.

3. We study a basic problem, task scheduling, which requires new techniques and prove the following:

   - We design an n-approximation mechanism, where n is the number of agents.
   - We prove a lower bound of 2 to the approximation ratio that can be achieved by any mechanism. This bound is tight for the case of two agents, but leaves a gap for more agents. We conjecture that the upper bound is tight in general and prove it for two restricted classes of mechanisms.

[1] This notion from micro-economics is often used in mechanism design.

[2] The term is taken from the distributed AI community which have introduced the usage of mechanism design in a computational setting. We use it, however, in a much more restricted and well-defined sense.

[3] This is the standard term used in mechanism design.

[4] We are not the first to use notions from mechanism design in a computational setting. See section 1.3.

- We design a randomized mechanism that beats the deterministic lower bound.

4. We extend the basic model, formalizing a model where the mechanism has more information. We call this model a mechanism with verification and argue that it is justified in certain applications.

5. We study the task scheduling problem in the extended model and obtain two main results:

   - An optimal mechanism with verification for task scheduling (that requires exponential computation time).
   - A polynomial time $(1+\epsilon)$-approximation mechanism with verification for a sub-case of the problem.

A preliminary version of this paper appeared at the thirty-first annual symposium on theory of computing (Nisan and Ronen (1999)).

1.3 Extant Work

There have been many works that tried to introduce economic or game-theoretic aspects into computational questions. (See e.g. Lamport, Shostak and Pease (1982), Ferguson, Nikolaou and Yemini (1995), Huberman and Hogg (1995), Papadimitriou and Yannakakis (1993), Papadimitriou and Yannakakis (1991) and a survey by Lineal (1994)). Most of these were not aimed at the problem of the cooperation of selfish entities, and those that were (Monderer and Tennenholtz (1998), Papadimitriou (1996), Korilis, Lazar and Orda (1991) and Sandholm (1996)) did not pursue our direction. Many subfields of game theory and economics are also related to our work, see, e.g. Mas-Collel, Whinston and Green (1995) chapters 14, 21 and 22. We list below the research work that is most relevant to our direction.

Mechanism Design

The field of mechanism design (also known as implementation theory) aims to study how privately known preferences of many people can be aggregated towards a "social choice". The main motivation of this field is micro-economic, and the tools are game-theoretic. Emphasis is put on the implementation of various types of auctions. In the last few years this field has received much interest, especially due to its influence on large privatizations and spectrum allocations (McMillan (1994)). An introduction to this field can be found in Mas-Collel, Whinston and Green (1995) chapter 23, Osborne and Rubistein (1994) chapter 10 and an influential web site in http://www.market-design.com.

Distributed AI

In the last decade or so, researchers in AI have studied cooperation and competition among "software agents". The meaning of agents here is very broad, incorporating attributes of code-mobility, artificial-intelligence, user-customization and self-interest. A subfield of this general direction of research takes a game theoretic analysis of agents' goals, and in particular uses notions from mechanism design (Rosenschein and Zlotkin (1994), Sandholm (1996), Ephrati and Rosenschein (1991) and Shoham and Tanaka (1997)). A related subfield of Distributed AI, sometimes termed market-based computation (Walsh, Wellman, Wurman and MacKie-Mason (1998), Ferguson, Nikolaou and Yemini (1995) and Walsh and Wellman (1998)), aims to leverage the notions of free markets in order to solve distributed problems. These subfields of DAI are related to our work.

Communication Networks

In recent years researchers in the field of network design adopted a game theoretic approach (See e.g. Korilis, Lazar and Orda (1995)). In particular mechanism design was applied to various problems including resource allocation (Lazar and Semret (1998)), cost sharing and pricing (Shenkar, Clark and Hertzog (1996)).

Scheduling

The specific problem we address is the minimization of the make-span of independent tasks on unrelated parallel machines, which was extensively studied from an algorithmic point of view. It is known that solving the problem or even approximating it within a factor of $3/2 - \epsilon$ is NP-hard, but a polynomial-time 2-approximation exists (Lenstra, Shmoys and Tardos (1987)). For a fixed number of processors, a fully polynomial approximation scheme was presented by Horowitz and Sahni (1976). A survey of scheduling algorithms can be found in Hochbaum (1997).

2 The Model

In this section we formally present our model. We attempt, as much as possible, to use the standard notions from both mechanism design and algorithmics. We limit ourself to the discussion of a dominant strategy implementation in quasi-linear environments.

Subsection 2.1 describes what a mechanism design problem is. In subsection 2.2 we define what a good solution is: an implementation with dominant strategies. Subsection 2.3 defines a special class of good solutions: truthful implementations, and states the well-known fact that restricting ourselves to such solutions loses no generality. For familiarization with our basic model and notations we suggest viewing the shortest paths example given in section 3.2.

2.1 Mechanism Design Problem Description

Intuitively, a mechanism design problem has two components: the usual algorithmic output specification, and descriptions of what the participating agents want, formally given as utility functions over the set of possible outputs.

Definition 1 (Mechanism Design Problem) *A* mechanism design problem *is given by an output specification and by a set of agents' utilities. Specifically:*

1. *There are n agents, each agent i has available to it some* **private** *input $t^i \in T^i$ (termed its type). Everything else in this scenario is public knowledge.*
2. *The* output specification *maps to each type vector $t = t^1...t^n$ a set of allowed outputs $o \in O$.*
3. *Each agent i's preferences are given by a real valued function: $v^i(t^i, o)$, called its* valuation. *This is a quantification of its value from the output o, when its type is t^i, in terms of some common currency. I.e. if the mechanism's output is o and in addition the mechanism hands this agent p^i units of this currency, then its* utility *will be $u^i = p^i + v^i(o, t^i)$*[5]*. This utility is what the agent aims to optimize.*

In this paper we only consider the important special case of optimization problems. In these problems the output specification is to optimize a given objective function. We present the definition for minimization problems.

Definition 2 (Mechanism Design Optimization Problem) *This is a mechanism design problem where the output specification is given by a positive real valued objective function $g(o, t)$ and a set of feasible outputs F. In the exact case we require an output $o \in F$ that minimizes g, and in the approximate case we require any $o \in F$ that comes within a factor of c, i.e. such that for any other output $o' \in F$, $g(o, t) \leq c \cdot g(o', t)$.*

[5]This is termed "quasi-linear utility". In this paper we limit ourselves to this type of utilities.

2.2 The Mechanism

Intuitively, a mechanism solves a given problem by assuring that the required output occurs, when agents choose their strategies as to *maximize their own selfish utilities.* A mechanism needs thus to ensure that the agents' utilities (which it can influence by handing out payments) are compatible with the algorithm.

Notation: We will denote $(a^1, ...a^{i-1}, a^{i+1}, ...a^n)$ by a^{-i}. (a^i, a^{-i}) will denote the tuple $(a^1, \ldots a^n)$.

Definition 3 (A Mechanism) *A* mechanism $m = (o, p)$ *is composed of two elements: An* output function $o()$*, and an n-tuple of* payments $p^1()...p^n()$*. Specifically:*

1. *The mechanism defines for each agent i a family of* strategies A^i*. The agent can chose to perform any $a^i \in A^i$.*
2. *The first thing a mechanism must provide is an* output *function $o = o(a^1...a^n)$.*
3. *The second thing a mechanism provides is a* payment $p^i = p^i(a^1...a^n)$ *to each of the agents.*
4. *We say that a mechanism is an* implementation with dominant strategies [6] *(or in short just an implementation) if*
   - *For each agent i and each t^i there exists a strategy $a^i \in A^i$, termed dominant, such that for all possible strategies of the other agents a^{-i}, a^i maximizes agent i's utility. I.e. for every $a'^i \in A^i$, if we define $o = o(a^i, a^{-i})$, $o' = o(a'^i, a^{-i})$, $p^i = p^i(a^i, a^{-i})$, $p'^i = p^i(a'^i, a^{-i})$, then $v^i(t^i, o) + p^i \geq v^i(t^i, o') + p'^i$*
   - *For each tuple of dominant strategies $a = (a^1...a^n)$ the output $o(a)$ satisfies the specification.*

We say that a mechanism is poly-time computable if the output and payment functions are computable in polynomial time. In this paper we purposefully do not consider the details of how the mechanism is computed in a distributed system. We view this topic as an important direction for further research.

[6] Several "solution concepts" are discussed in the mechanism design literature. In this paper we discuss only dominant strategy implementation.

2.3 The Revelation Principle

The simplest types of mechanisms are those in which the agents' strategies are to simply report their types.

Definition 4 (Truthful Implementation) *We say that a mechanism is* truthful *if*

1. *For all i, and all t^i, $A^i = T^i$, i.e. the agents' strategies are to report their type. (This is called a direct revelation mechanism.)*
2. *Truth-telling is a dominant strategy, i.e. $a^i = t^i$ satisfies the definition of a dominant strategy above.*

Definition 5 *We say that a mechanism is* strongly truthful *if truth-telling is the only dominant strategy.*

A simple observation, known as the *revelation principle*, states that without loss of generality one can concentrate on truthful implementations.

Proposition 2.1 *(Mas-Collel, Whinston and Green (1995) page 871) If there exists a mechanism that implements a given problem with dominant strategies then there exists a truthful implementation as well.*

Proof: (sketch) We let the truthful implementation simulate the agents' strategies. I.e. given a mechanism $(o, p^1, ...p^n)$, with dominant strategies $a^i(t^i)$, we can define a new one by $o^*(t^1...t^n) = o(a^1(t^1)...a^n(t^n))$ and $(p^*)^i(t^1...t^n) = p^i(a^1(t^1)...a^n(t^n))$. $\square$

3 Vickrey-Groves-Clarke Mechanisms

Arguably the most important positive result in mechanism design is what is usually called the generalized Vickrey-Groves-Clarke (VGC) mechanism (Vickrey (1961); Groves (1973); Clarke (1971)). We first describe these mechanisms in our notation and then demonstrate their usage in an algorithmic setting, that of shortest paths.

3.1 Utilitarian Functions

The VGC mechanism applies to mechanism design maximization problems where the objective function is simply the sum of all agents' valuations. The set of possible outputs is assumed to be finite.

Definition 6 *A maximization mechanism design problem is called* utilitarian *if its objective function satisfies* $g(o,t) = \sum_i v^i(t^i, o)$.

Definition 7 *We say that a direct revelation mechanism* $m = (o(t), p(t))$ *belongs to the* VGC *family if*

1. $o(t) \in \arg\max_o(\sum_{i=1}^n v^i(t^i, o))$.

2. $p^i(t) = \sum_{j \neq i} v^j(t^j, o(t)) + h^i(t^{-i})$ *where* $h^i()$ *is an arbitrary function of* t^{-i}.

Theorem 3.1 *(Groves (1973)) A VGC mechanism is truthful.*

$\square$

Thus, a VGC mechanism essentially provides a solution for any utilitarian problem (except for the possible problem that there might be dominant strategies other than truth-telling). It is known that (under mild assumptions) VGC mechanisms are the only truthful implementations for utilitarian problems (Green and Laffont (1977)).

Similarly, a weighted version can be implemented as well.

Definition 8 *A maximization mechanism design problem is called* weighted utilitarian *if there exist real numbers* $\beta^1, \ldots, \beta^n > 0$ *such that the problem's objective function satisfies* $g(o,t) = \sum_i \beta^i \cdot v^i(t^i, o)$.

Definition 9 *We say that a direct revelation mechanism* $m = (o(t), p(t))$ *belongs to the* weighted VGC *family if*

1. $o(t) \in \arg\max_o(g(o,t))$.

2. $p^i(t) = \frac{1}{\beta^i} \cdot \sum_{j \neq i} \beta^j \cdot v^j(t^j, o(t)) + h^i(t^{-i})$ *where* $h^i()$ *is an arbitrary function of* t^{-i}.

Theorem 3.2 *(Roberts (1977)) A weighted VGC mechanism is truthful.*

Proof: Let $d^1, \ldots, d^n$ denote the declarations of the agents and $t^1, \ldots, t^n$ denote their real types. Suppose that truth telling is not a dominant strategy, then there exists d, i, t, d'^i such that

$$v^i(t^i, o(d^{-i}, t^i)) + p^i(t^i, o(d^{-i}, t^i)) + h^i(d^{-i}) < v^i(t^i, o(d^{-i}, d'^i)) + p^i(t^i, o(d^{-i}, d'^i)) + h^i(d^{-i})$$

thus

$$v^i(t^i, o(d^{-i}, t^i)) + \frac{1}{\beta^i} \cdot \sum_{j \neq i} \beta^j \cdot v^j(t^j, o(d^{-i}, t^i)) < v^i(t^i, o(d^{-i}, d'^i)) + \frac{1}{\beta^i} \cdot \sum_{j \neq i} \beta^j \cdot v^j(t^j, o(d^{-i}, d'^i))$$

Multiplying both sides by β^i we obtain

$$\sum_{j=1}^{n} \beta^j \cdot v^j(t^j, o(d^{-i}, t^i)) < \sum_{j=1}^{n} \beta^j \cdot v^j(t^j, o(d^{-i}, d'^i))$$

In contradiction with the definition of $o()$. □

Comment: An output function of a VGC mechanism is required to maximize the objective function. In many cases (e.g. combinatorial auctions (Harstad, Rothkopf and Pekec (1995))), this makes the mechanism computationally intractable. Replacing the optimal algorithm with a non-optimal approximation usually leads to untruthful mechanisms – see for example section 5.6.

3.2 Example: Shortest Paths

Many algorithmic mechanism design problems can be solved using the VGC mechanism. Let us give a natural example.

Problem definition: We have a communication network modeled by a directed graph G, and two special nodes in it x and y. Each edge e of the graph is an agent. Each agent e has private information (its type) $t^e \geq 0$ which is the agent's cost for sending a single message along this edge. The goal is to find the cheapest path from x to y (as to send a single message from x to y). I.e the set of feasible outputs are all paths from x to y, and the objective function is the path's total cost. Agent e's valuation is 0 if its edge is not part of the chosen path, and $-t^e$ if it is. We will assume for simplicity that the graph is bi-connected.

A Truthful Implementation:

The following mechanism ensures that the dominant strategy for each agent is to report its true type t^e to the mechanism. When all agents honestly report their costs, the cheapest path is chosen: The output is obtained by a simple shortest path calculation. The payment p^e given to agent e is 0 if e is not in the shortest path and $p^e = d_{G|e=\infty} - d_{G|e=0}$ if it is. Here $d_{G|e=\infty}$ is the length of the shortest path which does not contain e (according to the inputs reported), and $d_{G|e=0}$ is the length of the shortest path when the cost of e is assumed to be zero (again according to the reported types).

Notice that the shortest path is indeed a minimization of the total cost. Also notice that the given mechanism is a VGC mechanism: $d_{G|e=\infty}$ corresponds to $h^i(t^{-i})$ and $d_{G|e=0}$ to $\sum_{j\neq i} v^j(t^j, o(t))$.

Many other graph problems, where agents are edges, and their valuations proportional to the edges' weights, can be implemented by a VGC mechanism. In particular minimum spanning tree and max-weight matching seem natural problems in this setting. A similar solution applies to the more general case where each agent holds some subset of the edges.

Open Problem: How fast can the payment functions be computed? Can it be done faster than computing n versions of the original problem? For the shortest paths problem we get the following equivalent problem: given a directed graph G with non-negative weights, and two vertices in it x, y. Find, for each edge e in the graph, the shortest path from x to y that does not use e. Using Disjktra's algorithm for each edge on the shortest path gives an $O(nm \log n)$ algorithm. Is anything better possible? Maybe $O(m \log n)$? For the similar problem with minimum spanning tree, it has been pointed out to us by Valerie King that the known fully dynamic algorithms (or alternatively the known sensitivity-analysis algorithms) for MST provide a nearly linear time solution.

4 Task Scheduling

In this section we analyze the task allocation problem. Subsection 4.1 formally presents the problem, subsection 4.2 gives a (weak) upper bound, subsection 4.3 provides our lower bounds, and finally in subsection 4.4 we exhibit a randomized solution that beats the lower bound.

4.1 The Problem

Definition 10 (Task Allocation Problem) *There are k tasks that need to be allocated to n agents. Each agent i's type is, for each task j, the minimum amount of time t^i_j it is capable of performing this task in. The goal is to minimize the completion time of the last assignment (the make-span). The valuation of an agent i is the negation of the total time it has spent on the tasks allocated to it.*

More formally:

- *The feasible outputs of the mechanism are all partitions $x = x^1 \ldots x^n$ of the tasks to the agents, where x^i is the set of tasks allocated to agent i.*

- *The objective function is $g(x, t) = \max_i \sum_{j \in x^i} t^i_j$.*

- *Agent i's valuation is $v^i(x,t^i) = -\sum_{j\in x^i} t^i_j$.*

We will consider both the exact and approximate versions.

Notation: We denote a direct revelation mechanism for the task scheduling problem by $m = (x, p)$, where $x = x(t)$ is the allocation algorithm and $p = p(t)$ the payment. (These are functions of the declared types.)

4.2 An Upper Bound

A simple, but not very good, approximation for the task scheduling problem is to minimize the total work done. It turns out that this approximation can be used as a basis for an approximation mechanism[7].

Definition 11 (MinWork Mechanism)

- **allocation:** *each task is allocated to the agent who is capable of doing it in a minimal amount of time (tasks with equal time declarations are allocated arbitrarily).*
- **payment:** *the payment for each agent i is defined as $p^i(t) = \sum_{j\in x^i(t)} \min_{i'\neq i}(t^{i'}_j)$ (i.e. for each task allocated to it, the agent is given payment equal to the time of the second best agent for this task).*

Theorem 4.1 *MinWork is a strongly truthful n-approximation mechanism for the task scheduling problem.*

Proof: We prove that the MinWork mechanism is strongly truthful and that it is an n-approximation.

Claim 4.2 *MinWork is strongly truthful .*

Proof: We will first show that MinWork belongs to the VGC family, and therefore, by theorem 3.1 it is truthful. The output is an allocation that maximizes the utilitarian function $\sum_{i=1}^n v^i(t^i, x)$; Let h^{-i} be $\sum_{j=1}^k \min_{i'\neq i} t^{i'}_j$, then $\sum_{i'\neq i} v^{i'}(t^{i'}, x) + h^{-i}$ is exactly the mechanism's payment function.

We now show that truth-telling is the only dominant strategy. We will show it for the case of a single task. The argument for $k > 1$ is similar. We note that a similar proof can be found in

[7]The mechanism can be viewed as auctioning each task separately in a Vickrey auction (Vickrey (1961)).

Vickrey (1961) for the analysis of the famous Vickrey auction. Let d denote the agents' declarations and t their real types. Consider the case where $d^i \neq t^i$ $(i = 1, 2)$. If $d^i > t^i$, then for d^{3-i} such that $d^i > d^{3-i} > t^i$,the utility for agent i is $t^i - d^i < 0$, instead of 0 in the case of truth-telling. A similar argument holds for the case of $d^i < t^i$. □

Claim 4.3 *MinWork is an n-approximation for the task scheduling problem.*

Proof: Let $opt(t)$ denote an optimal allocation. The proof follows immediately from the fact that $g(x(t), t) \leq \sum_{j=1}^{k} \min_i t_j^i$ and $g(opt(t), t) \geq \frac{1}{n} \cdot \sum_{j=1}^{k} \min_i t_j^i$. □

The theorem is an immediate outcome of claims 4.2 and 4.3. □

4.3 Lower Bounds

Due to the revelation principle (proposition 2.1) it suffices to prove the lower bound for truthful implementations. Thus for the rest of this section, $m = (x, p)$ is always assumed to be a truthful mechanism for the task scheduling problem.

4.3.1 Basic Properties of Truthful Implementations

We now formulate, in our settings, two basic observations from mechanism design. (Similar arguments can be found in Mas-Collel, Whinston and Green (1995) pp. 876-880.)

Proposition 4.4 (Independence) *Let t_1 and t_2 be type vectors, and i be an agent. If $t_1^{-i} = t_2^{-i}$ and $x^i(t_1) = x^i(t_2)$, then $p^i(t_1) = p^i(t_2)$.*

Proof: Without loss of generality assume that $p^i(t_1) < p^i(t_2)$. Then, if i's type is t_1^i, it is better off cheating, declaring t_2^i. A contradiction to the truthfulness. □

This proposition states that the payment offered to an agent does not depend on its type declaration (as long as the other agents' types and the allocation are fixed). The payment can thus be represented using the following well defined function.

Definition 12 *Let t be a type vector, i an agent. For a set X of tasks, we define the* price offered for X to agent i *as:*

$$p^i(X, t^{-i}) = \begin{cases} p^i(t'^i, t^{-i}) & \textit{if there exists } t'^i \textit{ s.t. } x^i(t'^i, t^{-i}) = X \\ 0 & \textit{otherwise} \end{cases}$$

Usually it will be more convenient to describe a mechanism by its price rather than by its payment function. Note that any function of the form $h^i(t^{-i})$ can be added to the payment of each agent i without changing its considerations. We therefore assume w.l.o.g. that the payment given to an agent is zero if no tasks are assigned to it.

Notation: Let i be an agent of type t^i, and let X be a set of tasks. We denote the time needed for i to perform all tasks of X, as $t^i(X) = \sum_{j \in X} t^i_j$.

Proposition 4.5 (Maximization) *For each type vector t and agent i,*

$$x^i(t) \in \arg\max_{X \subseteq \{1,\ldots,k\}} (p^i(X, t^{-i}) - t^i(X))$$

Proof:(sketch) Since $p^i(x^i, t^{-i}) - t^i(x^i))$ is agent i's utility, the above statement simply states that the mechanism has to maximize the agent's benefit. Otherwise the agent will do so itself, i.e. cheat as to get the maximum benefit! $\square$

We can now prove the main theorem of this subsection.

4.3.2 Basic Lower Bound

Theorem 4.6 *There does not exist a mechanism that implements a c-approximation for the task scheduling problem for any $c < 2$.*

Proof: We start with a lemma.

Notation: Let i be an agent, t a type vector, and A and B two disjoint sets of tasks. We define the *price difference* $\Delta^i(A, B)$ to be $p^i(A \bigcup B, t^{-i}) - p^i(A, t^{-i})$ (suppressing the type t).

Lemma 4.7 *Let t be a type vector and let $X = x^i(t)$. For each set $D \neq X$ of tasks the following inequalities hold:*

1. *If $D \subset X$ then $\Delta^i(D, X - D) \geq t^i(X - D)$.*
2. *If $D \supset X$ then $\Delta^i(X, D - X) \leq t^i(D - X)$.*
3. *otherwise, let $L = D \bigcap X$, then $\Delta^i(L, X - L) - t^i(X - L) \geq \Delta^i(L, D - L) - t^i(D - L)$*

Moreover, if a set Y of tasks satisfies these inequalities sharply for all D's, then $Y = X = x^i(t)$.

Proof: The fact that the above inequalities hold for $x^i(t)$ is an immediate consequence of proposition 4.5 (maximization) and the definition of the utility as $u^i = p^i(x^i(t), t^{-i}) - t^i(x^i(t))$. When the inequalities are strict, X is clearly the unique set of tasks that maximizes i's utility. $\square$

We prove the theorem for the case of two agents. For $n > 2$ we can reduce to this case by having the other agents be much slower than agents 1 and 2.

Notation: Let t be a type vector, i an agent and X a set of tasks. Let $\alpha > 0$ be a real number. We denote by $\hat{t} = t(X \xrightarrow{i} \alpha)$ the type obtained by

$$\hat{t}_j^{i'} = \begin{cases} \alpha & \text{if } i' = i \text{ and } j \in X \\ t_j^{i'} & \text{otherwise} \end{cases}$$

In the same manner we define $\hat{t} = t(X^1 \xrightarrow{i_1} \alpha_1, X^2 \xrightarrow{i_2} \alpha_2, \ldots)$ to be the result of a sequence of the above transformations.

Let $k \geq 3$, and t be the type vector defined by $t_j^i = 1$ for each agent i and task j. Without loss of generality we assume that $|x^1(t)| \leq |x^2(t)|$. Let $x = x^1(t)$ and let $\bar{x}$ denote its complement $(x^2(t))$.

Claim 4.8 *Let* $0 < \epsilon < 1$, $\hat{t} = t(x \xrightarrow{1} \epsilon, \bar{x} \xrightarrow{1} 1 + \epsilon)$. *Then* $x(\hat{t}) = x(t)$.

Proof: Since $n = 2$, it is enough to show that $x^1(\hat{t}) = x^1(t)$. As the type of agent 2 has not changed, the prices offered to agent 1 remain the same. For type t, $x^1(t)$ fulfills the inequalities of lemma 4.7. Thus, by inspection, they are strict when the type becomes $\hat{t}$, and therefore the allocation remains the same. $\square$

Assuming $|x^2(t)|$ is even, the lower bound follows since $g(x(\hat{t}), \hat{t}) = |x^2(\hat{t})| = |x^2(t)|$, but $g(opt(\hat{t}), \hat{t}) \leq \frac{1}{2} \cdot |x^2| + k \cdot \epsilon$ (for the allocation that gives agent 1, in addition to the original $x^1(t)$, half of agent 2's original allocation).

For the case of odd $|x^2(t)|$ it must be that $|x^2(t)| \geq 3$. We choose an arbitrary task $j \in x^2(t)$ and consider the type $\hat{t}(\{j\} \xrightarrow{2} \epsilon)$, which still yields the same allocation. $\square$

This lower bound is tight for the case of two agents. We conjecture that, in general, the upper bound is tight:

Conjecture 4.9 *There does not exist a mechanism that implements a c-approximation for the task scheduling problem with n agents for any* $c < n$.

Although we have not been able to prove this conjecture, we can show that it is correct for two natural special cases presented in the next subsection.

4.3.3 Tight Bounds for Special Cases

Definition 13 *A mechanism is called* additive *if for each agent i, type vector t and set X of tasks, $p^i(X, t^{-i}) = \sum_{j \in X} p^i(\{j\}, t^{-i})$.*

Theorem 4.10 *There does not exist any additive mechanism that solves the c-approximation problem for any $c < n$.*

Proof: Let $k \geq n^2$ and let $t^i_j = 1$ for each agent i and task j. Without loss of generality we assume that $|x^1(t)| \geq n$. Let $x = x^1(t)$ and let $\bar{x}$ denote its complement.

Claim 4.11 *Fix $0 < \epsilon < 1$ and let $\hat{t} = t(x^1 \xrightarrow{1} 1 - \epsilon, \bar{x} \xrightarrow{1} \epsilon)$. Then $x^1(\hat{t}) \supseteq x^1(t)$.*

Proof: Since t^2 has not changed, the prices offered to agent 1 remain the same. Clearly the price offered to agent 1 for x is strictly greater than the time $\hat{t}^1(x)$ required for it to perform x. Since the payment is additive, the set $x^1(\hat{t})$ which maximizes 1's utility must contain all the tasks in x. □

It follows that $g(x(\hat{t}), \hat{t}) \geq |x^1| \geq n$. Like in theorem 4.6, we can assume w.l.o.g. that $|x^1| = n$. The lower bound is then obtained since an optimal allocation would split these tasks among the n agents. □

Definition 14 *We say that a mechanism is* local *if for each agent i, type vector t and set X of tasks, the price $p^i(X, t^{-i})$ depends only on the agents' values on the tasks in X (i.e. $\{t^{l \neq i}_j | j \in X\}$).*

Theorem 4.12 *There does not exist a local mechanism that solves the c-approximation problem for any $c < n$.*

Proof: We start with a simple lemma that will enable us to turn the inequalities of lemma 4.7 into sharp ones.

Lemma 4.13 *For each type vector t and $\epsilon > 0$, there exists a type vector t' such that $\|t - t'\| < \epsilon$ and where the sets that maximize the agents' utility are unique for all agents.*

Proof:(sketch) The lemma is proved using a simple measure-theoretic argument. Let i be an agent, $A \neq B$ two sets of tasks. Because of the independence property (proposition 4.4), the following set has a zero measure on the type-space of agent i:

$$E^i(A, B, t^{-i}) = \{t^i | p^i(A, t^{-i}) - t^i(A) = p^i(B, t^{-i}) - t^i(B)\}$$

From this we obtain that for almost every type vector t', the inequalities in lemma 4.7 (for all agents) are strict. $\square$

Let $k \geq n^2$ and let $t^i_j = 1$ for each agent i and task j. By lemma 4.13, we assume w.l.o.g. that $x^i(t)$ uniquely maximizes i's utility for all agents i. Without loss of generality we assume that $|x^1(t)| \geq n$.

Claim 4.14 *Let $x = x^2(t)$ and $\hat{t} = t(x \xrightarrow{2} \epsilon)$ for some $0 < \epsilon < 1$. Then $x(\hat{t}) = x(t)$.*

Proof: Clearly $x^2(\hat{t}) = x^2(t)$. Consider another agent $i \neq 2$. The mechanism must allocate to agent i a set $x^i(\hat{t})$ that maximizes i's utility among all the sets X which are *disjoint* from $x^2(t)$. But since the mechanism is local, these prices have not changed from t to $\hat{t}$. Therefore $x^i(t)$ remains the unique set that maximizes i's utility. $\square$

By the same argument the allocation for the type $\hat{t} = t(x^2(t) \xrightarrow{2} \epsilon, \ldots, x^n(t) \xrightarrow{n} \epsilon)$ remains $x(t)$.

Like in theorem 4.6 we can assume that $|x^1(t)| = n$ and thus the lower bound is obtained since an optimal allocation will split these tasks among the n agents. $\square$

4.4 The Power of Randomization

In section 4.3 we showed that no mechanism can achieve a better than 2-approximation for the task scheduling problem. Here we show that *randomized* mechanisms can do better. The model of randomization that we use does not weaken the demands of dominant strategies at all: Although the agents choose their strategies without knowing the results of the random coin tosses, we require the strategy to be dominant for *all* possible tosses.

Definition 15 *(***A Randomized Mechanism***)* *A* randomized mechanism *is a probability distribution over a family $\{m_r | r \in I\}$ of mechanisms, all sharing the same sets of strategies and possible outputs.*

The outcome *of such a mechanism is a probability distribution over outputs and payments; the problem specification must specify what output distributions are required. For the case of optimization problems, the* objective function *on such a distribution is taken to be the expectation, i.e. $g(a, t) = E_{r \in I}(g(o_{m_r}(a), t))$.*

Definition 16 *(***Universally Dominant Strategy***)* *A strategy a^i is called* universally dominant *(in short, dominant) for agent i if it is a dominant strategy for every mechanism in the support* *of the randomized mechanism. A randomized mechanism is called* universally truthful *(in short, truthful) if truth-telling is a dominant strategy, and* strongly truthful *if it is the only one.*

We will design a strongly truthful randomized mechanism that achieves better performance than the deterministic lower bound. The randomized mechanism will be a distribution over *biased min work* mechanisms defined in figure 1.

Parameters: A real number $\beta \geq 1$ and a bit vector $s \in \{1,2\}^k$.

Input: The reported type vectors $t = (t^1, t^2)$.

Output: An allocation $x = (x^1, x^2)$, and a payment $p = (p^1, p^2)$.

Mechanism:
$x^1 \leftarrow \emptyset$; $x^2 \leftarrow \emptyset$; $p^1 \leftarrow 0$; $p^2 \leftarrow 0$.
For each task $j = 1...k$ do
  Let $i = s_j$ and $i' = 3 - i$
  If $t^i_j \leq \beta \cdot t^{i'}_j$
    Then $x^i \leftarrow x^i \bigcup \{j\}$; $p^i \leftarrow p^i + \beta \cdot t^{i'}_j$
    Else $x^{i'} \leftarrow x^{i'} \bigcup \{j\}$; $p^{i'} \leftarrow p^{i'} + \beta^{-1} \cdot t^i_j$

Figure 1: the biased min work mechanism (for two agents)

Lemma 4.15 *For all parameter values, the biased min work mechanism is strongly truthful .*

Proof: Since the overall utility of each agent can be described as the sum of the utilities aggregated in each step, it is enough to consider the case of $k = 1$. In this case the mechanism is equivalent to a weighted VGC (definition 9) with weights $\{1, \beta\}$ or $\{\beta, 1\}$ (depending on s_j). □

Definition 17 (The Randomly Biased Min Work Mechanism) *The* randomly biased min work mechanism *is the distribution on biased min work mechanisms given by $\beta = 4/3$, and a uniform distribution of $s \in \{1,2\}^k$.*

Theorem 4.16 *The randomly biased min work mechanism is a (polynomial time computable) strongly truthful implementation of a 7/4-approximation for task scheduling with two agents.*

The proof of the theorem is immediate from the following two lemmas.

Lemma 4.17 *The randomly biased min work mechanism is strongly truthful .*

This is immediate from lemma 4.15.

Lemma 4.18 *The allocation obtained by the randomly biased min work mechanism is a 7/4-approximation for the task scheduling problem.*

Comment: Our original analysis yielded a bound of 1.823. Daniel Lehmann (Lehmann (1999)) provided us with a tighter case analysis, improving the bound to 7/4. With Daniel's permission we include his refined analysis in our proof.

Proof: Let $opt(t)$ denote an optimal allocation algorithm. Let t_{opt} denote its make-span, and let t_{bmw} denote the (expected) make-span of the randomly biased min work mechanism.

We call a task j a k-task if one of the agents is considerably more efficient than the other on it (i.e. $t_j^1/t_j^2 > \beta$ or $t_j^2/t_j^1 > \beta$); otherwise we call it an l-task . Note that the mechanism allocates each k-task to the agent which is *efficient* on it, and *randomly* allocates the l-tasks.

Claim 4.19 *It is enough to consider the following case:*

1. *For each k-task, the efficiency discrepancy between the agents is arbitrarily close to β (therefore we shall assume that it equals β).*
2. *If opt allocates an l-task j to agent i, then $t_j^{3-i}/t_j^i = \beta$.*
3. *Under opt both agents have the same finishing time.*
4. *One of the agents is more efficient than the other on all k-tasks (w.l.o.g. let it be agent 1).*
5. *There are at most four tasks, where at most one k-task and at most one l-task is allocated by opt to each agent.*

Proof:

1. Since the mechanism always allocates each k-task j to the agent i which is more efficient on it, reducing t_j^{3-i} down to $\beta \cdot t_j^i$ can only help *opt* and leaves t_{bmw} unchanged.
2. If *opt* allocates an l-task j to agent i, then increasing t_j^{3-i} will not affect t_{opt} but will increase t_{bmw}.

3. Otherwise, w.l.o.g. assume that agent 1 finishes δ-time before agent 2. Adding an l-task j such that $t_j^1 = \delta$ and $t_j^2 = \beta \cdot \delta$ does not change t_{opt} but increases t_{bmw}.

4. Assume that there are two k-tasks a and b such that $t_a^2/t_a^1 = t_b^1/t_b^2 = \beta$. W.l.o.g. $t_a^1 \geq t_b^2$. If a is replaced by two k-tasks a' and a'' such that $t_a^i = t_{a'}^i + t_{a''}^i$ then t_{bmw} remains the same while t_{opt} can only decrease. In particular we can choose $t_{a'}^1 = t_b^2$.

   The mechanism allocates both tasks a' and b to the agent which is efficient on them. If *opt* is doing the same then clearly removing both a' and b can just increase the ratio t_{bmw}/t_{opt}. Obviously, opt cannot allocate a' to agent 2 and b to agent 1. Therefore it is enough to consider the case where *opt* allocates both tasks to one of the agents. One can verify that in this case replacing both tasks with equivalent l-tasks (i.e. l-tasks with the same computational times as the original ones) does not affect t_{opt} but will increase t_{bmw} by at least $\frac{\beta-1}{2} \cdot t_{a'}^1$.

5. Let a and b be two k-tasks that *opt* allocates to the same agent i. Recall that $t_a^{3-i}/t_a^i = t_b^{3-i}/t_b^i = \beta$. Clearly, replacing both tasks with a single task c such that $t_c^i = t_a^i + t_b^i$, does not affect *opt* nor the mechanism. We now consider the case where a and b are both l-tasks. Again, t_{opt} does not change as a consequence of such a replacement. We will show that t_{bmw} can only increase. Let Y be an allocation of all the tasks except a and b; let $t_{Y,a,b}$ denote the expected make-span when all other tasks are allocated according to Y and a and b are randomly allocated; let $t_{Y,c}$ denote the expected make-span when a and b are replaced by c which is allocated randomly. Clearly, it is enough to show that $t_{Y,a,b} \leq t_{Y,c}$.

   Let T^1 and T^2 denote the finishing times of both agents respectively when the allocation is Y. If one of the agents i finishes after the other regardless of how a and b are allocated, then clearly $t_{Y,a,b} = T^i + \frac{t_a^i + t_b^i}{2} = t_{Y,c}$. Otherwise, if agent i finishes last iff both a and b are allocated to it, then $t_{Y,a,b} = \frac{T^i + t_a^i + t_b^i}{4} + \frac{T^{3-i} + t_a^{3-i}}{4} + \frac{T^{3-i} + t_b^{3-i}}{4} + \frac{T^{3-i} + t_a^{3-i} + t_b^{3-i}}{4}$. Since $T^{3-i} \leq T^i + t_a^i + t_b^i$, we obtain that $t_{Y,a,b} \leq \frac{T^i + t_a^i + t_b^i}{2} + \frac{T^{3-i} + t_a^{3-i} + t_b^{3-i}}{2} = t_{Y,c}$. Finally w.l.o.g. assume that $t_a^i \geq t_b^i$ (for $i = 1, 2$) and consider the case where the agent to which a is allocated finishes last. In this case $t_{Y,a,b} = \frac{T^1 + t_a^1 + t_b^1}{4} + \frac{T^1 + t_a^1}{4} + \frac{T^2 + t_a^2 + t_b^2}{4} + \frac{T^2 + t_a^2}{4} \leq \frac{T^1 + t_a^1 + t_b^1}{2} + \frac{T^2 + t_a^2 + t_b^2}{2} = t_{Y,c}$.

□

Following the above claim we prove the lemma for the case of four tasks k_1, k_2, l_1, l_2, such that k_i and l_i denote the k-task and l-task which are allocated to agent i by *opt* (cases in which there are less than four tasks can be represented by zero times). The reduced case is described in figure 2.

	t_j^1	t_j^2	opt-alloc	bmw-alloc
k_1	a	$\beta \cdot a$	1	1
k_2	b	$\beta \cdot b$	2	1
l_1	c	$\beta \cdot c$	1	rnd
l_2	$\beta \cdot d$	d	2	rnd

Figure 2: the reduced case

Since both agents have the same finishing time in opt, $t_{opt} = a + c = \beta \cdot b + d$. We show that $\frac{t_{bmw}}{t_{opt}} \leq 7/4$ by considering three separate sub-cases.

Case 1: $a + b + \beta \cdot d \leq \beta \cdot c$.
Considering all four possible allocations of the mechanism we obtain that $t_{bmw} = 1/4 \cdot ((a+b+c+\beta \cdot d)+(a+b+c)+(\beta \cdot c)+(\beta \cdot c+d))$. Substituting $\beta = 4/3$ we get $t_{bmw} = 1/2 \cdot a + 1/2 \cdot b + 7/6 \cdot c + 7/12 \cdot d$ and one can verify that $t_{bmw} \leq 7/4 \cdot (a + c) = 7/4 \cdot t_{opt}$.

Case 2: Otherwise, $a + b + \beta \cdot d \geq \beta \cdot c$. Consider the case where $a + b \leq \beta \cdot c + d$.
In this case $t_{bmw} = 1/4 \cdot ((a + b + c + \beta \cdot d) + (a + b + c) + (a + b + \beta \cdot d) + (\beta \cdot c + d))$. Substituting β we get $t_{bmw} = 3/4 \cdot a + 3/4 \cdot b + 5/6 \cdot c + 11/12 \cdot d$ and it is not difficult to verify that $t_{bmw} \leq 7/4 \cdot (a + c) = 7/4 \cdot t_{opt}$.

Case 3: Otherwise, $a + b + \beta \cdot d \geq \beta \cdot c$ and $a + b \geq \beta \cdot c + d$. In this case $t_{bmw} = 1/4 \cdot ((a + b + c + \beta \cdot d) + (a + b + c) + (a + b + \beta \cdot d) + (a + b))$. Substituting β we get $t_{bmw} = a + b + c/2 + 2/3 \cdot d$ and again it can be verified that $t_{bmw} \leq 7/4 \cdot (4/3 \cdot b + d) = 7/4 \cdot t_{opt}$. □

This completes the proof of the theorem. □

5 Mechanisms with Verification

The basic mechanism design model assumes that each agent can follow any of its strategies, independently of its type. Thus the mechanism cannot use any "real-world" information about the agents. This is the norm in mechanism design and it models well the negotiation stage in which agents do nothing but communicate. In many settings in distributed computation though, one could take advantage of the fact that computers actually act (execute a task, route a message, etc.) to gain extra information about the agents' types and actions.

A simple type of modification to the model suggests itself: a problem definition may limit the set of strategies A^i available to each agent as a function of its type t^i. Many variants are possible, with different types of information available at different stages of the mechanism. In this paper we concentrate on what we feel is a very natural model. We distinguish between two stages of the mechanism: a declaration phase in which agents "talk" and which results in a decision (e.g. allocation), and then an execution phase where the agents actually execute the agreed output. The payments need only to be given *after* the execution. Intuitively we view the execution part as allowing the mechanism to verify in some sense the agents' declarations, and "punish" them for lying.

For the task scheduling problem we assume that by the end of the execution the mechanism **knows** the exact execution time of each task. A reformulation of the problems is introduced in section 5.2. We then (in sections 5.3 and 5.4) present a family of optimal mechanisms[8] for this problem. In section 5.5 we show that versions of our mechanism can operate under a limited budget and can guarantee that the profit for a truthful agent is always non-negative. Since these mechanisms require optimal scheduling algorithms they are computationally intractable. In section 5.6 we discuss polynomial-time mechanisms. We define a sub-case of the scheduling problem for which we present a polynomial-time approximation schema. The existence of a (better than n) polynomial time approximation mechanism for the general problem is left open.

5.1 Mechanisms with Verification

Definition 18 (Mechanism with Verification)

- *An agent's strategy a^i is composed of two separate parts: a declaration d^i and an execution e^i.*
- *Each declaration d^i is chosen by the agent, based on its type t^i, in an unrestricted manner.*
- *The decision k of the mechanism must be a function of just the declarations $d^1, \ldots, d^n$.*
- *The agent's execution e^i may depend on t^i as well as on k. The problem specification specifies, for each t^i, the possible $e^i()$'s an agent of type t^i may choose.*
- *The output of the mechanism is the result of the decision k and the agents' executions $e^1(k), \ldots, e^n(k)$. The output function $o(k, e)$ is a part of the problem specification.*
- *The output o, determines both the objective function $g(o, t)$ and the agents' valuations $v^i(t^i, o)$.*

[8] Although these mechanisms are presented for the scheduling problem, they can be generalized for many situations.

- *The payment p^i that the mechanism provides depends on both, the declarations $d^1, \ldots, d^n$ and the executions $e^1(k) \ldots e^n(k)$.*

Definition 19 *A mechanism with verification is called* truthful *if*

1. *The agents' declarations are simply to report their types.*
2. *For each agent i of type t^i, there is a dominant strategy of the form $a^i = (t^i, e^i())$.*

We say that the mechanism is strongly truthful *if it is the only dominant strategy.*

Note that by applying the revelation principle 2.1 to the declaration part, we can limit the discussion to mechanisms where the agents are simply requested to reveal their types.

Notation: We denote a mechanism with verification by a pair $m = (k(d), p(d, e))$, where $k()$ is the decision and $p()$ the payment function.

5.2 A Reformulation of Task Scheduling

Definition 20 (Task Scheduling with Verification) *The problem is the same as the task allocation problem (definition 10), except that the mechanism is allowed to pay to the agents* after *the tasks have been performed. We assume that the times which the tasks were actually performed in are* known *to the mechanism.*

More formally:

- *A feasible output of the mechanism is denoted by a pair $(x, \tilde{t})$, where $x = x^1, \ldots, x^n$ denotes the allocation of the tasks to the agents, and $\tilde{t} = \tilde{t}_1, \ldots, \tilde{t}_k$ denotes the actual times that they were performed in. If $j \in x^i(t)$, then it must be that $\tilde{t}_j \geq t^i_j$.*
- *A strategy for an agent is composed of two parts: a declaration of its type and an execution of the tasks allocated to it. An agent may lie or choose to perform any task j allocated to it, in any time $\tilde{t}_j \geq t^i_j$.*
- *The objective function is $g(x, \tilde{t}) = \max_i \sum_{j \in x^i} \tilde{t}_j$ (the make-span).*
- *Agent i's valuation is $v^i(x, \tilde{t}) = -\sum_{j \in x^i} \tilde{t}_j$.*
- *A mechanism is a pair (x, p) such that $x(t) = x^1(t), \ldots, x^n(t)$ is the allocation function, and $p(t, \tilde{t}) = p^1(t, \tilde{t}), \ldots, p^n(t, \tilde{t})$ is the payment.*

5.3 The Compensation-and-Bonus Mechanism

The *Compensation-and-Bonus Mechanism* is composed of an optimal allocation algorithm, together with a well chosen payment function. The payment function is the sum of two terms, one is called the compensation, and the other the bonus.

Definition 21 (Compensation) *The function*

$$c^i(t,\tilde{t}) = \sum_{j \in x^i(t)} \tilde{t}_j$$

is called the compensation function *for agent i.*

The bonus function uses the following notion:

Definition 22 (Corrected Time Vector) *Let i be an agent, x an allocation. Given the agents' declarations t and the vector of actual times $\tilde{t}$, we define the* corrected time vector *for agent i as:*

$$corr^i(x,t,\tilde{t})_j = \begin{cases} \tilde{t}_j & \text{if } j \in x^i \\ t^l_j & \text{if } j \in x^l \text{ and } l \neq i \end{cases}$$

We define $corr^*(x,t)$ of x and t to be the (unique) vector that satisfies $corr^*(x,t)_j = t^i_j$ for all i and $j \in x^i$.

Definition 23 (Bonus) *The function*

$$b^i(t,\tilde{t}) = -g(x(t), corr^i(x(t),t,\tilde{t}))$$

is called the bonus function *for agent i.*

The bonus is calculated according to the *declarations* of the other agents and the *actual* times that the agent performed its assignments in.

Definition 24
(Compensation-and-Bonus Mechanism) *The Compensation-and-Bonus mechanism is given by an optimal allocation algorithm with the payment functions $p^i(t,\tilde{t}) = c^i(t,\tilde{t}) + b^i(t,\tilde{t})$.*

Theorem 5.1 *The Compensation-and-Bonus mechanism is a strongly truthful implementation of the task scheduling problem.*

Proof: We show that the only dominant strategy for each agent is to reveal its true type and to execute its tasks in minimal time.

Claim 5.2 *The Compensation-and-Bonus mechanism is strongly truthful.*

Proof: Let i be an agent, t^i its type and let d^{-i} denote the declarations for the other agents (note that the allocation and bonus given to i depend on d^{-i} but not on the actual execution times of the others). Let $t = (d^{-i}, t^i)$. Observing that the utility of an agent equals its bonus, and that for every allocation x the bonus for an agent i is maximized when executing its assignments in minimal time, it is enough to show that $-g(x(t), corr^*(x(t), t)) >= -g(x(t'^i, d^{-i}), corr^*(x(t'^i, d^{-i}), t))$ for each t'^i. This is immediately implied by the *optimality* of the allocation algorithm.

Clearly, when an agent does not follow this strategy, there are circumstances where this will increase the make-span and therefore decrease the agent's bonus. Therefore, the above strategy is the only dominant one. □

When all agents follow their dominant strategies, the best possible make-span is obtained due to the optimality of the allocation algorithm. □

An Example

	j_1	j_2	j_3
A_1	**10**	30	**45**
A_2	100	**60**	100

Figure 3: a type matrix for two agents

Consider the type matrix in figure 3. Assume first that both agents are truthful. The optimal allocation in this case is $\{\{j_1, j_3\}\{j_2\}\}$ and the make-span is 60, therefore the bonus given to each agent is -60. Consider the case where agent 1 tries to "loose" j_3 declaring t_3^1 as 200. The "optimal" make-span therefore reduces to 100 and consequently the bonus for each agent reduces to -100. Similarly, when agent 1 tries to "gain" j_2 declaring for example t_2^1 to be 4, its bonus is reduced to -85. If agent one is "lazy" exceuting its tasks in 100 units of time instead of 55, then its bonus reduces from -60 to -100.

5.4 The Generalized Compensation-and-Bonus Mechanism

We now generalize the Compensation-and-Bonus mechanism:

Definition 25 (Generalized Compensation) *The function* $c^i(t,\tilde{t})$ *is called a* generalized compensation *for agent* i *if*

- $c^i(t,\tilde{t}) \le \sum_{j \in x^i(t)} \tilde{t}_j$ *for all* t *and* $\tilde{t}$.
- *Equality exists in case that the agent is truthful.*

Definition 26 (Generalized Bonus) *Let* $m^i(t^{-i}, w)$ *be any positive real-valued function, that is strictly monotonically increasing in* w. *The function* $b^i(t,\tilde{t}) = m^i(t^{-i}, -g(x(t), corr^i(t,\tilde{t})))$ *is called a* generalized bonus *for agent* i.

Definition 27 (Generalized Compensation and Bonus Mechanism) *A* Generalized Compensation and Bonus *mechanism is a pair* $m = (o, p)$ *where:*

- $o()$ *is an optimal allocation algorithm.*
- $p^i(t,\tilde{t}) = c^i(t,\tilde{t}) + b^i(t,\tilde{t})$, *where* $c^i()$ *and* b^i *are generalized compensation and bonus functions respectively.*

Arguments, similar to 5.1 lead to the following theorem:

Theorem 5.3 *The Generalized Compensation and Bonus mechanism is a strongly truthful implementation of the task scheduling problem.*

□

5.5 Budget Considerations

Theorem 5.3 allows a lot of freedom in the design of a generalized Compensation-and-Bonus mechanism. In this section we take advantage of this freedom in order to satisfy two additional requirements: participation constraints and budget limits (see Mas-Collel, Whinston and Green (1995) chapter 23 for a detailed discussion).

Definition 28 (Participation Constraints) *We say that a mechanism satisfies* participation constraints *if whenever an agent is truth-telling, its utility is non-negative. More formally: for each* t *and* $\tilde{t}$ *and for each agent* i, *if* $\tilde{t}_j = t_j$ *for all* $j \in x^i(t)$, *then* $p^i(t,\tilde{t}) + v^i(x(t),\tilde{t}) \ge 0$.

Note: If a truthful approximation mechanism satisfies the property that whenever an agent is not allocated any tasks then its payment is non-negative, it needs to satisfy participation constraints as well. (Because by declaring high enough values the agent can force the mechanism to allocate no tasks to it.)

Theorem 5.4 *There exists a strongly truthful mechanism that satisfies participation constraints for the task scheduling problem.*

Proof: We define $g^{-i}(t^{-i})$ to be the optimal make-span among the allocations which do not include agent i. We define the *contribution* of agent i as $cont^i(t^{-i}, \tilde{t}) = g^{-i}(t^{-i}) - g(x(t), corr^i(x(t), t, \tilde{t}))$. A generalized Compensation-and-Bonus mechanism where the bonus of each agent equals its contribution clearly satisfies participation constraints. □

So far we have not limited the amount of money that a mechanism can pay to the agents. When considering mechanisms with limited budget one cannot ignore the fact that such a mechanism may not be able to allocate the tasks at all. We therefore change the problem definition.

Definition 29 (Constrained Task Scheduling) *The problem is the same as in definition 20 except that there is another possible output $\perp$ (indicating that the tasks are not allocated). $v^i(\perp)$ is taken to be zero and $g(\perp)$ is taken to be ∞.*

Definition 30 (Limited Budget) *We say that a mechanism $m = (o, p)$ has a* limited budget *b if for any tuple of strategies $s = s^1, \ldots, s^n$, $\sum_i p^i(s) \leq b$. We call a revelation mechanism* b-constrained *if it satisfies participation constraints and has a limited budget b.*

Note that if $Y(t)$ is the allocation of a b-constrained mechanism then it must be that $\sum_i \sum_{j \in Y^i} t^i_j \leq b$.

Definition 31 *Let $m = (o, p)$ and $m' = (o', p')$ be two mechanisms for the same minimization problem. We say that m is* as good as *m' if for any dominant strategies $d = d^1, \ldots, d^n$ for the agents in m and $s = s^1, \ldots, s^n$ for the agents in m', $g(o(d)) \leq g(o'(s))$.*

Theorem 5.5 *Let $b, h > 0$. Then there exists a truthful mechanism m for the constrained task scheduling problem such that*

1. *m is $(b + h)$-constrained.*

2. If m' is b-constrained, then m is as good as m'.

Proof: Consider the following mechanism $m = (o, p)$:

- The output algorithm $o(t)$ finds an optimal allocation among the allocations y such that $\sum_i \sum_{j \in y^i} t^i_j \leq b$ (or outputs $\perp$ if none exists).
- Let $m()$ be a bounded strictly monotone increasing function such that $0 \leq m() < h/n$. We define the contribution $cont^i$ of the agent as in 5.4 (except that the algorithm and the objective function are different). We define the bonus $b^i()$ as $m(cont^i())$ and the payment to be given by $p^i = c^i() + b^i()$ where $c()$ is a compensation function. The payment is defined to be 0 if the output is $\perp$.

The total compensation is bounded by b and the total bonus by h, therefore the budget is bounded by $b + h$. Arguments, similar to 5.1 show that the mechanism is truthful and that the only dominant strategies for an agent i are to reveal the truth (when $t^i_j \geq b$ the agent may declare on any $d^i_j \geq b$) and to perform its tasks as efficient as it can.

Recalling that any b-constrained mechanism m' needs to choose an allocation z such that $\sum_i \sum_{j \in z^i} t^i_j \leq b$, the theorem is proved. □

5.6 Poly-Time Mechanisms

While the Compensation-and-Bonus mechanisms are optimal, note that they are intractable from a computational point of view due to their use of the exponential-time optimal allocation algorithm. One would be tempted to take a known polynomial-time approximation algorithm for the problem and base a mechanism upon it, obtaining a polynomial-time approximation mechanism. Unfortunately, this is not so simple and we do not know how to do this in general. In this section we first show that replacing the optimal allocation algorithm with a non-optimal approximation in the Compensation-and-Bonus mechanism does not preserve truthfulness. A similar argument can be made for the important case of VGC mechanisms (section 3.1). We then define a sub-case of the scheduling problem, where the number of agents is fixed and there are known bounds for the execution times t^i_j. This problem is still NP-hard and the lower bounds presented in sections 4.3.2 and 4.3.3 can be applied to it (with slightly weaker constants depending on the bounds). Nevertheless, for any $\epsilon > 0$ we are able to present a $1 + \epsilon$ polynomial approximation mechanism for this variant. Our approximation mechanism is based on a rounding technique developed by Horowitz and Sahni (1976).

Definition 32 *Let $x()$ be an allocation algorithm. The* Compensation-and-Bonus mechanism based on x *is the same as 5.3 except that the optimal algorithm is replaced by $x()$.*

Theorem 5.6 *Let $x()$ be a non-optimal approximation algorithm for task scheduling. Let $m = (x, p)$ be the Compensation-and-Bonus mechanism based on $x()$. Then m is not truthful.*

Proof: Assume by contradiction that it is truthful. For an allocation y and a type t, let $g(y, t)$ denote the make-span – $\max_i \sum_{j \in y^i} t^i_j$; let $opt(t)$ denote an optimal allocation.

Let t be a type such that $g(opt(t), t) < g(x(t), t)$, and let t'^1 be a type for agent 1 such that

$$t'^1_j = \begin{cases} t^1_j & \text{if } j \in opt^1(t) \\ \infty & \text{otherwise} \end{cases}$$

where ∞ stands for an arbitrary high value.

Claim 5.7 *Let $t' = t'^1, t^2, \ldots, t^n$. Then $g(x(t'), t') \geq g(x(t), t)$.*

Proof: Otherwise, in the case where agent 1's type is t^1, it would be more beneficial for it to "pretend" to be t'^1 (note that this cannot be verified!). This contradicts the truthfulness of the mechanism. □

Corollary 5.8 *Let s be a type such that*

$$s^i_j = \begin{cases} t^i_j & \textit{if } j \in opt^i(t) \\ \infty & \textit{otherwise} \end{cases}$$

Then $g(x(s), s) \geq g(x(t), t)$.

□

Since $g(x(s), s) \geq g(x(t), t) > g(opt(t), t) = g(opt(s), s)$, we obtain that $x(s) \neq opt(s)$. Thus there exists an agent who is allocated an ∞ job, in contradiction to the approximation ratio of $x()$. □

Definition 33 (Bounded Scheduling Problem) *The problem is the same as in definition 20, except that the number of agents n is fixed to a constant and there exist fixed $b > a > 0$ such that for all i, j $a \leq t^i_j \leq b$.*

The *rounding* algorithm presented in Horowitz and Sahni (1976) provides a $(1+\epsilon)$-approximation for bounded scheduling and runs in polynomial time. It basically works as follows: The entries t^i_j

are first rounded up to integer multiples of δ (a parameter chosen as a function of a and ϵ). It then exactly solves this rounded problem using dynamic programming (in polynomial time). The solution of the rounded problem is shown to be a $1+\epsilon$ approximation to the original one.

We will attach to this algorithm a carefully chosen payment function as to obtain our mechanism. The idea is to use the exact times for the compensation function, but the rounded ones for the bonus function.

Notation: For a vector t, let $\hat{t}$ denote the vector where all entries are rounded up to an integer multiple of δ. Denote also $\hat{g}(x, \tilde{t}) = g(x, \hat{\tilde{t}})$, where g is the make-span objective function.

Definition 34 (Rounding Mechanism) *The* rounding mechanism *is defined as follows:*

- *The allocation algorithm is the rounding algorithm of Horowitz and Sahni (1976) sketched above.*
- *The payment function is given by:* $p^i(t, \tilde{t}) = c^i(t, \tilde{t}) + b^i(t, \tilde{t})$, *where*

$$c^i(t, \tilde{t}) = \sum_{j \in x^i(t)} \tilde{t}_j$$

$$b^i(t, \tilde{t}) = -\hat{g}(x(\hat{t}), \hat{corr}^i(x(t), t, \tilde{t}))$$

The rounding mechanism compensates the agents according to their actual work, but computes the bonus according to the rounded declarations and execution times.

Theorem 5.9 *For every fixed $\epsilon > 0$ the rounding mechanism is a polynomial time mechanism with verification that truthfully implements a $1+\epsilon$ approximation for the bounded task scheduling problem.*

Proof:(sketch) When the types and the actual computation times are rounded, $\hat{g}$ is exactly the make-span, and the rounding algorithm is optimal. Arguments, similar to those in 5.1, therefore show that the only dominant strategies for agent i are to declare on a type t'^i such that t'^i and t^i have the same rounded value, and to execute its tasks such that after rounding, $\hat{corr}^i(x(t), t, \tilde{t})$ equals $\hat{corr}^*(x(t), t)$. Clearly, when all agents follow such strategies, the result is a $1+\epsilon$ approximation. In particular, truth-telling is among the dominant strategies. $\square$

6 Conclusions and Further Research

In this paper we suggested a framework for studying optimization problems that involve selfish participants. We studied a representative task scheduling problem under two main models: a basic mechanism design based model and a model that allows more information to be incorporated into the mechanism. Under the assumptions of the basic model we showed that the problem cannot be approximated within a factor of $2 - \epsilon$. Then, under the second model assumptions, we introduced several novel mechanisms including optimal, constrained optimal and polynomial-time approximation mechanisms. We have also shown that worst case behavior can be improved using randomness without weakening the "game-theoretic" requirements of the mechanism.

We believe that our work is only a first step towards understanding the notion of algorithmic cooperation among selfish agents. There are clearly many open problems and research directions, and we are far from a situation where we could design, analyze, and implement protocols and algorithms that directly take into account the participants' differing goals.

We divide the basic issues for further research into three main categories: questions directly coming out of our work, game-theoretic extensions to our model and distributed computation issues.

Several questions directly stem from our work. For example, there are large gaps between the upper and lower bounds for both task scheduling without verification and for poly-time task scheduling with verification.

Many game-theoretic extensions to our model are possible. For example one may consider different settings (e.g. repeated games), different solution concepts (e.g. Bayesian-Nash), and different assumptions (e.g. partial verification).

Finally, in this work we have treated the mechanism as a black box, and have not considered how its function is actually carried out in a distributed manner. A whole set of open problems comes from trying to "open up" this black box, and analyze the steps taken in implementing the mechanism from a distributed point of view. For example when communication costs are considered, even the revelation principle breaks up; non complete network topology may be exploited by the agents to extract information about others and to cooperate; cryptography may be introduced and distributed handling of the payments may be considered.

Acknowledgments: We thank Dov Monderer, Motty Perry and Ori Regev for helpful discussions at various stages of this work. We thank Daniel Lehmann, Ofer Neyman, Dana Peer, Inbal Ronen and Moshe Tennenholtz for comments on earlier drafts of this paper.

References

[1] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, pages 17–33, 1971.

[2] Eithan Ephrati and Jeffrey S. Rosenschein. The clarke tax as a concensus mechanism among automated agents. In *Proceedings of the national Conference on Artificial Intelligence*, pages 173–178, July 1991.

[3] Donald F. Ferguson, Christos Nikolaou, and Yechiam Yemini. Economic models for allocating resources in computer systems. In Scott Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation.* World Scientific, 1995.

[4] J. Green and J.J. Laffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica*, pages 427–438, 1977.

[5] T. Groves. Incentives in teams. *Econometrica*, pages 617–631, 1973.

[6] R. M. Harstad, Rothkopf M. H., and Pekec A. Computationally manageable combinatorial auctions. Technical Report 95-09, DIMACS, Rutgers university, 1995.

[7] Dorit S. Hochbaum. *Approximation Algorithms for NP-Hard Problems.* PWS publishing company, 1997.

[8] Ellis Horowitz and Sartaj Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the association for computing machinery*, pages 317–327, 1976.

[9] Brnardo A. Huberman and Tad Hogg. Distributed computation as an economic system. *Journal of Economic Perspectives*, pages 141–152, 1995.

[10] Y.A Korilis, A. A. Lazar, and A. Orda. Architecting noncooperative networks. *IEEE Journal on Selected Areas in Communication (Special Issue on Advances in the Fundamentals of Networking)*, 13(7):1241–1251, September 1991.

[11] Leslie Lamport, Robert E. Shostak, and C. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, pages 382–401, 1982.

[12] A.A. Lazar and N. Semret. The progressive second price auction mechanism for network resource sharing. In *8th International Symposium on Dynamic Games*, Maastricht, The Netherlands, July 1998.

[13] Daniel Lehmann. Private communication, 1999.

[14] Jan Karel Lenstra, David B. Shmoys, and Eva Tardos. Approximation algorithms for scheduling unrelated parallel machines. In *28th Annual Symposium on Foundations of Computer Science*, pages 217–224. IEEE, 1987.

[15] Nathan Lineal. Game theoretic aspects of computing. In *Handbook of Game Theory*, volume 2, pages 1339–1395. Elsevier Science Publishers B.V, 1994.

[16] A. Mas-Collel, Whinston W., and J. Green. *Microeconomic Theory*. Oxford university press, 1995.

[17] J. McMillan. Selling spectrum rights. *Journal of Economic Perspectives*, pages 145–162, 1994.

[18] Market design inc. Web Page: http://www.market-design.com.

[19] Dov Monderer and Moshe Tennenholtz. Distributed games. To appear in Games and Economic Behaviour.

[20] Noam Nisan. Algorithms for selfish agents. In *To appear in Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science*, Trier, Germany, March 1999.

[21] Noam Nisan and Amir Ronen. Algorithmic mechanism design (extended abstract). In *The Thirty First Annual ACM symposium om Theory of Computing(STOC99)*, pages 129–140, May 1999.

[22] M. J. Osborne and A. Rubistein. *A Course in Game Theory*. MIT press, 1994.

[23] Christos H. Papadimitriou. Computational aspects of organization theory. In *Proceedings of the 1996 European Symposium on Algorithms*. Springer LNCS, 1996.

[24] Christos H. Papadimitriou and Mihalis Yannakakis. On the value of information in distributed decision-making (extended abstract). In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, pages 61–64, Montreal, Quebec, Canada, August 1991.

[25] Christos H. Papadimitriou and Mihalis Yannakakis. Linear programming without the matrix (extended abstract). In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 121–129, San Diego, California, May 1993. ACM.

[26] Kevin Roberts. The characterization of implementable choise rules. In Jean-Jacques Laffont, editor, *Aggregation and Revelation of Preferences*, pages 321–349. North-Holland, 1979. Papers presented at the first European Summer Workshop of the Econometric Society.

[27] Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT Press, 1994.

[28] Tuomas W. Sandholm. Limitations of the vickrey auction in computational multiagent systems. In *Proceedings of the Second International Conference on Multiagent Systems (ICMAS-96)*, pages 299–306, Keihanna Plaza, Kyoto, Japan, December 1996.

[29] S. Shenkar, Clark D. E., and Hertzog S. Pricing in computer networks: Reshaping the research agenda. *ACM Computational Comm. Review*, pages 19–43, 1996.

[30] Yoav Shoham and Katsumi Tanaka. A dynamic theory of incentives in multi-agent systems (preliminary report). In *Proceedings of the Fifteenth International Joint Conferences on Artificial Intelligence*, pages 626–631, August 1997.

[31] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, pages 8–37, 1961.

[32] W.E. Walsh and M.P. Wellman. A market protocol for decentralized task allocation: Extended version. In *The Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS-98)*, 1998.

[33] W.E. Walsh, M.P. Wellman, P.R. Wurman, and J.K. MacKie-Mason. Auction protocols for decentralized scheduling. In *Proceedings of The Eighteenth International Conference on Distributed Computing Systems (ICDCS-98)*, Amsterdam, The Netherlands, 1998.