Algorithms For Rational Agents

Amir Ronen*

School of Computer Science and Engineering, The Hebrew University of Jerusalem.

Abstract. Many recent applications of interest involve self-interested participants. As such participants, termed agents, may manipulate the algorithm for their *own* benefit, a new challenge emerges: The design of algorithms and protocols that perform well when the agents behave according to their own self-interest.
This led several researchers to consider computational models that are based on a sub-field of game-theory and micro-economics called mechanism design.
This paper introduces this topic mainly through examples. It demonstrates that in many cases selfishness can be satisfactorily overcome, surveys some of the recent trends in this area and presents new challenging problems.
The paper is mostly based on classic results from mechanism design as well as on recent work by the author and others.

1 Introduction

A large part of research in computer science is concerned with protocols and algorithms for inter-connected 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. While this assumption seems obvious for "traditional" applications programmed by a single entity, it cannot be taken for granted when considering, for example, applications which are intended to operate on the Internet. Many of these applications involve self-interested participants (e.g. individual users and private companies). Such participants have their *own* goals and preferences and are likely to follow their own self interest. Algorithmic problems that stem from such applications give rise to a new and exciting challenge: The design of algorithms and protocols that perform well when the participants behave selfishly.

Applications that involve self-interested parties emerge from several domains: Participants in an electronic trade pursue their personal profit, users of a wide area network want their own traffic to be optimized and software agents are self-interested by definition. Problems that stem from such applications are essentially different from traditional algorithmic problems. To make this difference concrete, consider an imaginary system aimed to utilize resources on the Internet:

* This research was supported by grants from the Israeli academy of science and the Israeli ministry of science. Email: amiry@cs.huji.ac.il

Ideally, such a system would function like a regular distributed system. It would efficiently allocate CPU-intensive jobs to CPU-servers, store data in computers with free disk space, extract necessary information from various available databases and combine the services of the many software packages found on the Internet. Beyond many difficult coordination and optimization problems which need to be overcome, such a system faces a major novel challenge: All these resources *belong* to self-interested parties (such as private companies or institutions) and there is no a-priory reason that these parties will allow the system to freely use them, in particular when they are costly. The system may, thus, need to provide some motivation for these owners to "play along". Supplying such motivation is likely to involve some sort of payment. Payment, real or virtual, carries however problems of its own. A payment model which is not carefully designed can easily lead into a situation where manipulating the system's algorithms and protocols can significantly increase the profits of the participants (for example, participants may pretend that they are not capable of performing tasks which are not profitable enough). Such manipulations might severely damage the efficiency of such a system, preventing it to accomplish the purposes which it was designed for. As we shall see, the algorithmic and the monetary aspects of protocols for such systems are tightly coupled. Any solution for an algorithmic problem that involves selfish participants should simultaneously deal with these two aspects. Solving such problems is therefore a difficult and exciting challenge!

Mechanism design is a sub field of game theory and micro economics which deals with the design of protocols for rational agents. Generally, a mechanism design problem can be described as the task of selecting from a collection of feasible games, a game which will yield desirable results for the designer. Specifically, the theory has focused on problems where the goal is to satisfactorily aggregate privately known preferences of several agents towards a "social choice".

An application of this theory within the framework of computer science is by no means straight-forward. There are two main reasons for that:

Different goals Traditionally, mechanism design theory was developed for applications that arise from economics. Naturally, problems that stem from computer science applications have different goals and assumptions.

Computational complexity Mechanism design theory ignores the computational aspect of the protocol. This turns essentially all solutions proposed by this theory to be impractical for computationally complex problems.

In recent years, problems which lie on the border of computer science and mechanism design were studied by several researchers. Numerous fundamental result however are yet to be discovered.

This paper is intended to serve as an introduction for this topic. It demonstrates that in many cases selfishness can be satisfactorily overcome, surveys some of the recent trends in this area and presents new challenging problems. The paper is mostly based on classic results from mechanism design as well as on recent work by the author and others.

The rest of this paper is organized as follows: Sections 2 and 3 exemplify mechanism design problems that arise in computer science and demonstrates that they can be handled. Section 4 formally defines mechanism design problems and their valid solutions, describes the celebrated VCG mechanisms and shows some of their properties. Section 5 discusses the computational aspects of mechanisms focusing on the important problem of combinatorial auctions. Section 6 considers problems which cannot be addressed by the standard tools of mechanism design theory. Section 8 surveys recent work on the intersection between computer science and mechanism design and section 9 describes some fundamental open issues.

2 A Simple Routing Problem

We begin with a problem which has a very simple combinatorial structure:

Consider a communication network consisting of $k > 1$ parallel links between two points s and t. Each link e is owned by a different agent (e.g. a company) and the cost c_e of sending a message along this link is *privately known* to its owner.

A natural requirement from a "traditional" network is to be efficient, i.e. to send each message from s to t along the cheapest link. Clearly, such efficiency cannot be guaranteed in a network where resources are privately owned. We would like however, to construct a protocol that will reach efficiency under reasonable assumptions on the agents' behavior.

2.1 Some Naive Solutions

First, consider a "traditional" algorithmic approach for such a problem: The algorithm first requests from each agent to report its cost. It then chooses the link with the minimal declared cost and routes the message along that link.

Although this approach is "correct" from an algorithmic perspective, it is likely to fail. Clearly, a rational agent will try to avoid being chosen by declaring a very high cost. Since such a strategy has nothing to do with the real cost of the agent, the chosen link will be arbitrary.

It seems therefore unavoidable that a mechanism (a protocol) for such a problem will also be able to *pay* the agents. Such payment however needs to be done carefully.

Consider for example a mechanism which is identical to the previous one except that it pays the chosen agent a fixed amount (say \$100) per message. Such a mechanism may have two potential problems: First, all agents with costs which are higher than \$100 are likely to declare arbitrarily high values. As this may include all agents, a problem similar to the previous might occur. Second, agents with costs which are lower than \$100 are likely to declare arbitrarily small values. This is likely to reduce the efficiency of the network.

An auction may seem a better idea: each agent will declare her cost d_e , the mechanism will select the link with the lowest declared cost and pay the corresponding agent d_e units of currency. Note however that an agent can gain money only if she declares a cost which is higher than her actual cost. Her optimal declaration depends heavily on the declarations of the others. As even the costs of the other agents are not known to her, it is hard to give a convincing prediction of what will happen[1].

Another possible solution is the following multi-rounded protocol: In each round j the mechanism declares a payment p_j starting from a low value and incrementing it. The protocol stops when the first agent accepts the offer. This agent will be payed the p_{j*} units of currency where $j*$ denotes the last round. Although such a protocol is likely to make the right decision and route the message through the cheapest link, it might use a lot of communication. In the next subsection we'll see that we can achieve the same result in a single round!

2.2 A Good Solution

Consider the following solution (hereby called a mechanism) to our routing problem: Each agent is first required to declare its cost. Let d_e denote the declaration of e's agent. Note that it may be falsified. The mechanism selects the link with the minimal declared cost (ties are broken arbitrarily) and pays the owner of this link the minimum among the declared costs of the other agents, i.e. $\min_{e' \neq e} d_{e'}$.

Let us examine some of the properties of this mechanism:

Truthfulness It is not difficult to see that it is *always* for the benefit of an agent to declare her *true* cost to the mechanism. In a game theoretic language, truth-telling is a dominant strategy. In the literature, a mechanism with this property is called *truthful* or *incentive compatible*. One possible way to prove the truthfulness of this mechanism is to observe that the selected agent is payed an amount that equals the *maximal* (supremum) declaration that will still allow her to win. This can be thought of as if the mechanism offers the agent a payment of 0 if she is not selected and a payment of $\min_{e' \neq e} d_{e'}$ if she is. These two payments do not depend on the agent's declaration. The only thing one must verify is that the agent's benefit is maximized when she declares her true cost. This is obvious since a (truthful) agent is selected iff her costs are lower than the maximal amount that allows her to be selected, i.e. iff her costs are lower than her payment. A rational agent will therefore report her *true* cost to the mechanism.

Participation constraints and zero normalization Another desirable property of such a mechanism is that the profit of a truthful agent is *guaranteed* to be non-negative. In many cases the agent's participation is voluntary and therefore this property is important. In the literature, a mechanism with this property is said to satisfy *participation constraints*. We shall say that a mechanism is *zero normalized* if an agent who is not selected by the mechanism is always payed zero. In principal, this property implies participation

[1] Analysis of such situations under various assumptions do exist in the economic and game theoretic literature.

constraints except for some degenerate mechanisms. From reasons, similar to the truthfulness of the proposed mechanism, it is also zero normalized and in particular satisfies participation constraints.

Specification compatibility The fact that our mechanism is truthful and satisfies participation constraints has a strong influence on the agent's behavior. A rational agent will choose to cooperate with the mechanism and to reveal her true cost to it. When the agents act rationally, our mechanism makes the *right decision* and selects the link with the minimal cost. We shall call a mechanism *specification compatible* if when the agents report their true input (cost in our case), the problem's specification is met.

The proposed mechanism is likely to make the right decision despite the fact that the "input" of the problem is not known to it and that it is exposed to agents' manipulations!

Definition 1. (good mechanism) *We say that a mechanism is* good *if it is truthful, zero normalized, satisfies participation constraints and is specification compatible.*

Note that the suggested mechanism is very simple in two perspectives. First, the agents' dominant strategies are straight-forward. Second, it is a single rounded protocol. Throughout this paper we shall mostly seek for good mechanisms.

3 A More Complicated Routing Problem

We now complicate the example in section 2 and consider a general network topology. Formally:

We have a communication network modeled by a directed graph G, and two special nodes s and t in it. Each link e is owned by a different agent and the cost c_e of sending a message along this link is *privately* known to its owner. The goal of the mechanism is to find the cheapest path from s to t. We will assume for simplicity that the graph is bi-connected.

Note that unlike the previous problem, each possible solution may involve several agents. Despite that, we can construct a good mechanism for this problem as well.

3.1 A Good Mechanism

The mechanism first requires each agent to report its cost to it. Let us denote the declaration of an agent that corresponds to a link e by d_e. According to these declarations, the mechanism selects the cheapest path from s to t. The trick is in the payment function: agent e is given 0 if e is not in the chosen path and $p_e = d_{G|e=\infty} - d_{G|e=0}$ if it is. Here $d_{G|e=\infty}$ denotes the cost of the cheapest path which does not contain e (according to the inputs reported), and $d_{G|e=0}$ is the cost of the cheapest path when the cost of e is assumed to be zero (again according to the reported types).

It is a good exercise to prove the correctness of this mechanisms directly. In fact, the mechanism is a private case of *VCG* mechanisms. Their properties will be proven in section 4.2.

3.2 Further Comments

The proposed mechanism solves this problem in a central fashion requiring all agents to report directly to it. For many applications, a real *distributed* mechanism is necessary. This is a major open issue which gives rise to many open problems. For example, what is the influence of the network topology on the agents' strategies? How good can a mechanism perform when only local decisions are allowed? Etc.

Many other natural open questions can be asked on such a network. In particular, several problems emerge when considering settings in which messages are coming *stochastically* into the network. This affects both the requirements from the mechanism and the agents' strategies.

Finally, the calculation of the payments of the suggested mechanism hinders an interesting algorithmic problem: Is it possible to do better than calculating the payment of each agent on the chosen path separately?

4 Mechanism Design

The problems that were presented in the previous sections differ from usual algorithmic problems in two main aspects. Firstly, the problems' input was not accessible to the mechanism but was privately held by the participants. Secondly, while the goal of the mechanism was to maximize the efficiency of the network, each participant had a totally different goal which was to maximize her own profit.

Similar problems has been studied for several decades by economists and game theorist within the framework of mechanism design theory. Generally, a mechanism design problem could be described as task of selecting from a collection of feasible games one that will yield results which are desired by the designer. Specifically, the theory focuses on various variants of the canonical problem presented in section 4.1. An introduction to this field can be found at [18, chapter 23].

As we shall see, the previous examples can all be described as sub-cases of this canonical problem. However, many additional questions emerge when applying mechanism design theory to applications that stem from computer science applications.

4.1 Mechanism Design Problems And Solutions

In this section we shall formally define the canonical problem of mechanism design and its valid solutions.

Definition 2. (The Canonical Problem) *A* mechanism design problem *is described by the following:*

1. *A finite set O of* allowed outputs.
2. *Each agent $i = (1, \ldots, n)$ has a real function $v^i(o \in O)$ called her* valuation *or* type. *This is a quantification of her benefit from each possible output o in terms of some common currency. v^i is* **privately** *known to agent i. The space V^i of all possible valuation functions is called the* type space *of the agent.*
3. *If the mechanism's output is o and in addition the mechanism hands an agent p^i units of this currency, then her* utility *u^i equals*[2] *$v^i(o) + p^i$. This utility is what the* **agent** *aims to optimize.*
4. *The goal of the* **mechanism** *is to select an output $o \in O$ that maximizes the* total welfare *$g(v, o) = \sum_i v^i(o)$.*

Let us try to present the example of section 3 within the framework of the canonical problem. The set of allowable outputs contains all paths from s to t. The valuation of agent e from each possible output o, equals $-c_e$ if e is in the chosen path o, or 0 otherwise. The utility of each agent equals the sum of her valuation from the chosen output and her payment. The goal of the mechanism is to select a path with minimal total cost. Clearly, such a path maximizes the total welfare. This problem can therefore be presented as a sub-case of the canonical problem.

In a direct revelation mechanism, the participants are simply asked to reveal their types to the mechanism. Based on these declarations the mechanism then computes the output o and the payment p^i for each of the agents.

Definition 3. (A mechanism) *A* (direct revelation) mechanism *is a pair $m = (k, p)$ such that:*

- *The* output function *$k(.)$ accepts as input a vector $w = (w^1, \ldots, w^n)$ of declared valuation functions and returns an output $k(w) \in O$.*
- *The* payment function *$p(w) = (p^1(w), \ldots, p^n(w))$ returns a real vector. This is the payment handed by the mechanism to each of the agents.*

A revelation mechanism computes its output according to the type declarations of the agents. As agents may **lie** to the mechanism, it should be carefully designed such that it will be for the *benefit* of each agent to reveal her true type to the mechanism.

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

Definition 4. (truthful mechanism) *A mechanism is called* truthful *if truth-telling is a dominant strategy. I.e. for every agent i of type v^i and for every type declaration w^{-i} for the other agents, the agent's utility is* maximized *when she declares her real valuation function v^i.*

[2] This is called the quasi-linearity assumption.

A simple well known observation called the revelation principle [18, page 871] greatly facilitates the analysis of mechanism design problems. It states that in order to prove or disprove the existence of good mechanisms for a given problem, one can consider only truthful revelation mechanisms.

4.2 VCG mechanisms

Arguably the most important result in mechanism design is the observation that the canonical problem can be solved by a class of mechanisms called *VCG* (named after Vickrey, Clarke and Groves) [36, 1, 8]. Intuitively, these mechanisms solve the canonical problem by identifying the utility of truthful agents with the declared total welfare.

Definition 5. (VCG mechanism) *A mechanism* $m = (k, p)$ *belongs to the* VCG *family if:*

- $k(w)$ *maximizes the total welfare according to* w.
- *The payment is calculated according to the* VCG formula*:* $p^i(w) = \sum_{j \neq i} w^j(k(w)) + h^i(w^{-i})$ *(*$h^i()$ *is an arbitrary function of* w^{-i}*).*

Proof. Assume by contradiction that the mechanism is not truthful. Then there exists an agent i of type v^i, a type declaration w^{-i} for the other agents, and $w^i \neq v^i$ such that $v^i(k((v^i, w^{-i}))) + \sum_{j \neq i} w^j(k((v^i, w^{-i}))) + h^i(w^{-i}) < v^i(k((w^i, w^{-i}))) + \sum_{j \neq i} w^j(k((w^i, w^{-i}))) + h^i(w^{-i})$. Let $o = k((v^i, w^{-i}))$ denote the chosen output when the agent is truthful and let $o' = k((w^i, w^{-i}))$. The above inequality implies that $g((v^i, w^{-i}), o) < g((v^i, w^{-i}), o')$. This contradicts the optimality of k. □

It is worth notify that weighted versions of this method are possible as well (e.g. [28] [26]). To date, *VCG* is the only general known method for the construction of truthful mechanisms. Roberts [28] showed that under certain conditions, VCG mechanisms are indeed the only truthful mechanisms. These conditions however are not satisfied for many natural problems.

Clarke's Mechanism: Clarke's mechanism is a VCG mechanism which satisfies participation constraints and zero-normalization. (It is not feasible to every mechanism design problem, but it is for most of the natural ones.) The mechanism chooses $h^i(w^{-i})$ to be the maximal welfare that can be achieved if agent i would choose not to participate. Formally if there exists a type $\underline{v}^i$ such that for each v^{-i}, $g((\underline{v}^i, v^{-i}), k((\underline{v}^i, v^{-i}))) \leq g((v^i, v^{-i}), k((v^i, v^{-i})))$, then the mechanism defines $h^i(w^{-i}) = g((\underline{v}^i, v^{-i}), k((\underline{v}^i, v^{-i})))$. For the example of section 3 we can define $\underline{v}^i$ as an infinite cost. In this case the output algorithm never selects a path that goes through this agent's link. The mechanism satisfies participation constraints as the total utility of a truthful agent i equals $g((v^i, v^{-i}), k((v^i, v^{-i}))) - g((\underline{v}^i, v^{-i})) \geq 0$. It is not difficult to see that the zero normalization condition is also satisfied. The reader is welcome to verify that all the good mechanisms presented so far are Clarke mechanisms.

We shall comment again that an efficient computation of the agents' payments is an interesting algorithmic challenge.

5 The computational aspect

The theory of mechanism design does not take into consideration the computational complexity of the mechanism. In recent years, mechanisms have become quite complicated, requiring implementation on computer systems. Consequently, the necessity of a theory that will regard both the computational and the game theoretic aspects of mechanisms became clear to many researchers. In this section we bring as an example, the important problem of combinatorial auctions and briefly survey works that are trying to cope with its computational difficulty.

5.1 Combinatorial Auctions

The problem of combinatorial auctions has been extensively studied in recent years (see e.g. [16] [32] [7] [9] [24]). The importance of this problem is twofold. First, several important applications rely on it (e.g. the FCC auction sequence [19] that raised Billions of dollars). Second, it is a generalization of many other problems of interest, in particular in the field of electronic commerce.

The problem: A seller wishes to sell a set S of items (radio spectra licenses, electronic devices, etc.) to a group of agents who desire them. Each agent i has, for every subset $s \subseteq S$ of the items, a number $v^i(s)$ that represents how much s is worth for her. We assume that $v^i(.)$ is privately known to the agent.

We take two standard additional assumptions on the type space of the agents:

No externalities The valuation of each agent depends only on the items allocated to her, i.e. $\{v^i(s)|s \subseteq S)\}$ completely represents the agent's valuation.

Free disposal Items have non-negative values. I.e if $s \subseteq t$ then $v^i(s) \leq v^i(t)$. Also $v^i(\phi) = 0$.

Note that the problem allows items to be complementary, i.e. $v^i(S \bigcup T) \geq v^i(S) + v^i(T)$ or substitutes, i.e. $v^i(S \bigcup T) \leq v^i(S) + v^i(T)$ (S and T are disjoint). For example a buyer may be willing to pay $200 for T.V set, $150 for a VCR, $450 for both and only $200 for two VCRs.

When an agent's payment is p^i for a set of items s^i, her overall utility is $p^i + v^i(s^i)$. Note that p^i is always non-positive. This utility is what each agent tries to optimize. For example, an agent prefers to buy a $1000 valued VCR for $600 gaining $400 to buying a $1500 valued VCR for $1250.

In a VCG mechanism for such an auction, the participants are first required to reveal their valuation functions to the mechanism. The mechanism then computes, according to the declarations of the agents, an allocation s that maximizes the total welfare. The payment for each of the agents is calculated according to the VCG formula.

Consider however the computational task faced by such a mechanism. After the types are declared, the mechanism needs to select, among all possible allocations, one that maximizes the total welfare. This problem is known to be

NP-Complete. Therefore, unless the number of agents and items is very small, such a mechanism is computationally infeasible. Note that even the problem of finding an allocation that approximates the optimal allocation within a reasonable factor is NP-Complete (under the common complexity assumption that $RP \neq Co - NP$, see e.g. [32, 16]).

5.2 Coping With The Computational Complexity Of Combinatorial Auctions

Several researchers in recent years have considered the computational difficulty of combinatorial auctions. Many researchers focused on the algorithmic side of the problem [32, 7, 9, 24, 35]. These efforts include various tractable sub-cases where additional limitations where assumed on the agents' valuations and also several heuristics. A truthful mechanism which does not fall into the VCG category was constructed for a sub-case of combinatorial auction at [16].

A recent paper [27] by Nisan and the author studies VCG mechanisms where the optimal algorithm is replaced by by a computationally tractable approximation algorithm or heuristic. The paper shows that it is impossible to construct, in this method, truthful computationally feasible mechanisms that produce reasonable results for combinatorial auctions as well as for many cost minimization problems. Since VCG is the only general method known to date for the construction of truthful mechanisms, there is not much hope for solving mechanism design problems by truthful mechanisms. The paper introduces a slight relaxation of the truthfulness called feasible truthfulness. It shows that under reasonable assumptions on the agents, it is possible to turn *any* VCG-based mechanism into a feasibly truthful one, using an additional appeal mechanism. The resulting mechanism also satisfies participation constraints and zero normalization.

The proposed mechanism may also be used to overcome another problem that emerges from combinatorial auctions – that a full description of the agents' valuation may also be intractable. Thus, it is not clear how to construct an "interface" between the agent and the mechanism [24]. For details see [29].

The results of [27] however have not yet been checked experimentally.

6 Beyond The Canonical Problem: Task Scheduling

The problems that we have presented so far are all sub-cases of the canonical problem and henceforth can be addressed by VCG mechanisms. For many problems that stem from computer science applications this is not likely to be the case. First, in many cases, in particular for task allocation problems, the designer's goal is dependent, not on agents' valuations but on other private parameters such as her capability to perform various tasks. Secondly, even when such a dependency does exist, the objective of the designer may not be to maximize the total welfare.

Nisan and the author studied [26] (a preliminary version appeared at [25]), the following representative problem:

Task Scheduling: 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_j^i in which she is capable of performing this task. 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 she has spent on the tasks allocated to her.

The results of this paper refer to three different models – the "classical" model of mechanism design, a model where the mechanism can use randomization and a model where additional natural verification information is available to the mechanism. In particular the paper shows that when additional verification information is available, the problem can be *optimally* solved. This result remains true even without the (artificial) assumption that the agent's valuation equals the negation of her working time.

Many problems were left open in this paper. In particular it was conjectured that no classical mechanism can approximate this problem by a factor better than the number of agents. However, only a basic lower bound of 2 was proven. The limitations of the model that allows verification are also not known.

7 Revenue Considerations

So far we have treated money only as a mean of motivating the agents to behave well. For many applications however, revenue is a major concern. Very little is currently known about the revenue of truthful mechanisms. In this section we demonstrate that many beautiful open questions can be asked even on a very simple problem – the allocation of a single object. This section is based on [29].

The Problem: A *single object allocation* problem is a mechanism design problem of the following form:

- There are $n+1$ possible outputs. An output $i > 0$ means that the object is allocated to agent i. 0 means that the object is not allocated to any agent.
- The type of each agent is any strictly positive number.

We shall call a mechanism for this problem *agent-compatible* if it is truthful and satisfies zero-normalization and participation constraints. We shall first characterize these mechanisms.

Definition 6. (proper function) *An output function $x : R^n_{++} \rightarrow \{0, \dots, n\}$ is called* proper *if for every type vector $t = t^1, \dots, t^n$, and an agent i such that $i = x(t)$, if $s^i \geq t^i$ then $i = x((s^i, t^{-i}))$.*

In other words if an agent wins the object with type t^i she will keep winning when her type increases.

Definition 7. (implied payment) *Let x be a proper function. We say that a payment function p is* implied *by x, if $-p^i(t)$ is 0 when $i \neq x(t)$ or equals $\inf\{s^i | i = x((s^i, t^{-i}))\}$ when $i = x(t)$.*

In other words i pays zero if she looses or the minimal type that enables her to win the object. It is not difficult to see that the infimum exists.

Theorem 1. *A mechanism for this problem is agent-compatible iff it is of the form $m = (k, p)$ where:*

- $k(.)$ *is proper.*
- $p(.)$ *is implied by* k.

$\square$

Given a probability distribution on the type space, there are several interesting open questions that emerge. Here are a few examples:

- Does a polynomial algorithm that chooses a mechanism that maximizes the expected revenue exist?
- Given also a risk parameter $0 < r < 1$ and a probability $\hat{p}$, is it possible to find a mechanism that in probability at least $\hat{p}$ generates a revenue of at least $r \cdot \max_i t^i$?
- For the above questions: Each agent-compatible mechanism defines a boolean function of the type space (1 means good, e.g. close to the maximum possible revenue, and 0 means bad). Is the resulting family learnable in the PAC sense?

8 Work On The Border Of Computer Science And Mechanism Design

Problems on the border of computer science and mechanism design were studied by several researchers in recent years. This section briefly surveys these works.

Many researchers studied the problem of combinatorial auctions. This line of research is surveyed in section 5.1. Other cases where mechanism design models were applied to computational problems include resource and task allocation problems [25, 39, 38], communication networks [5, 14, 13], multi-agent systems [30, 40, 4] and others [37, 15, 6, 34, 17]. Several aspects of mechanism design were recently studied in [22, 21, 15, 33]. The computational aspect of mechanisms for cost sharing was studied at [5]. This recent paper investigates two ingredients absent from this work – a distributed model and budget balance requirement. These two fundamental aspects are surely to play a significant role in future research in this area. The implementation of mechanism design theory for Internet environments may require new assumptions. This issue is discussed at [31, 21].

Another line of research which is related to mechanism design are models which are based on markets [6, 10, 20, 23, 23, 3]. The main difference between these models and mechanism design is that instead of doing strategic manipulations, the participants are assumed to "honestly response to prices" (i.e. behave as price-takers).

Problems that stem from electronic commerce have a tight relationship with mechanism design and game theory and economics in general. Several papers on these topics can be found at [11, 2].

Finally, works in which the author was involved were surveyed in sections 5.1 and 6.

9 Further Research

This paper demonstrates that in many cases it is possible to design protocols that perform well despite the selfishness of the participants. However, there are numerous directions for further research. I mention some of the most basic ones: The computational power of strategy-proof mechanisms is far from being understood, even in the classical model. Probabilistic models and further relaxation of the concept of dominant strategies are yet to be explored. Concrete connections between mechanism design and computational complexity are yet to be discovered. Moreover, in practice, mechanisms will have to handle additional issues like irrationality and collusive behavior of the agents. Issues like revenue maximization, risk, and budget balance are very important to many applications. In a lot of cases, mechanisms will have to be carried out in a "real" distributed environment. Finally, solution concepts other than dominant strategy implementation (e.g. Bayesian Nash) may be considered.

I am certain that all these issues and a lot of others will provide researchers with many beautiful research problems in the years to follow!

Acknowledgments: I would like to thank Noam Nisan and Inbal Ronen 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. *Proceedings of the first ACM conference on electronic commerce (EC99)*, 1999.
3. Bernardo Huberman (ed.). *The Ecology of Computation.* Elsevier Science Publishers/North-Holland, 1988.
4. Eithan Ephrati. *Planning and consensus among autonomous agents.* PhD thesis, Hebrew University of Jerusalem, Departement of Computer Science, 1993.
5. Joan Feigenbaum, Christos Papadimitriou, and Scott Shenker. Sharing the cost of multicast transmissions. In *Thirty-Second Annual ACM Symposium on Theory of Computing (STOC00)*, May 2000.
6. 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.
7. Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *IJCAI-99*, 1999.
8. T. Groves. Incentives in teams. *Econometrica*, pages 617–631, 1973.
9. R. M. Harstad, Rothkopf M. H., and Pekec A. Computationally manageable combinatorial auctions. Technical Report 95-09, DIMACS, Rutgers university, 1995.
10. Brnardo A. Huberman and Tad Hogg. Distributed computation as an economic system. *Journal of Economic Perspectives*, pages 141–152, 1995.

11. *Proceedings of The first international conference on information and computation economies (ICE-98)*, 1998.
12. Paul Klemperer. Auction theory: a guide to the literature. *Journal of economic surveys*, pages 227–286, 1999.
13. 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.
14. Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *STACS 99, the 16th Annual Symposium on Theoretical Aspects of Computer Science*, March 1999.
15. Ron Lavi and Noam Nisan. Competitive analysis of online auctions. To appear.
16. Daniel Lehmann, Liadan O'Callaghan, and Yoav Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. In *ACM Conference on Electronic Commerce (EC-99)*, pages 96–102, November 1999.
17. Kevin Leyton-Brown, Moshe Tennenholtz, and Yoav Shoham. Bidding clubs: Institutionalized collusion in auctions. To appear.
18. A. Mas-Collel, Whinston W., and J. Green. *Microeconomic Theory*. Oxford university press, 1995.
19. J. McMillan. Selling spectrum rights. *Journal of Economic Perspectives*, pages 145–162, 1994.
20. Welleman Michael. Market-oriented programming. Web Page: http://ai.eecs.umich.edu/people/wellman/MOP.html.
21. Dov Monderer and Moshe Tennenholtz. Distributed games. To appear in Games and Economic Behaviour.
22. Dov Monderer and Moshe Tennenholtz. Optimal auctions revisited. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.
23. Miller M.S. and Drexler K. E. *The Ecology of Computation*, chapter Markets and Computation: Agoric open systems. North Hollan, 1988.
24. Noam Nisan. Bidding and allocation in combinatorial auctions. To appear.
25. 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.
26. Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behaviour*, 2000. To appear.
27. Noam Nisan and Amir Ronen. Computationally feasible vcg mechanisms. Submitted to the Second ACM conference on electronic commerce (EC00), 2000.
28. 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.
29. Amir Ronen. *Solving optimization problems among selfish agents*. PhD thesis, School of Computer Science and Engineering, The Hebrew University of Jerusalem, 2000.
30. Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT Press, 1994.
31. 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.
32. Tuomas W. Sandholm. Approaches to winner determination in combinatorial auctions. *Decision Support Systems*, to appear.

33. Y. Shoham and Tennenholtz M. Rational computation and the coomunication complexity of auctions. To appear.
34. 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.
35. Moshe Tennenholtz. Some tractable combinatorial auctions. In *the national conference on artificial intelligence (AAAI-2000)*, 2000.
36. W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, pages 8–37, 1961.
37. Nir Vulkan and Ken Binmore. Applying game theory to automated negotiation. To appear.
38. 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.
39. 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.
40. Gilad Zlotin. *Mechanisms for Automated Negotiation among Autonomous Agents*. PhD thesis, Hebrew University of Jerusalem, Departement of Computer Science, 1994.