Computationally Feasible VCG Mechanisms

Noam Nisan* Amir Ronen†

Abstract

One of the major achievements of mechanism design theory is the family of truthful (incentive compatible) mechanisms often called VCG (named after Vickrey, Clarke and Groves). When applying VCG mechanisms to complex mechanism design problems such as combinatorial auctions a problem emerges: even finding optimal outcomes is computationally intractable. A striking observation is that if the optimal outcome is replaced by the results of computationally tractable approximation algorithms or heuristics then the resulting mechanism (termed VCG-based) is no longer necessarily truthful!

The first part of this paper considers this problem in depth and shows that it is almost universal. Specifically, we prove that essentially all reasonable approximations or heuristics for combinatorial auctions as well as a wide class of cost minimization problems yield non-truthful VCG-based mechanisms.

The second part of this paper proposes a method for handling this non-truthfulness. We introduce a notion of *feasible truthfulness* that captures the limitation on agents imposed by their own computational limits. We then show 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.

1 Introduction

The theory of mechanism design may be described as studying the design of protocols under the assumption that the participants behave according to their own goals and preferences and not necessarily as "instructed" by the protocol. This theory has been traditionally used in economic settings such as auctions of various kinds (for an introduction see e.g. [12, 18]). Recently, with the emergence of the Internet as *the* distributed computing platform, the

*School of Computer Science and Engineering, The Hebrew University of Jerusalem. This research was supported by grants from the Israeli academy of science and the Israeli ministry of science. Email: noam@cs.huji.ac.il

†School of Computer Science and Engineering, The Hebrew University of Jerusalem. This research was supported by grants from the Israeli academy of science and the Israeli ministry of science. Email: amiry@cs.huji.ac.il

theory of mechanism design has been applied in computational settings such as distributed resource and task allocation [16, 27, 26], communication networks [3, 9], multi-agent systems [21], and others.

The canonical mechanism design problem can be described as follows: A set of rational agents need to collaboratively choose an outcome o from a finite set O of possibilities. Each agent i has a privately known valuation function $v^i : O \to R$ quantifying the agent's benefit from each possible outcome. The agents are supposed to report their valuation functions $v^i(\cdot)$ to some centralized mechanism that chooses an outcome o that maximizes the total welfare $\sum_i v^i(o)$.

The main difficulty is that agents may choose not to reveal their true valuations but rather report carefully designed lies in an attempt to influence the outcome to their liking. The tool that the mechanism uses to motivate the agents to reveal the truth is monetary payments – these payments are to be designed in a way that ensures that rational players always reveal their true valuations – making the mechanism, so called, incentive compatible or truthful.

There is only one general technique known for designing such a payment structure, sometimes called the generalized Vickrey auction [25], the Clarke pivot rule [1] the Groves mechanism [7], or, as we will, VCG. In certain senses this payment structure is unique [6, 20].

In recent years mechanisms have become quite complicated, requiring implementation on computer systems. Cases in point include combinatorial auctions [23, 4, 8, 11, 24, 14, 15, 2, 13], where multiple items are concurrently sold in an auction, as well as various computational task and resource allocation problems [16, 27, 26, 10, 3]. For many of these applications the space of possible outcomes is huge and finding an outcome that maximizes the declared total welfare is computationally infeasible (NP-complete). In these cases any realistic mechanism will not be able to obtain the optimal outcome.

A natural general approach for the development of mechanisms for complex problems would be to use a computationally feasible approximation algorithm or heuristic for obtaining a near-optimal outcome. The payments in such an approach would presumably be derived by applying the VCG payment rules to the underlying algorithm. We term such mechanisms *VCG-based.*

The starting point of this paper is the surprising observation, noticed already by some researchers ([11, 17]), that VCG-based mechanisms are not necessarily truthful! I.e. if the choice of outcome is sub-optimal, then applying the VCG payment rules will not necessarily motivate the agents to reveal the truth – rational agents may lie taking advantage of quirks in the outcome choice algorithm.

The first part of this paper exams this phenomena in depth and shows that it is near universal: essentially *all* reasonable VCG-based mechanisms are *not* truthful. We first point our attention at combinatorial auctions and characterize the class of truthful VCG-based mechanisms for this problem. This characterization yields, in particular, the following corollary. Call an allocation algorithm for combinatorial auctions reasonable if whenever an item is valued by only a single agent, then this agent receives that item.

Theorem: Any reasonable VCG-based mechanism for combinatorial auctions is not truthful (unless it uses the – computationally intractable – optimal allocation algorithm).

We then study a family of problems termed "cost minimization allocation problems". We call an algorithm for such a problem degenerate if it produces results which are arbitrarily far from the optimal.

Theorem: Any non-degenerate VCG-based mechanism for any cost minimization allocation problem is not truthful (unless it uses the optimal allocation algorithm).

We then examine carefully why VCG-based mechanisms are non-truthful, and observe that the only reason for an agent to lie about her valuation is to "help" the algorithm to improve the overall result. However, if finding a better outcome is hard, then we would not expect an agent to be computationally able to do so! This motivates us to define a feasible notion of truthfulness, a notion that takes into account the computational limits of the agents.

In the second part of this paper we first define this "feasible" notion of truthfulness. We then introduce a variant on VCG-based mechanims, called the second-chance mechanism[1] and show that under reasonable assumptions, any such mechanism is indeed "feasibly truthful".

Our notion of truthfulness is based on the view that each agent has some *strategic knowledge* and that her range of actions is determined by this knowledge. This knowledge may

[1]The mechanism is patent pending.

represent the computational limitations of the agent or some structural limitations on her information or behavior. We say that an action is feasibly dominant for the agent if her strategic knowledge does not contain awareness of *any* circumstances where another action is better for her. We say that a mechanism is feasibly truthful if the truth is feasibly dominant for all agents.

Given *any* algorithm for the corresponding optimization problem we define the *second-chance* mechanism based on it. This mechanism is a modification of the VCG-based mechanism where in addition to their type declarations, the agents are allowed to submit *appeal functions* as well. An appeal function is an encapsulation of the agents' algorithmic ability to help the mechanisms' underlying outcome determination algorithm by means of input modification.

We show that under reasonable assumptions on the agents, the addition of this appeal mechanism is enough to ensure feasible truthfulness. Specifically we prove several variants of theorems of the following schema. The different theorems differ from each other in the exact definition of computational limitation.

Theorem Schema: If all agents' strategic knowledge functions are computationally limited and the outcome determination algorithm is computationally limited then the second chance mechanism is feasibly truthful. Furthermore, the appeal functions are all computationally limited, and the mechansim's' outcome is at least as good as that of the underlying algorithm. The resulting mechanism satisfies participation constraints as well.

We present our work in the context of VCG-mechanisms. However our results are more general. In particular all they can be applied to the compensation and bonus mechanism [17] and to weighted versions of the VCG method.

The rest of this paper is organized as follows: In section 2 we introduce our model and notations. Limitations of truthful VCG-based mechanisms are studied in section 3. In section 4 we introduce the second-chance mechanism and show some of its properties.

2 Preliminaries

In this section we formally present our model. We attempt, as much as possible, to use the standard notions from both mechanism design and computational complexity. We focus on dominant strategy implementation under quasi-linear environments.

In subsection 2.1 we describe mechanisms and mechanism design problems[2]. In subsection 2.2 we describe the celebrated VCG mechanisms and present our VCG-based mechanisms.

In subsection 2.3 we discuss the computational burden of VCG mechanisms and briefly describe some fundamental notions of computational complexity.

As an example, we formulate, in subsection 2.4, the important problem of combinatorial auctions.

2.1 Mechanism design problems

Definition 1 (Utilitarian mechanism design problem) *A* (utilitarian) 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 its* valuation *or* type. *This is a quantification of its 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 the agent p^i units of this currency, then her* utility *u^i equals*[3] *$v^i(o) + p^i$. 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 *$g(v, o) = \sum_i v^i(o)$.*

An example for such a problem can be found in section 2.4.

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.

[2] In this paper we discuss only what are called utilitarian problems. Without loss of generality we limit ourselves to direct revelation mechanisms.

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

Definition 2 (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*[4] *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 (e.g. if* $p^i = -2$ *then the agent pays two units of currency to the mechanism).*

A revelation mechanism computes its output according to the type declarations of the agents. As they 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 3 (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*.*

As an example consider the famous Vickrey auction [25]: A seller wishes to sell one item in an auction. There are n buyers, each privately knowing her valuation v^i for this item. In a Vickrey auction each of the buyers is simply asked for her valuation. The item is allocated to the buyer with the highest bid for the price equal the second highest bid. The reader may verify that this mechanism is truthful, i.e. that it is always for the agent's benefit to declare her true type v^i. Another example for a truthful mechanism can be found in section 2.4.

2.2 VCG-based mechanisms

In this subsection we present the celebrated VCG mechanisms. Intuitively these mechanisms solve utilitarian problems by identifying the utility of truthful agents with the declared total welfare.

[4] We do not consider the issue of how to represent the valuations.

Definition 4 (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}).*

Theorem 2.1 *([7]) A VCG mechanism is truthful.*

□

It is worth notifying that weighted VCG mechanisms are possible as well (see e.g. [20] [17]).

For many applications, the task of finding an output $k(w)$ that maximizes the total welfare is computationally infeasible. In this paper we consider mechanisms where the optimal algorithm is replaced by a sub-optimal but computationally feasible one.

Definition 5 (VCG-based mechanism) *Let $k(w)$ be an algorithm that maps type declarations into allowable outputs. We call $m = (k(w), p(w))$ a* VCG mechanism based on k *if p is calculated according to the* VCG formula*: $p^i(d) = \sum_{j \neq i} w^j(k(w)) + h^i(w^{-i})$ (where $h^i()$ is an arbitrary function of w^{-i}).*

Obviously, a VCG-based mechanism that is based on an optimal algorithm is a VCG mechanism.

2.3 Computational considerations in mechanism design

Computer scientists make a sharp distinction between two major classes of computational problems. The first class is called P and contains all the easy problems. I.e. problems that can be solved in time polynomial in the bit-size of the input. The second class is called NP-Complete and contains problems which are provably hard. To date, no sub-exponential algorithm for any NP-Complete problem has been found. Unfortunately, many problems of interest (e.g. finding an allocation that maximizes the total welfare in a combinatorial auction) are NP-Complete. Fortunately, for many of these problems, various polynomial time heuristics and approximation algorithms are available. An introduction to the theory of computational complexity can be found in [5] [19].

We define these two principal notions in terms of mechanisms:

Definition 6 *A mechanism* (k, p) *is called* polynomial time computable *if both* $k(w)$ *and* $p(w)$ *run in polynomial time (using a standard encoding of* w*).*

Note that a VCG-based mechanism is polynomial iff its output algorithm is polynomial. We sometimes call polynomial algorithms and mechanisms computationally feasible.

Definition 7 *A mechanism design problem is called* NP-complete *if the problem of finding an output that maximizes the total welfare is* NP*-complete.*

2.4 Example: Combinatorial auctions

The problem of combinatorial auction has been extensively studied in recent years (see e.g. [11] [23] [4] [8] [15]). The importance of this problem is twofold. First, several important applications rely on it (e.g. the FCC auction [13]). 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, T disjointed). 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 payment is p^i for a set of items s^i then her overall utility is $p^i + v^i(s^i)$. Note that here the payment is 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. By theorem 2.1, the utility $u^i = p^i + v^i(s^i)$ of each of the agents is maximized when she reveals her true valuation to the mechanism. When all agents are truthful, the mechanism maximizes the total welfare.

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. [23, 11]). Nevertheless, various heuristics and tractable sub-cases have been analyzed in the literature [23] [4] [8] [15] [24].

3 Limitations of truthful VCG-based mechanisms

In this section we study the limitations of truthful VCG-based mechanisms. In subsection 3.1 we characterize these mechanisms for important problem of combinatorial auctions (see subsection 2.4). This characterization precludes the possibility of obtaining truthfulness by applying VCG rules to many of the proposed heuristics for combinatorial auctions, e.g. the greedy algorithms in [11] and [15]. Moreover we show that any truthful non-optimal VCG-based mechanism for combinatorial auctions suffers from abnormal behavior. In subsection 3.2 we show that for many natural cost minimization problems, any truthful VCG-based mechanism is either optimal or produces results which are arbitrarily far from the optimal. As a result, when such a problem is computationally intractable, any truthful computationally feasible VCG-based mechanism will produce unreasonable results.

3.1 Truthful VCG-based mechanisms for combinatorial auctions

In this section we characterize the class of truthful VCG-based mechanisms for combinatorial auctions (subsection 2.4).

Definition 8 *Let $k(w)$ be an algorithm that maps type declarations into allowable outputs. Let V' be a subspace of the space of all possible types $V \stackrel{df}{=} \prod_{i=1}^{n} V^i$. Let $\mathcal{O}$ denote the range of k at V', i.e. $\mathcal{O} = \{k(w) | w \in V'\}$. We say that k is* maximal in its range at V' *if for every type $w \in V'$, $k(w)$ maximizes g over $\mathcal{O}$. We say that k is* maximal in its range *if it is maximal in its range at V.*

As an example consider an algorithm for combinatorial auctions that allocates all the items (the set S) to the agent with the highest valuation $v^i(S)$. Clearly, this computationally efficient algorithm is maximal in its range . The result of this algorithm is never worse than $1/n$ and also $1/|S|$ times the optimal result (n denotes the number of the agents).

Proposition 3.1 *A VCG-based mechanism with an output algorithm that is maximal in its range is truthful.*

Proof: Such a mechanism is a VCG mechanism where the set of allowable outputs is the range of its output algorithm. By theorem 2.1 such a mechanism is truthful. □

We show that this almost characterizes the class of truthful VCG-based mechanisms for the combinatorial auction problem.

Notation: We let $\tilde{V}$ denote the space of all types $v = (v^1, \ldots, v^n)$ such that for any two different allocations x and y, $g(v, x) \neq g(v, y)$. (Recall that $g(.)$ denotes the total welfare.)

It is not difficult to see that $\tilde{V}$ contains almost all the types, i.e. $V - \tilde{V}$ is zero measured.

Theorem 3.2 *If a VCG-based mechanism for the combinatorial auction problem is truthful then its output algorithm is maximal in its range at $\tilde{V}$.*

Proof: Assume by contradiction that $m = (k, p)$ is truthful but k is not maximal in its range at $\tilde{V}$. Without loss of generality assume that $p^i(w) = \sum_{j \neq i} w^j(k(w))$ (i.e. we let $h^i()$ in the VCG formula to be zero). The utility of a truthful agent i, when the declarations of the other agents are w^{-i}, equals $g((v^i, w^{-i}), k(v^i, w^{-i}))$.

Let $\mathcal{O}$ denote the range of k at $\tilde{V}$ and let $v \in \tilde{V}$ be a type such that $k(v)$ is not optimal over $\mathcal{O}$. Let $y = opt_{\mathcal{O}}(v)$. Among the optimal allocations in $\mathcal{O}$ we also require that $y = (y^1, \ldots, y^n)$ allocates all the items to the agents, i.e. $\bigcup_i y^i = S$. Note that this is possible due to the free disposal assumption. Finally let $w \in \tilde{V}$ be a type such that $y = k(w)$.

Define a type z by

$$z^i(s) = \begin{cases} v^i(s) & \text{if } s \not\supseteq y^i \\ \alpha & \text{if } s \supseteq y^i \end{cases}$$

where α stands for a sufficiently large number. In other words, each agent i strongly desires the set y^i. We assume that $z \in \tilde{V}$, otherwise we could add sufficiently small "noise" $\epsilon^i(s)$ to z such that all the following claims remain true.

Lemma 3.3 $y = k(z)$.

Proof: Define a sequence of type vectors by:

$$\begin{aligned} w_0 = & \quad (w^1, \ldots, w^n) \\ w_1 = & \quad (z^1, w^2, \ldots, w^n) \\ \vdots & \\ w_n = & \quad (z^1, \ldots, z^n) \end{aligned}$$

We assume that $w_j \in \tilde{V}$ for all j. It is not difficult to see that z could be defined in a way that guarantees this.

Claim 3.4 $k(w_1) = y$.

Proof: Assume by contradiction that this is false. From the definition of $\tilde{V}$ we obtain that $g(w_1, k(w_1)) \neq g(w_1, y)$. Consider the case where agent 1's type is z^1 and the types of the others are $w^2, \ldots, w^n$. By declaring w^1, agent 1 can force the algorithm to decide on y. Therefore it must be that $g(w_1, k(w_1)) > g(w_1, y)$. In particular since α is large, it must be that $k^1(w^1) \supseteq y^1$. Thus, $\alpha + \sum_{j=2}^n w^j(k(w_1)) > \alpha + \sum_{j=2}^n w^j(y)$. As, due to the free disposal assumption, $w^1(k^1(w^1)) \geq w^1(y)$, we obtain that $w^1(k^1(w^1)) + \sum_{j=2}^n w^j(k(w_1)) > w^1(y) + \sum_{j=2}^n w^j(y)$ i.e. $g(w_0, k(w^1)) > g(w_0, y)$. Therefore when the type of agent 1 is w^1, she is better off declaring z^1 in contradiction to the truthfulness of the mechanism. $\square$

Similarly we obtain that $k(w_j) = y$ for all j. This completes the proof of lemma 3.3. $\square$

Consider the following sequence of type vectors:

$$\begin{aligned} v_0 = & \quad (v^1, \ldots, v^n) \\ v_1 = & \quad (z^1, v^2, \ldots, v^n) \\ \vdots & \\ v_n = & \quad (z^1, \ldots, z^n) \end{aligned}$$

Claim 3.5 *For all v_j, y maximizes g on $\mathcal{O}$.*

Proof: y maximizes g for v_0. Due to the free disposal assumption, an optimal allocation for v_j allocates y^i for every agent $i \leq j$. It must also to allocate the rest of the items optimally among the rest of the agents. This is also done according to y. □

Claim 3.6 $k(v_{n-1}) = y$.

Proof: Otherwise, when agent n's type is v^n, she is better off declaring z^n obtaining $g(v_{n-1}, y)$. This contradicts the truthfulness of the mechanism. □

Similarly, we obtain that $k(v_0) = y$. A contradiction. This completes the proof of theorem 3.2. □

This result characterize the output algorithms that could be incorporated in a VCG-based mechanisms accept maybe a measure zeroed subset of the types. We can now give a complete characterization:

Corollary 3.7 *Consider a VCG-based mechanism for a combinatorial auction with an output algorithm k. If the mechanism is truthful then there exists an output algorithm $\tilde{k}$, maximal in its range, such that for every v, $g(v, k(v)) = g(v, \tilde{k}(v))$.*

Proof: Let $g_k(v) \stackrel{df}{=} g(v, k(v))$. It is not difficult to see that $g_k(v)$ must be contiguous in v. As $\tilde{V}$ is dense in V our corollary is proved. □

This result gives rise to several interesting algorithmic and combinatorial questions. For example given an approximation factor $c \geq 1$, what is the minimal size of a sub-family $\mathcal{O} \subseteq O$ such that for every v, $\max_{y \in \mathcal{O}} g(v, y) \geq c \cdot g_{opt}(v)$.

We now show that non-optimal truthful VCG-based mechanisms suffer from the following disturbing abnormal behavior:

Definition 9 (reasonable mechanism) *A mechanism for combinatorial auctions is called* reasonable *if whenever there exists an item j and an agent i such that*

- *For all S, if $j \notin S$ then $v^i(S \cup \{j\}) > v^i(S)$.*
- *For every agent $l \neq i$, $v^i(S \cup \{j\}) = v^i(S)$.*

then j is allocated to agent i.

In other words, when one of the agents desires some item and the others do not, then the "right one" gets it.

Theorem 3.8 *Any non-optimal truthful VCG-based mechanism for combinatorial auctions is not reasonable.*

Proof: By theorem 3.7, let $S = (S^1, \ldots, S^n)$ be an allocation which is not in the range of the output algorithm. Define a type vector v by $v^i(X) = |X \cap S^i|$. Clearly each item j is desired by a single agent. As the mechanism's allocation is not S, there exist at least one item which is not allocated to the "right" agent. □

3.2 Truthful VCG-based mechanisms for cost minimization problems

We now show that for many natural cost minimization problems, any truthful VCG-based mechanism is either optimal or produces results which are arbitrarily far from the optimal.

We start with an example.

Multicast transmissions: A communication network is modeled by a directed graph $G = (V, E)$. Each edge e is a privately owned link. The cost t_e of sending a message along that edge is *privately* known to its owner. Given a source $s \in V$ and a set $T \subseteq V$ of terminals, the mechanism must select a subtree rooted in s that covers all the terminals. The message is then broadcasted along this tree. We assume that no agent owns a cut in the network.

Naturally, the goal of the mechanism is to select, among all possible trees, a tree R that minimizes the total cost: $\sum_{e \in R} t_e$. The natural goal of each agent however is to maximize her *own* profit: $p^i - \sum_{(e \in R \text{ owned by } i)} t_e$. It is not difficult to see that this a utilitarian mechanism design problem.

This example was introduced at [3] (under a different model). It is motivated by the need to broadcast long messages (e.g. movies) over the Internet. We now generalize this example.

Definition 10 (cost minimization allocation problem)

A cost minimization allocation problem (CMAP) *is a mechanism design problem described by:*

Type space *The type of each agent i is described by a vector $(v_1^i, \ldots, v_{m_i}^i)$. We let $m = \sum_i m_i$. (In our multicast example the v_j^is correspond to the negation of the costs t_e.)*

Allowable outputs *Each output is denoted by a bit vector $x = (x_1^1, \ldots, x_{m_1}^1, \ldots, x_1^n, \ldots, x_{m_n}^n) \in \{0,1\}^m$. We denote $(x_1^i, \ldots, x_{m_i}^i)$ by x^i. There may be additional constraints on the set O of allowable outputs. (In our example x should correspond to a tree the network's graph where x_j^i equals 1 iff the corresponding edge is in the tree.)*

such that the following conditions are satisfied:

Non-boundedness *if $v^i = (v_1^i, \ldots, v_{m^i}^i)$ describes a type for agent i and $w^i \leq v^i$ (as vectors), them w^i also describe a type.*

Independence and monotonicity *Each valuation v^i depends only on i's bits x^i. (In our example, the agent valuation of a given tree depends only on her own edges in it.) If for all j, $w_j^i \leq v_j^i$ then for every output x, $w^i(x^i) \leq v^i(x^i)$.*

Forcing condition *For every type v, an allowable output x and a real number α, define a type $v[\alpha]$ by*

$$v[\alpha]_j^i = \begin{cases} v_j^i & \text{if } x_j^i = 1 \\ \alpha & \text{otherwise} \end{cases}$$

The forcing condition is satisfied if for every allowable output $y \neq x$, $\lim_{\alpha \to -\infty} g(t(\alpha), y) = -\infty$.

Many natural problems in which the goal is to minimize the total cost under given constraints belong to this class. In particular the reader may verify that our multicast example is of this kind.

Notations: For a type v we let $g_{opt}(v)$ denote the optimal value of g. We denote $g(v, k(v))$ by $g_k(v)$.

Definition 11 (degenerate algorithm) *An output algorithm k is called* degenerate *if the ratio $r_k(v) = \frac{g_k(v) - g_{opt}(v)}{|g_{opt}(v)|+1}$ is unbounded. I.e. there exist v's such that $r_k(v)$ is arbitrarily large.*

A degenerate algorithm is arbitrarily far from optimal both additively and multiplicatively. Note that this should not be confused with the standard notion of an approximation ratio, as our definition corresponds to a single problem. In particular the number of agents is fixed.

Theorem 3.9 *If a VCG-based mechanism for a CMAP is truthful then its output algorithm is either optimal or degenerate.*

Proof: Let $m = (k, p)$ be a non-optimal truthful VCG-based mechanism for a CMAP. Like in theorem 3.2 assume that $p^i(w) = \sum_{j \neq i} w^j(k(w))$. Let v be a type (description) such that $k(v)$ is not optimal and let $y = opt(v)$ be an optimal output.

We define a type z by:

$$z^i_j = \begin{cases} v^i_j & \text{if } y^i_j = 1 \\ -\alpha & \text{otherwise} \end{cases}$$

where α is arbitrarily large.

Consider the type sequence:

$$\begin{aligned} v_0 = &\quad (v^1, \ldots, v^n) \\ v_1 = &\quad (z^1, v^2, \ldots, v^n) \\ \vdots & \\ v_n = &\quad (z^1, \ldots, z^n) \end{aligned}$$

Claim 3.10 *For all* j, $y = opt(v_j)$.

Proof: y is optimal for v_0. From the independence condition, $g(v_j, y) = g(v_0, y)$. From the monotonicity, $g(v_j, x) \leq g(v_0, x)$ for all x. □

Claim 3.11 $g(v_1, k(v_1)) < g(v_1, y)$

Proof: Otherwise $g(v_1, k(v_1)) = g(v_1, y) = g(v_0, y) > g(v_0, k(v_0))$. Therefore, when agent 1's type is v^1 and the declarations of the other agents are $v^2, \ldots, v^n$, agent 1 is better off declaring z^1. This contradicts the truthfulness of the mechanism. □

Similarly, we obtain that $g(z, k(z)) < g(z, y) = g(v_0, y)$. By the forcing condition, $g(z, k(z)) \rightarrow -\infty$ when $\alpha \rightarrow \infty$. By that the theorem is proved. □

When a CAMP is NP-Complete, an optimal algorithm is infeasible from a computational point of view. Thus, any computationally feasible truthful VCG-based mechanism for such a problem will produce unreasonable results. If we insist on solving problems under dominant strategies we are therefore in trouble. Another interesting class of mechanisms is one where the agents first declare their types, then calculate the output according to some protocol and finally calculate the payments according to the VCG formula. An immediate outcome of theorem 3.9 is that if such a mechanism has a non-optimal ex-post Nash equilibrium, then there are equilibriums which are arbitrarily far from optimal. Finally we comment that we do not know how to prove such theorems for Bayesian models.

4 Computationally feasible VCG mechanisms

To date, VCG is the only known general method for the development of strategy-proof mechanisms. Therefore the results in the previous section do not leave much hope for the development of satisfying, computationally feasible strategy-proof mechanisms.

Our goal in this section is to develop feasible mechanisms that provide solution concepts of the same spirit as strategy-proofness.

Taking into consideration the computational limitations of the agents, we define the concept of feasible dominance. It is meant to be used in the context of revelation games. We assume that each of the agents chooses her action according to some *knowledge* she has at the beginning of the game. This knowledge may represent the computational limitations of the agent or some structural limitations on her information or behavior. We say that an action is feasibly dominant for the agent if she is not *aware* of *any* circumstances where another action is better for her. We argue that if an agent has such actions available, then it is irrational for her not to choose one of them.

When VCG-based mechanisms are carefully inspected one sees that the only reason for an agent to lie about her type is to help the algorithm to improve the overall result. This leads to the intuition that if the agents cannot improve upon the underlying algorithm then they can do no better than being truthful.

Given any algorithm for the corresponding optimization problem we define the second-chance mechanism based on it. This mechanism is a modification of the VCG-based mechanism where in addition to their type declarations, the agents are allowed to submit appeal functions as well. We show that under reasonable assumptions on the agents, truth-telling is feasibly dominant. When the agents act rationally and report their true types, the mechanism's result is *at least as good* as the underlying algorithm's. Our mechanism also satisfies participation constraints.

The reader is noted that our solution concept and the reasoning of why an agent should tell the truth to the mechanism are essentially different from previous works on bounded rationality. (For a recent book on this topic see [22].)

Subsection 4.1 describes the concept of feasibly dominant actions. The second-chance mechanism is introduced in subsection 4.2. In subsection 4.3 we describe why agents should tell the truth to our mechanism. Additional implementation issues are discussed in subsection 4.4. Our method is summarized in subsection 4.5.

4.1 Feasibly dominant actions

The basic models of equilibria in game theory are justified by the (hidden) assumption that the agents are capable of computing their best response functions.

In many natural games however, the action space is huge and this function is too complex to be calculated or even to be approximated within a reasonable amount of time. In such situations this assumption seems no longer valid.

In this section we re-formulate the concept of dominant actions (strategies) under the assumption that agents have a limited capability of computing their best response. Our concept is meant to be used in the context of revelation games. We assume that each of the agents chooses her action according to some **knowledge** she has at the beginning of the game. We define an action to be feasibly dominant if, when the agent has chosen her action, she was not *aware* of *any* circumstances where another action is better for her.

Notations: We denote the action space of agent i by A^i. Given a tuple $a = (a^1, \ldots, a^n)$ of actions chosen by the agents, we denote the utility of agent i by $u^i(a)$.

In a direct revelation mechanism (definition 2) $A^i = V^i$ and $u^i(w) = v^i(k(w)) + p^i(w)$. Note that the agent's type is suppressed.

Definition 12 (strategic knowledge) Strategic knowledge *(or* knowledge *for short) of agent i is a partial function $b^i : A^{-i} \to A^i$.*

Knowledge is a function by which the agent describes (for herself!) how she would like to respond to any given situation. The semantics of $a^i = b^i(a^{-i})$ is "when the others' actions are a^{-i} the best action I can think of is a^i".

Naturally we will assume that each agent is capable of computing her own knowledge and henceforth that b^i can be computed in a reasonable amount of time. (This assumption is formulated in section 4.4.)

Definition 13 (feasible best response) *An action a^i for agent i is called* feasible best response *to a^{-i} if either a^{-i} is not in the domain of the agent's knowledge b^i or $u^i((b^i(a^{-i}), a^{-i})) \leq u^i(a)$.*

In other words, other actions may be better against a^{-i} but at least when choosing her action the agent was not aware of these.

Note that when the agent's knowledge b^i is optimal, i.e. it is a best response function, this definition corresponds to the standard one. Note also that if a^{-i} is not in the domain of b^i, then any action a^i is a feasible best response to it.

The definition of feasibly dominant actions now follows naturally.

Definition 14 (feasibly dominant action) *An action a^i for agent i is called* feasibly dominant *if it is a feasible best response against any a^{-i}. We also call such an action fda.*

A dominant action is obviously feasibly dominant.

To exemplify the concept, consider a match between two chess-players, White and Black, where each of the opponents is required to submit a computer program that plays on her behalf. Black's knowledge for example specifies what program she would like to submit if she knew White's. Of course one cannot expect such knowledge to be optimal. A program A_b is a feasible best response to A_w if Black does not know how to play better against A_w, more precisely, if when submitting A_b Black was not aware of any better way to play against A_w. Note that since this is a revelation game, we do not refer to knowledge that may be acquired during the game. A_b is feasibly dominant for Black if she cannot think of any change in the program that would help her against any possible opponent! We argue that if an agent has feasibly dominant actions available, then it is irrational not to choose one of them.

Our goal is to define a mechanism where truth-telling is feasibly dominant.

4.2 The second-chance mechanism

Although VCG-based mechanisms may not be truthful, it is not difficult to see that the only reason for an agent to lie about her type is to help the algorithm to improve the overall result. An agent who believes that the declarations of the others are such that she may better lie to the algorithm, might be wrong and henceforth cause damage to the overall result.

In order to prevent this we introduce the usage of appeal functions. The semantics of an appeal l is: "when the agents' type is $v = (v^1, \ldots, v^n)$ I believe that the output algorithm k produces a better result if it is given $l(v)$ instead of the actual input v". Given the agents' declarations w, the mechanism then computes $k(w)$ and the results of all the appeals $k(l^1(w)), \ldots, k(l^n(w))$ and selects the best output among these results.

The idea is that instead of declaring a falsified type, it is better for the agent to report it *truthfully* and to ask the mechanism to *check* whether the false declaration would have lead to better results. When the agents are truth-telling, the output chosen by the mechanism is at least as good as $k(v)$.

Definition 15 (appeal function) *Let $V = \prod_i V^i$ denote the type space of the agents. An* appeal *is a function $l : V \to V$.*

We now define the second-chance mechanism.

Definition 16 (second-chance mechanism) *Given an output algorithm k, the* second chance mechanism based on k *is the following game:*

1. *Each agent sends a type declaration w^i and an appeal function l^i to the mechanism.*

2. *Let $w = (w^1, \ldots, w^n)$. The mechanism computes $k(w), k(l^1(w)), \ldots, k(l^n(w))$ and chooses among these outputs the one that maximizes the total welfare (according to w). I.e. the mechanism tries all the appeals and chooses the one that yields the best result.*

3. *Let $\hat{o}$ denote the chosen output. The mechanism then calculates the payments according to the VCG formula: $p^i = \sum_{j \neq i} w^j(\hat{o}) + h^i(w^{-i})$ (where h^i is any real function).*

The actual implementation of the appeal functions is discussed in subsection 4.4. Note that $h^i(w^{-i})$ is independent of the agents' appeals. Until section 4.4 we can simply assume that it is always zero.

An action in a second chance mechanism is a pair (w^i, l^i) where w^i is a type declaration and l^i is an appeal function.

Definition 17 (truthful action) *An action $a^i = (w^i, l^i)$ is called* truthful *if $w^i = v^i$, i.e. the agent reveals her true type to the mechanism. A mechanism is called* feasibly truthful *if truth-telling is feasibly dominant for all the agents.*

The following observation is a key property of the mechanism.

Proposition 4.1 *Consider a second-chance mechanism with an output algorithm k. For every type $v = (v^1, \ldots, v^n)$, if all agents are truth-telling, the output chosen by the mechanism is at least as good as $k(v)$.*

□

In other words, when the agents are truth-telling, the mechanism is at least as good as the underlying algorithm. In particular, when the algorithm is guaranteed to be within a certain constant from the optimal, then so is the mechanism.

We will show that under reasonable assumptions on the agents, the mechanism is indeed feasibly truthful. We also need to guarantee that all the computations can be carried out in a reasonable amount of time.

An example

To exemplify the strength of the mechanism consider Alice who participates in a combinatorial auction for houses. She is interested in a particular pair of adjacent cottages and is willing to pay up to $\$200,000$ for them. She is also willing to pay up to $\$60,000$ for each one separately.

Alice's type is therefore $v^i = \{200, 60, 60\}$ (in thousands, regarding only these particular two houses). After experimenting with the mechanism Alice notices that in many situations a declaration of $w^i = \{200, 0, 0\}$ causes the algorithm to produce a better result.

In a VCG-based mechanism Alice may choose to declare w^i instead of v^i. Sometimes, depending on the type declarations of the others, it would indeed improve the overall result and would therefore increase her own profit as well. On other occasions it would reduce the quality of the overall result and therefore decrease Alice's profit.

In the second chance mechanism Alice can report her true valuation v^i and define $l^i(v^i, w^{-i})$ as (w^i, w^{-i}). This way she can enjoy the best of both worlds: In cases where declaring w^i would improve the overall result, the mechanism will choose $k(l^i(v))$. In cases where lying to the algorithm would reduce the quality of the overall result, the mechanism will prefer $k(v)$, and thus prevent the damage.

4.3 Existence of computationally efficient feasibly dominant truth-telling actions

In this subsection we will show that under reasonable assumptions, computationally efficient feasibly dominant truth-telling actions do exist for the agents. Note that our assumptions are different from what is customary in game theory.

We first comment that if no time limit is enforced on the appeal functions then the agents have dominant actions available: Consider an agent i participating in a second chance mechanism. Let k denote the output algorithm. Given the type declarations w^{-i} of the other agents, there exists a type vector $\hat{w} \stackrel{df}{=} l^i(w^{-i})$ that maximizes $g((v^i, w^{-i}), k(\hat{w}))$. It is not difficult to see that (v^i, l^i) is dominant and henceforth feasibly dominant action for the agent. However, if the original problem is too hard to compute then so is l^i.

When the agents are limited to submit only computationally limited appeal functions, the existence of feasibly dominant truth-telling actions is less obvious. In this section we show that such actions are available under reasonable assumptions on the way that the agents' knowledge is *obtained*. We argue that in practice this knowledge will not "fall out of the blue", but there will be a few reasonable ways for the agents to obtain it.

We focus on two types of knowledge that look to us the most natural ones:

The first is obtained by an agent who simply explores the output algorithm. Such an agent ignores potential appeals for the other agents. Therefore knowledge of this kind is actually knowledge about the algorithm. The justification behind the assumption that agents will have this kind of knowledge is that the space of possible appeals is so huge and complex that it is *beyond the agents' capability* to reason about it.

The second type of knowledge is obtained by an agent who explores, in addition to the output algorithm, a small family of potential appeals for the other agents.

In both cases we show the existence of computationally efficient feasibly dominant truth-telling actions for the agent.

Clearly, when an agent lies about her type, there exist cases where she will consequently loose. Therefore the agents are further motivated to be truthful.

We begin with a formal definition for computationally limited algorithms and mechanisms.

Definition 18 (algorithm of degree d) *We say that an output algorithm is* of degree d *if its running time is bounded by some polynomial of degree d in the number of agents.*

The definition for appeal functions, actions, knowledge and mechanisms are similar.

Proposition 4.2 *If the output algorithm of a second-chance mechanism is of degree d then the mechanism is of degree $d+1$.*

□

In subsection 4.4 we discuss ways to force the agents to submit only appeal functions of degree d.

We now refer to the first type of knowledge obtained by an agent who simply explores the output algorithm:

Definition 19 (appeal independent knowledge) *Knowledge b^i is called* appeal independent *if it is of the form $b^i : V^{-i} \to A^i$. Such knowledge is called* declaration based *if it is of the form $b^i : V^{-i} \to V^i$.*

The semantics of declaration based knowledge is: "when the others declare w^{-i}, I would like to declare $b^i(w^{-i})$". Alice in our example has such kind of knowledge. The semantics of $(w^i, l^i) = b^i(w^{-i})$ where b^i is appeal independent is: "when the others declare w^{-i}, I would like to declare w^i and to submit the appeal l^i".

Theorem 4.3 *If b^i is a declaration based knowledge for agent i then (v^i, b^i) is feasibly dominant for the agent.*

Proof: Assume by contradiction that it is not feasibly dominant. Therefore there exists a vector of actions a^{-i} for the others such that $(w^i, \phi) \overset{df}{=} b^i(a^{-i})$ is strictly better for the agent than playing (v^i, b^i). Note that i's appeal is always empty. Since b^i is declaration based, we can assume that all the appeals of the other agents are also empty. Denote a^{-i} by (w^{-i}, ϕ). Consider the case where the agent's action is (v^i, b^i). If the mechanism "rejects the appeal" and chooses the type (v^i, w^{-i}) then $g((v^i, w^{-i}), k(v^i, w^{-i})) \geq g((v^i, w^{-i}), k(w^i, w^{-i}))$. By declaring w^i the agent forces the mechanism to choose $k(w^i, w^{-i})$ which can only reduce the utility of the agent. If the mechanism "accepts the appeal" and chooses the type (w^i, w^{-i}), then it does the same when the agent declares w^i. $\square$

Theorem 4.4 *If the output algorithm is of degree d and the agent has appeal independent knowledge of degree d then there exists a truthful feasibly dominant action of degree d for this agent.*

Proof: Let b^i be a knowledge of degree d for agent i. Define an appeal l^i as follows: Given w^{-i}, let $(w^i, \psi^i) = b^i(w^{-i})$. The appeal computes $k(w^i, w^{-i})$ and $k(\psi((w^i, w^{-i})))$ and takes the better result according to (v^i, w^{-i}). Obviously l^i is feasibly dominant and of degree d $\square$

We now consider a more general scenario, where the agent explores, in addition to the output algorithm, a small family of potential appeals for the other agents.

Definition 20 (d-obtainable knowledge) *Knowledge b^i is called* d-obtainable *if the following holds:*

1. *b^i is of degree d.*
2. *Every appeal function that appears, in the domain or in the range of b^i, is of degree d.*
3. *There are at most n^d appeal functions that appear in the domain or in the range of b^i. Moreover there exists a representative family L^i of no more than n^d $(n-1)$-tuples of appeals such that for every tuple ϕ^{-i} that appears in the domain of b^i there exists a $\psi^{-i} \in L^i$ such that for all w^{-i}, $b^i((w^{-i}, \phi^{-i})) = b^i((w^{-i}, \psi^{-i}))$.*

The second condition is justified by the assumption that an agent can not reason about functions that she cannot compute. The representative family in the third condition stands for all the cases that the agent has checked. The underlying assumption is that an agent cannot reason about functions and tuples that she has not checked at least once. Naturally we assume that an agent can check only a reasonable number of cases.

Theorem 4.5 *If the output algorithm is of degree d and an agent has a d-obtainable knowledge, then she has a truthful feasibly dominant action of degree $3 \cdot d$.*

Proof: Let b^i be a d-obtainable knowledge for agent i. Given w^{-i} we shall define an appeal l^i as follows: Let L be the family of all appeals that appear in the domain or in the range of b^i. Let L^i be a representative family and let $W = \{w^i | \exists \psi^{-i} \in L^i, \phi^i s.t. (w^i, \phi^i) = b^i((w^{-i}, \psi^{-i}))\}$. Obviously $|W|, |L|$ are bounded by a polynomial of degree d. l^i computes for every pair $(w^i \in W, l \in L)$, $k(l((w^i, w^{-i})))$ and takes the best output according to (v^i, w^{-i}). It is not difficult to see that (v^i, l^i) is feasibly dominant and that l^i is of degree $3 \cdot d$. $\square$

When the agents' knowledge is obtained in one of the ways mentioned above then if the time limit enforced on the appeals is not too small, the agents are strongly motivated to report their true types to the mechanism. We argue that this will be the case for many applications of interest.

Truthful feasibly dominant actions may not exist in situations where an agent has significant knowledge about properties of the potential appeals of the others. Such situations may arise in repeated settings (e.g an auction that runs every week). This may be solved by merging each appeal that improves the result of the output algorithm into it and letting the agents update their appeal functions.

4.4 Other implementation issues

4.4.1 Implementation of the appeal functions

We see two natural ways to implement the concept of appeal functions. The first is simply to let the agents compute their appeals by themselves and to send the results to the mechanism in a second round. In this method however, the type of each agent is revealed to all the others. This is undesirable for many applications. Another natural way is to supply the agents with a language for the description of their appeals and to perform the computation on the mechanism's machine.

In both methods it is not difficult to impose a time limit on the agents or to ask them to pay for additional computational time. However an arbitrary time limit may prevent the existence of feasible dominant actions.

An interesting direction is to impose on the description of the appeals a structure that reflects the agents' knowledge of the output algorithm. One possible structure is a decision tree where the agents are required to supply, for each leaf α, a type vector t_α such that the algorithm's result is strictly improved when it is given $l(t_\alpha)$ instead of the actual input t_α. In this method the computational time of the mechanism is naturally limited by the computational power of the agents themselves.

4.4.2 Participation constraints

One desirable property of mechanisms is that the utility of a truthful agent is always non-negative. This is often called participation constraints (see e.g. [12]). In this section we construct a second-chance mechanism that satisfies this property.

We shall assume that for each agent i, there exists a type $\underline{v}^i$ such that for every $v = (v^1, \ldots, v^n)$, $g_{opt}((\underline{v}^i, v^{-i})) \leq g_{opt}(v)$. In a combinatorial auction for example, this type is defined by $\underline{v}^i(s) = 0$ for every combination s of items.

The Clarke mechanism ([1]) is a VCG mechanism with payments $p^i(w) = \sum_{j \neq i} w^j(opt(w)) - g_{opt}(\underline{v}^i, (w^{-i}))$. It is not difficult to check that the mechanism satisfies participation constraints. When the optimal algorithm in the Clarke mechanism is replaced by a sub-optimal one, the result of the algorithm may be improved when w^i is replaced by $\underline{v}^i$. Therefore participation constraints may no longer be satisfied. However this is not difficult to fix.

Definition 21 *We say that an output algorithm k* satisfies participation constraints *if for every type $v = (v^1, \ldots, v^n)$ and agent i, $g_k(v) \geq g_k((\underline{v}^i, v^{-i}))$.*

Proposition 4.6 *If the output algorithm k is of degree d, then there exists an algorithm $\tilde{k}$ of degree $d+2$ that satisfies participation constraints such that for every type v, $g_{\tilde{k}}(v) \geq g_k(v)$.*

Proof: Given v we define $\tilde{k}(v)$ as follows. At the beginning we let $k_0 = k$. In the $(j+1)$th phase, if for every agent i, $g_{k_j}(v) \geq g_{k_j}(\underline{v}^i, v^{-i})$ then fix $\tilde{k}(v) = k_j(v)$ and stop; Otherwise fix the type of one of the agents i who does not satisfy the above condition to be $\underline{v}^i$ and continue. It is not difficult to verify that $\tilde{k}$ satisfies the required conditions. $\square$

Given an algorithm $\tilde{k}$ that satisfies participation constraints we shall define the payment function of the second chance mechanism based on it as $p^i = \sum_{j \neq i} w^j(o) - g_{\tilde{k}}(\underline{v}^i, (w^{-i}))$ where w denotes the agents' declarations and o the chosen output. Since $g(w, o) \geq g_{\tilde{k}}(w)$, participation constraints are satisfied.

4.5 Summary of the method

We summarize the necessary steps the designer of a mechanism needs to take when using our method. First the designer needs to develop an algorithm k for the corresponding optimization problem. This could be done without any constraints on the global behavior of the algorithm. The designer can automatically transform k into an algorithm $\tilde{k}$ that satisfies participation constraints. The next step is to find a reasonable time limit on the appeals or better to develop a language (API) for the appeals that reflects the agents' knowledge of the algorithm. After this is done the designer can give the mechanism to the agents to experiment with and to learn $\tilde{k}$. Finally, after the participants are ready, the second-chance mechanism could be executed. In repeated settings the designer may want to merge (automatically) successful appeals into $\tilde{k}$ and let the agents update their appeal functions afterwards. Note that all the steps required for turning the algorithm k into a mechanism are either automatic or common for all mechanism design problems. Using our method, the development of a mechanism for a mechanism design problem is therefore reduced to the development of a good algorithm for the corresponding optimization problem.

Acknowledgments: We thank Abraham Newman and Motty Perry for helpful discussions at various stages of this work. We thank Ron Lavi, Ahuva Mu'alem, Elan Pavlov 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] Peter Cramton. The fcc spectrum auction: an early assessment. *Journal of Economics and Management Strategy*, 6:431–495, 1997.

[3] 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.

[4] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *IJCAI-99*, 1999.

[5] Michael R. Garey and David S. Johnson. *Computers and Intractability : A Guide to the Theory of Np-Completeness*. W.H. Freeman and Company, 1979.

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

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

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

[9] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *STACS 99, the 16th Annual Symposium on Theoretical Aspects of Computer Science*, March 1999.

[10] 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.

[11] 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.

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

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

[14] Paul Milgrom. Putting auction theory to work: the simultaneous ascending auction. Technical Report 98-002, Dept. of Economics, Stanford University, 1998.

[15] Noam Nisan. Bidding and allocation in combinatorial auctions. To appear.

[16] 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.

[17] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behaviour*, 2000. To appear.

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

[19] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley Publishing, 1994.

[20] 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.

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

[22] Ariel Rubinstein. *Modeling Bounded Rationality.* MIT Press, 1998.

[23] Tuomas W. Sandholm. Approaches to winner determination in combinatorial auctions. *Decision Support Systems*, to appear.

[24] Moshe Tennenholtz. Some tractable combinatorial auctions. In *the national conference on artificial intelligence (AAAI-2000)*, 2000.

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

[26] 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.

[27] 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.