Automated Mechanism Design for a Self-Interested Designer[*]

Vincent Conitzer and **Tuomas Sandholm**
{conitzer, sandholm}@cs.cmu.edu
Computer Science Department
Carnegie Mellon University
Pittsburgh PA 15213

Abstract

Often, an outcome must be chosen on the basis of the preferences reported by a group of agents. The key difficulty is that the agents may report their preferences insincerely to make the chosen outcome more favorable to themselves. *Mechanism design* is the art of designing the rules of the game so that the agents are motivated to report their preferences truthfully, and a desirable outcome is chosen. In a recently emerging approach proposed in UAI-02—called *automated mechanism design*—a mechanism is computed for the preference aggregation setting at hand. This has several advantages, but the downside is that the mechanism design optimization problem needs to be solved anew each time. Unlike the earlier work on automated mechanism design that studied a benevolent designer, in this paper we study automated mechanism design problems *where the designer is self-interested*. In this case, the center cares only about which outcome is chosen and what payments are made to it. The reason that the agents' preferences are relevant is that the center is constrained to making each agent at least as well off as the agent would have been had it not participated in the mechanism. In this setting, we show that designing optimal deterministic mechanisms is $\mathcal{NP}$-complete in two important special cases: when the center is interested only in the payments made to it, and when payments are not possible and the center is interested only in the outcome chosen. We then show how allowing for randomization in the mechanism makes problems in this setting computationally easy.

1 Introduction

In multiagent settings, often an *outcome* must be chosen on the basis of the preferences reported by a group of agents. Such outcomes could be potential presidents, joint plans, allocations of goods or resources, etc. The preference aggregator generally does not know the agents' preferences *a priori*. Rather, the agents report their preferences to the coordinator. Unfortunately, an agent may have an incentive to misreport its preferences in order to mislead the mechanism into selecting an outcome that is more desirable to the agent than the outcome that would be selected if the agent revealed its preferences truthfully.

Manipulability is a pervasive problem across preference aggregation mechanisms. A seminal negative result, the *Gibbard-Satterthwaite theorem*, shows that under *any* nondictatorial preference aggregation scheme, if there are at least 3 possible outcomes, there are preferences under which an agent is better off reporting untruthfully [6, 16]. (A preference aggregation scheme is called dictatorial if one of the agents dictates the outcome no matter what preferences the other agents report.)

What the aggregator would like to do is design a preference aggregation mechanism so that 1) the self-interested agents are motivated to report their preferences truthfully, and 2) the mechanism chooses an outcome that is desirable from the perspective of some objective. This is the classic setting of *mechanism design* in game theory. In this paper, we study the case where the designer is *self-interested*, that is, the designer does not directly care about how the outcome relates to the agents' preferences, but is rather concerned with its own agenda for which outcome should be chosen, and with maximizing payments to itself. This is the mechanism design setting most relevant to electronic commerce.

In the case where the mechanism designer is interested in maximizing some notion of social welfare, the importance of collecting the agents' preferences is clear. It is perhaps less obvious why they should be collected when the designer is self-interested and hence its objective is not directly related to the agents' preferences. The reason for this is that often the agents' preferences impose limits on how the designer chooses the outcome and payments. The most common such constraint is that of *individual rationality (IR)*, which means that the mechanism cannot make any agent worse off than the agent would have been had it not participated in the mechanism. For instance, in the setting of optimal auction

[*] This material is based upon work supported by NSF under CAREER Award IRI-9703122, Grant IIS-9800994, ITR IIS-0081246, and ITR IIS-0121678.

design, the designer (auctioneer) is only concerned with how much revenue is collected, and not per se with how well the allocation of the good (or goods) corresponds to the agents' preferences. Nevertheless, the designer cannot force an agent to pay more than its valuation for the bundle of goods allocated to it. Therefore, even a self-interested designer will choose an outcome that makes the agents reasonably well off. On the other hand, the designer will not necessarily choose a social welfare maximizing outcome. For example, if the designer always chooses an outcome that maximizes social welfare with respect to the reported preferences, and forces each agent to pay the difference between the utility it has now and the utility it would have had if it had not participated in the mechanism, it is easy to see that agents may have an incentive to misreport their preferences—and this may actually lead to less revenue being collected. Indeed, one of the counterintuitive results of optimal auction design theory is that sometimes the good is allocated to nobody even when the auctioneer has a reservation price of 0.

Classical mechanism design provides some general mechanisms, which, under certain assumptions, satisfy some notion of nonmanipulability and maximize some objective. The upside of these mechanisms is that they do not rely on (even probabilistic) information about the agents' preferences (e.g., the Vickrey-Clarke-Groves (VCG) mechanism [17, 2, 7]), or they can be easily applied to any probability distribution over the preferences (e.g., the dAGVA mechanism [5, 1], the Myerson auction [12], and the Maskin-Riley multi-unit auction [11]). However, the general mechanisms also have significant downsides:

- The most famous and most broadly applicable general mechanisms, VCG and dAGVA, only maximize social welfare. If the designer is self-interested, as is the case in many electronic commerce settings, these mechanisms do not maximize the designer's objective.
- The general mechanisms that do focus on a self-interested designer are only applicable in very restricted settings—such as Myerson's expected revenue maximizing auction for selling a single item, and Maskin and Riley's expected revenue maximizing auction for selling multiple identical units of an item.
- Even in the restricted settings in which these mechanisms apply, the mechanisms only allow for payment maximization. In practice, the designer may also be interested in the outcome per se. For example, an auctioneer may care which bidder receives the item.
- It is often assumed that side payments can be used to tailor the agents' incentives, but this is not always practical. For example, in barter-based electronic marketplaces—such as Recipco, firstbarter.com, BarterOne, and Intagio—side payments are not allowed. Furthermore, among software agents, it might be more desirable to construct mechanisms that do not rely on the ability to make payments, because many software agents do not have the infrastructure to make payments.

In contrast, we follow the recently emerging approach where the *mechanism is designed automatically for the specific problem at hand.* This approach addresses all of the downsides listed above. We formulate the mechanism design problem as an optimization problem. The input is characterized by the number of agents, the agents' possible types (preferences), and the aggregator's prior distributions over the agents' types. The output is a nonmanipulable mechanism that is optimal with respect to some objective. This approach is called *automated mechanism design.*

The automated mechanism design approach has four advantages over the classical approach of designing general mechanisms. First, it can be used even in settings that do not satisfy the assumptions of the classical mechanisms (such as availability of side payments or that the objective is social welfare). Second, it may allow one to circumvent impossibility results (such as the Gibbard-Satterthwaite theorem) which state that there is no mechanism that is desirable across all preferences. When the mechanism is designed for the setting at hand, it does not matter that it would not work more generally. Third, it may yield better mechanisms (in terms of stronger nonmanipulability guarantees and/or better outcomes) than classical mechanisms because the mechanism capitalizes on the particulars of the setting (the probabilistic information that the designer has about the agents' types). Given the vast amount of information that parties have about each other today, this approach is likely to lead to tremendous savings over classical mechanisms, which largely ignore that information. For example, imagine a company automatically creating its procurement mechanism based on statistical knowledge about its suppliers, rather than using a classical descending procurement auction. Fourth, the burden of design is shifted from humans to a machine.

However, automated mechanism design requires the mechanism design optimization problem to be solved anew for each setting. Hence its computational complexity becomes a key issue. Previous research has studied this question for benevolent designers—that wish to maximize, for example, social welfare [3]. In this paper we study the computational complexity of automated mechanism design in the case of a self-interested designer. This is an important setting for automated mechanism design due to the shortage of general mechanisms in this area, and the fact that in most e-commerce settings the designer is self-interested.

The rest of this paper is organized as follows. In Section 2, we justify our focus on nonmanipulable mechanisms. In Section 3, we define the problem. In Section 4, we show that designing an optimal deterministic mechanism is $\mathcal{NP}$-complete even when the designer only cares about the payments made to it. In Section 5, we show that designing an optimal deterministic mechanism is also $\mathcal{NP}$-complete when payments are not possible and the designer is only interested in the outcome chosen. Finally, in Section 6, we show that an optimal randomized mechanism can be designed in polynomial time even in the general case.

2 Justifying the focus on nonmanipulable mechanisms

Before we define the computational problem of automated mechanism design, we should justify our focus on nonmanipulable mechanisms. After all, it is not immediately obvious that there are no manipulable mechanisms that, even when agents report their types strategically and hence sometimes untruthfully, still reach better outcomes (according to whatever objective we use) than any nonmanipulable mechanism. This does, however, turn out to be the case: given any mechanism, we can construct a nonmanipulable mechanism whose performance is identical, as follows. We build an interface layer between the agents and the original mechanism. The agents report their preferences (or *types*) to the interface layer; subsequently, the interface layer inputs into the original mechanism the types *that the agents would have strategically reported* to the original mechanism, if their types were as declared to the interface layer. The resulting outcome is the outcome of the new mechanism. Since the interface layer acts "strategically on each agent's behalf", there is never an incentive to report falsely to the interface layer; and hence, the types reported by the interface layer are the strategic types that would have been reported without the interface layer, so the results are exactly as they would have been with the original mechanism. This argument is known in the mechanism design literature as the *revelation principle* [10]. (There are computational difficulties with applying the revelation principle in large combinatorial outcome and type spaces [4, 15]. However, because here we focus on flatly represented outcome and type spaces, this is not a concern here.) Given this, we can focus on truthful mechanisms in the rest of the paper.

3 Definitions

We now formalize the automated mechanism design setting.

Definition 1 *In an* automated mechanism design setting*, we are given*

A finite set of outcomes O;

A finite set of N agents;

For each agent i,

- *a finite set of types Θ_i,*
- *a probability distribution γ_i over Θ_i (in the case of correlated types, there is a single joint distribution γ over $\Theta_1 \times \ldots \times \Theta_N$),*
- *a utility function $u_i : \Theta_i \times O \to \mathbb{R}$;* [1]

An objective function whose expectation the designer wishes to maximize.

There are many possible objective functions the designer might have, for example, social welfare (where the designer seeks to maximize the sum of the agents' utilities), or the minimum utility of any agent (where the designer seeks to maximize the worst utility had by any agent). In both of these cases, the designer is *benevolent*, because the designer, in some sense, is pursuing the agents' collective happiness. However, in this paper, we focus on the case of a *self-interested* designer. A self-interested designer cares only about the outcome chosen (that is, the designer does not care how the outcome relates to the agents' preferences, but rather has a fixed preference over the outcomes), and about the net payments made by the agents, which flow to the designer.

Definition 2 *A* self-interested designer *has an objective function given by $g(o)+\sum_{i=1}^{N} \pi_i$, where $g : O \to \mathbb{R}$ indicates the designer's own preference over the outcomes, and π_i is the payment made by agent i. In the case where $g = 0$ everywhere, the designer is said to be* payment maximizing. *In the case where payments are not possible, g constitutes the objective function by itself.*

We now define the kinds of mechanisms under study.

Definition 3 *A* deterministic mechanism without payments *consists of an outcome selection function $o : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N \to O$. A* randomized mechanism without payments *consists of a distribution selection function $p : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N \to \mathcal{P}(\mathcal{O})$, where $\mathcal{P}(\mathcal{O})$ is the set of probability distributions over O. A* deterministic mechanism with payments *consists of an outcome selection function $o : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N \to O$ and for each agent i, a payment selection function $\pi_i : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N \to \mathbb{R}$, where $\pi_i(\theta_1, \ldots, \theta_N)$ gives the payment made by agent i when the reported types are $\theta_1, \ldots, \theta_N$. A* randomized mechanism with payments *consists of a distribution selection function $p : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N \to \mathcal{P}(\mathcal{O})$, and for each agent i, a payment selection function $\pi_i : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N \to \mathbb{R}$.*[2]

There are two types of constraint on the designer in building the mechanism.

3.1 Individual rationality (IR) constraints

The first type of constraint is the following. The utility of each agent has to be at least as great as the

[1]Though this follows standard game theory notation [10], the fact that the agent has both a utility function and a type is perhaps confusing. The types encode the various possible preferences that the agent may turn out to have, and the agent's type is not known to the aggregator. The utility function is common knowledge, but because the agent's type is a parameter in the agent's utility function, the aggregator cannot know what the agent's utility is without knowing the agent's type.

[2]We do not randomize over payments because as long as the agents and the designer are risk neutral with respect to payments, that is, their utility is linear in payments, there is no reason to randomize over payments.

agent's fallback utility, that is, the utility that the agent would receive if it did not participate in the mechanism. Otherwise that agent would not participate in the mechanism—and no agent's participation can ever hurt the mechanism designer's objective because at worst, the mechanism can ignore an agent by pretending the agent is not there. (Furthermore, if no such constraint applied, the designer could simply make the agents pay an infinite amount.) This type of constraint is called an *IR (individual rationality)* constraint. There are three different possible IR constraints: *ex ante*, *ex interim*, and *ex post*, depending on what the agent knows about its own type and the others' types when deciding whether to participate in the mechanism. *Ex ante* IR means that the agent would participate if it knew nothing at all (not even its own type). We will not study this concept in this paper. *Ex interim* IR means that the agent would always participate if it knew only its own type, but not those of the others. *Ex post* IR means that the agent would always participate even if it knew everybody's type. We will define the latter two notions of IR formally. First, we need to formalize the concept of the fallback outcome. We assume that each agent's fallback utility is zero for each one of its types. This is without loss of generality because we can add a constant term to an agent's utility function (for a given type), without affecting the decision-making behavior of that expected utility maximizing agent [10].

Definition 4 *In any automated mechanism design setting with an IR constraint, there is a* fallback *outcome $o_0 \in O$ where, for any agent i and any type $\theta_i \in \Theta_i$, we have $u_i(\theta_i, o_0) = 0$. (Additionally, in the case of a self-interested designer, $g(o_0) = 0$.)*

We can now to define the notions of individual rationality.

Definition 5 Individual rationality (IR) *is defined by:*

- *A deterministic mechanism is* ex interim IR *if for any agent i, and any type $\theta_i \in \Theta_i$, we have*
  $E_{(\theta_1,..,\theta_{i-1},\theta_{i+1},..,\theta_N)|\theta_i}[u_i(\theta_i, o(\theta_1,..,\theta_N)) - \pi_i(\theta_1,..,\theta_N)] \geq 0.$

  A randomized mechanism is ex interim IR *if for any agent i, and any type $\theta_i \in \Theta_i$, we have*
  $E_{(\theta_1,..,\theta_{i-1},\theta_{i+1},..,\theta_N)|\theta_i} E_{o|\theta_1,..,\theta_n}[u_i(\theta_i, o) - \pi_i(\theta_1,..,\theta_N)] \geq 0.$

- *A deterministic mechanism is* ex post IR *if for any agent i, and any type vector $(\theta_1, \ldots, \theta_N) \in \Theta_1 \times \ldots \times \Theta_N$, we have $u_i(\theta_i, o(\theta_1, \ldots, \theta_N)) - \pi_i(\theta_1, \ldots, \theta_N) \geq 0$.*

  A randomized mechanism is ex post IR *if for any agent i, and any type vector $(\theta_1, \ldots, \theta_N) \in \Theta_1 \times \ldots \times \Theta_N$, we have $E_{o|\theta_1,..,\theta_n}[u_i(\theta_i, o) - \pi_i(\theta_1,..,\theta_N)] \geq 0$.*

The terms involving payments can be left out in the case where payments are not possible.

3.2 Incentive compatibility (IC) constraints

The second type of constraint says that the agents should never have an incentive to misreport their type (as justified above by the revelation principle). For this type of constraint, the two most common variants (or *solution concepts*) are *implementation in dominant strategies*, and *implementation in Bayes-Nash equilibrium*.

Definition 6 *Given an automated mechanism design setting, a mechanism is said to* implement its outcome and payment functions in dominant strategies *if truthtelling is always optimal even when the types reported by the other agents are already known. Formally, for any agent i, any type vector $(\theta_1, \ldots, \theta_i, \ldots, \theta_N) \in \Theta_1 \times \ldots \times \Theta_i \times \ldots \times \Theta_N$, and any alternative type report $\hat{\theta}_i \in \Theta_i$, in the case of deterministic mechanisms we have*
$u_i(\theta_i, o(\theta_1, \ldots, \theta_i, \ldots, \theta_N)) - \pi_i(\theta_1, \ldots, \theta_i, \ldots, \theta_N) \geq u_i(\theta_i, o(\theta_1, \ldots, \hat{\theta}_i, \ldots, \theta_N)) - \pi_i(\theta_1, \ldots, \hat{\theta}_i, \ldots, \theta_N)$.

In the case of randomized mechanisms we have
$E_{o|\theta_1,..,\theta_i,..,\theta_n}[u_i(\theta_i, o) - \pi_i(\theta_1, \ldots, \theta_i, \ldots, \theta_N)] \geq E_{o|\theta_1,..,\hat{\theta}_i,..,\theta_n}[u_i(\theta_i, o) - \pi_i(\theta_1, \ldots, \hat{\theta}_i, \ldots, \theta_N)]$.

The terms involving payments can be left out in the case where payments are not possible.

Thus, in dominant strategies implementation, truthtelling is optimal regardless of what the other agents report. If it is optimal only *given* that the other agents are truthful, and given that one does not know the other agents' types, we have implementation in *Bayes-Nash equilibrium*.

Definition 7 *Given an automated mechanism design setting, a mechanism is said to* implement its outcome and payment functions in Bayes-Nash equilibrium *if truthtelling is always optimal to an agent when that agent does not yet know anything about the other agents' types, and the other agents are telling the truth. Formally, for any agent i, any type $\theta_i \in \Theta_i$, and any alternative type report $\hat{\theta}_i \in \Theta_i$, in the case of deterministic mechanisms we have*
$E_{(\theta_1,..,\theta_{i-1},\theta_{i+1},..,\theta_N)|\theta_i}[u_i(\theta_i, o(\theta_1, \ldots, \theta_i, \ldots, \theta_N)) - \pi_i(\theta_1, \ldots, \theta_i, \ldots, \theta_N)] \geq$
$E_{(\theta_1,..,\theta_{i-1},\theta_{i+1},..,\theta_N)|\theta_i}[u_i(\theta_i, o(\theta_1, \ldots, \hat{\theta}_i, \ldots, \theta_N)) - \pi_i(\theta_1, \ldots, \hat{\theta}_i, \ldots, \theta_N)]$.

In the case of randomized mechanisms we have
$E_{(\theta_1,..,\theta_{i-1},\theta_{i+1},..,\theta_N)|\theta_i} E_{o|\theta_1,..,\theta_i,..,\theta_n}[u_i(\theta_i, o) - \pi_i(\theta_1, \ldots, \theta_i, \ldots, \theta_N)] \geq$
$E_{(\theta_1,..,\theta_{i-1},\theta_{i+1},..,\theta_N)|\theta_i} E_{o|\theta_1,..,\hat{\theta}_i,..,\theta_n}[u_i(\theta_i, o) - \pi_i(\theta_1, \ldots, \hat{\theta}_i, \ldots, \theta_N)]$.

The terms involving payments can be left out in the case where payments are not possible.

3.3 Automated mechanism design

We can now define the computational problem we study.

Definition 8 *(AUTOMATED-MECHANISM-DESIGN*

*(AMD)) We are given an automated mechanism design setting, an IR notion (*ex interim*, *ex post*, or none), and a solution concept (dominant strategies or Bayes-Nash). Additionally, we are told whether payments and randomization are possible. Finally, we are given a target value G. We are asked whether there exists a mechanism of the specified type that satisfies both the IR notion and the solution concept, and gives an expected value of at least G for the objective.*

An interesting special case is the setting where there is only one agent. In this case, the reporting agent always knows everything there is to know about the other agents' types—because there are no other agents. Since *ex post* and *ex interim* IR only differ on what an agent is assumed to know about other agents' types, the two IR concepts coincide here. Also, because implementation in dominant strategies and implementation in Bayes-Nash equilibrium only differ on what an agent is assumed to know about other agents' types, the two solution concepts coincide here. This observation will prove to be a useful tool in proving hardness results: if we prove computational hardness in the single-agent setting, this immediately implies hardness for both IR concepts, for both solution concepts, for any number of agents.

4 Payment-maximizing deterministic AMD is hard

In this section we demonstrate that it is $\mathcal{NP}$-complete to design a deterministic mechanism that maximizes the expected sum of the payments collected from the agents. We show that this problem is hard even in the single-agent setting, thereby immediately showing it hard for both IR concepts, for both solution concepts. To demonstrate $\mathcal{NP}$-hardness, we reduce from the MINSAT problem.

Definition 9 (MINSAT) *We are given a formula ϕ in conjunctive normal form, represented by a set of Boolean variables V and a set of clauses C, and an integer K ($K < |C|$). We are asked whether there exists an assignment to the variables in V such that at most K clauses in ϕ are satisfied.*

MINSAT was recently shown to be $\mathcal{NP}$-complete [8]. We can now present our result.

Theorem 1 *Payment-maximizing deterministic AMD is $\mathcal{NP}$-complete, even for a single agent, even with a uniform distribution over types.*

Proof: It is easy to show that the problem is in $\mathcal{NP}$. To show $\mathcal{NP}$-hardness, we reduce an arbitrary MINSAT instance to the following single-agent payment-maximizing deterministic AMD instance. Let the agent's type set be $\Theta = \{\theta_c : c \in C\} \cup \{\theta_v : v \in V\}$, where C is the set of clauses in the MINSAT instance, and V is the set of variables. Let the probability distribution over these types be uniform. Let the outcome set be $O = \{o_0\} \cup \{o_c : c \in C\} \cup \{o_l : l \in L\}$, where L is the set of literals, that is, $L = \{+v : v \in V\} \cup \{-v : v \in V\}$. Let the notation $v(l) = v$ denote that v is the variable corresponding to the literal l, that is, $l \in \{+v, -v\}$. Let $l \in c$ denote that the literal l occurs in clause c. Then, let the agent's utility function be given by

- $u(\theta_c, o_l) = |\Theta| + 1$ for all $l \in L$ with $l \in c$;
- $u(\theta_c, o_l) = 0$ for all $l \in L$ with $l \notin c$;
- $u(\theta_c, o_c) = |\Theta| + 1$;
- $u(\theta_c, o_{c'}) = 0$ for all $c' \in C$ with $c \neq c'$;
- $u(\theta_v, o_l) = |\Theta|$ for all $l \in L$ with $v(l) = v$;
- $u(\theta_v, o_l) = 0$ for all $l \in L$ with $v(l) \neq v$;
- $u(\theta_v, o_c) = 0$ for all $c \in C$.

The goal of the AMD instance is $G = |\Theta| + \frac{|C|-K}{|\Theta|}$, where K is the goal of the MINSAT instance. We show the instances are equivalent. First, suppose there is a solution to the MINSAT instance. Let the assignment of truth values to the variables in this solution be given by the function $f : V \to L$ (where $v(f(v)) = v$ for all $v \in V$). Then, for every $v \in V$, let $o(\theta_v) = o_{f(v)}$ and $\pi(\theta_v) = |\Theta|$. For every $c \in C$, let $o(\theta_c) = o_c$; let $\pi(\theta_c) = |\Theta| + 1$ if c is not satisfied in the MINSAT solution, and $\pi(\theta_c) = |\Theta|$ if c is satisfied. It is straightforward to check that the IR constraint is satisfied. We now check that the agent has no incentive to misreport. If the agent's type is some θ_v, then any other report will give it an outcome that is no better, for a payment that is no less, so it has no incentive to misreport. If the agent's type is some θ_c where c is a satisfied clause, again, any other report will give it an outcome that is no better, for a payment that is no less, so it has no incentive to misreport. The final case to check is where the agent's type is some θ_c where c is an unsatisfied clause. In this case, we observe that for none of the types, reporting it leads to an outcome o_l for a literal $l \in c$, precisely because the clause is not satisfied in the MINSAT instance. Because also, no type besides θ_c leads to the outcome o_c, reporting any other type will give an outcome with utility 0, while still forcing a payment of at least $|\Theta|$ from the agent. Clearly the agent is better off reporting truthfully, for a total utility of 0. This establishes that the agent never has an incentive to misreport. Finally, we show that the goal is reached. If s is the number of satisfied clauses in the MINSAT solution (so that $s \leq K$), the expected payment from this mechanism is $\frac{|V||\Theta|+s|\Theta|+(|C|-s)(|\Theta|+1)}{|\Theta|} \geq \frac{|V||\Theta|+K|\Theta|+(|C|-K)(|\Theta|+1)}{|\Theta|} = |\Theta| + \frac{|C|-K}{|\Theta|} = G$. So there is a solution to the AMD instance.

Now suppose there is a solution to the AMD instance, given by an outcome function o and a payment function π. First, suppose there is some $v \in V$ such that $o(\theta_v) \notin \{o_{+v}, o_{-v}\}$. Then the utility that the agent derives from the given outcome for this type is 0, and hence, by IR, no payment can be extracted from the agent for this type. Because, again by IR, the maximum payment that can be extracted for any other type is $|\Theta| + 1$, it follows that the maximum expected payment that could be obtained is at most $\frac{(|\Theta|-1)(|\Theta|+1)}{|\Theta|} < |\Theta| < G$, contradicting that this is a solution to the AMD instance. It follows that in the solution to the AMD instance, for every $v \in V$, $o(\theta_v) \in \{o_{+v}, o_{-v}\}$. We can interpret this as an assignment of truth values to the variables: v is set to *true* if $o(\theta_v) = o_{+v}$, and to *false* if $o(\theta_v) = o_{-v}$. We claim this assignment is a solution to the MINSAT instance. By the IR constraint, the maximum payment we can extract from any type θ_v is $|\Theta|$. Because there can be no incentives for the agent to report falsely, for any clause c satisfied by the given assignment, the maximum payment we can extract for the corresponding type θ_c is $|\Theta|$. (For if we extracted more from this type, the agent's utility in this case would be less than 1; and if v is the variable satisfying c in the assignment, so that $o(\theta_v) = o_l$ where l occurs in c, then the agent would be better off reporting θ_v instead of the truthful report θ_c, to get an outcome worth $|\Theta| + 1$ to it while having to pay at most $|\Theta|$.) Finally, for any unsatisfied clause c, by the IR constraint, the maximum payment we can extract for the corresponding type θ_c is $|\Theta| + 1$. It follows that the expected payment from our mechanism is at most $\frac{V|\Theta|+s|\Theta|+(|C|-s)(|\Theta|+1)}{\Theta}$, where s is the number of satisfied clauses. Because our mechanism achieves the goal, it follows that $\frac{V|\Theta|+s|\Theta|+(|C|-s)(|\Theta|+1)}{\Theta} \geq G$, which by simple algebraic manipulations is equivalent to $s \leq K$. So there is a solution to the MINSAT instance. ■

Because payment-maximizing AMD is just the special case of AMD for a self-interested designer where the designer has no preferences over the outcome chosen, this immediately implies hardness for the general case of AMD for a self-interested designer where payments are possible. However, it does not yet imply hardness for the special case where payments are not possible. We will prove hardness in this case in the next section.

5 Self-interested deterministic AMD without payments is hard

In this section we demonstrate that it is $\mathcal{NP}$-complete to design a deterministic mechanism that maximizes the expectation of the designer's objective when payments are not possible. We show that this problem is hard even in the single-agent setting, thereby immediately showing it hard for both IR concepts, for both solution concepts.

Theorem 2 *Without payments, deterministic AMD for a self-interested designer is $\mathcal{NP}$-complete, even for a single agent, even with a uniform distribution over types.*

Proof: It is easy to show that the problem is in $\mathcal{NP}$. To show $\mathcal{NP}$-hardness, we reduce an arbitrary MINSAT instance to the following single-agent self-interested deterministic AMD without payments instance. Let the agent's type set be $\Theta = \{\theta_c : c \in C\} \cup \{\theta_v : v \in V\}$, where C is the set of clauses in the MINSAT instance, and V is the set of variables. Let the probability distribution over these types be uniform. Let the outcome set be $O = \{o_0\} \cup \{o_c : c \in C\} \cup \{o_l : l \in L\} \cup \{o^*\}$, where L is the set of literals, that is, $L = \{+v : v \in V\} \cup \{-v : v \in V\}$. Let the notation $v(l) = v$ denote that v is the variable corresponding to the literal l, that is, $l \in \{+v, -v\}$. Let $l \in c$ denote that the literal l occurs in clause c. Then, let the agent's utility function be given by

- $u(\theta_c, o_l) = 2$ for all $l \in L$ with $l \in c$;
- $u(\theta_c, o_l) = -1$ for all $l \in L$ with $l \notin c$;
- $u(\theta_c, o_c) = 2$;
- $u(\theta_c, o_{c'}) = -1$ for all $c' \in C$ with $c \neq c'$;
- $u(\theta_c, o^*) = 1$;
- $u(\theta_v, o_l) = 1$ for all $l \in L$ with $v(l) = v$;
- $u(\theta_v, o_l) = -1$ for all $l \in L$ with $v(l) \neq v$;
- $u(\theta_v, o_c) = -1$ for all $c \in C$;
- $u(\theta_v, o^*) = -1$.

Let the designer's objective function be given by

- $g(o^*) = |\Theta| + 1$;
- $g(o_l) = |\Theta|$ for all $l \in L$;
- $g(o_c) = |\Theta|$ for all $c \in C$.

The goal of the AMD instance is $G = |\Theta| + \frac{|C|-K}{|\Theta|}$, where K is the goal of the MINSAT instance. We show the instances are equivalent. First, suppose there is a solution to the MINSAT instance. Let the assignment of truth values to the variables in this solution be given by the function $f : V \to L$ (where $v(f(v)) = v$ for all $v \in V$). Then, for every $v \in V$, let $o(\theta_v) = o_{f(v)}$. For every $c \in C$ that is satisfied in the MINSAT solution, let $o(\theta_c) = o_c$; for every unsatisfied $c \in C$, let $o(\theta_c) = o^*$. It is straightforward to check that the IR constraint is satisfied. We now check that the agent has no incentive to misreport. If the agent's type is some θ_v, it is getting the maximum utility for that type, so it has no incentive to misreport. If the agent's type is some θ_c where c is a satisfied clause, again, it is getting the maximum utility for that type, so it has no incentive to misreport. The final case to check is where the agent's type is some θ_c where c is an unsatisfied clause. In this case, we observe that for none of the types, reporting it leads to an outcome o_l for a literal $l \in c$, precisely because the clause is not satisfied in the MINSAT instance. Because also, no type leads to the outcome o_c, there is no outcome that the mechanism ever selects that would give the agent utility greater than 1 for type θ_c, and hence the agent has no incentive to report falsely. This establishes that the agent never has an incentive to misreport. Finally, we show that the goal is reached. If s is the number of satisfied clauses in the MINSAT solution (so that $s \leq K$), then the expected value of the designer's objective function is $\frac{|V||\Theta|+s|\Theta|+(|C|-s)(|\Theta|+1)}{|\Theta|} \geq \frac{|V||\Theta|+K|\Theta|+(|C|-K)(|\Theta|+1)}{|\Theta|} = |\Theta| + \frac{|C|-K}{|\Theta|} = G$. So there is a solution to the AMD instance.

Now suppose there is a solution to the AMD instance, given by an outcome function o. First, suppose there is some $v \in V$ such that $o(\theta_v) \notin \{o_{+v}, o_{-v}\}$. The only other outcome that the mechanism is allowed to choose under the IR constraint is o_0. This has an objective value of 0, and because the highest value the objective function ever takes is $|\Theta| + 1$, it follows that the maximum expected value of the objective function that could be obtained is at most $\frac{(|\Theta|-1)(|\Theta|+1)}{|\Theta|} < |\Theta| < G$, contradicting that this is a solution to the AMD instance. It follows that in the solution to the AMD instance, for every $v \in V$, $o(\theta_v) \in \{o_{+v}, o_{-v}\}$. We can interpret this as an assignment of truth values to the variables: v is set to *true* if $o(\theta_v) = o_{+v}$, and to *false* if $o(\theta_v) = o_{-v}$. We claim this assignment is a solution to the MINSAT instance. By the above, for any type θ_v, the value of the objective function in this mechanism will be $|\Theta|$. For any clause c satisfied by the given assignment, the value of the objective function in the case where the agent reports type θ_c will be at most $|\Theta|$. (This is because we cannot choose the outcome o^* for such a type, as in this case the agent would have an incentive to report θ_v instead, where v is the variable satisfying c in the assignment (so that $o(\theta_v) = o_l$ where l occurs in c).) Finally, for any unsatisfied clause c, the maximum value the objective function can take in the case where the agent reports type θ_c is $|\Theta| + 1$, simply because this is the largest value the function ever takes. It follows that the expected value of the objective function for our mechanism is at most $\frac{V|\Theta|+s|\Theta|+(|C|-s)(|\Theta|+1)}{\Theta}$, where s is the number of satisfied clauses. Because our mechanism achieves the goal, it follows that $\frac{V|\Theta|+s|\Theta|+(|C|-s)(|\Theta|+1)}{\Theta} \geq G$, which by simple algebraic manipulations is equivalent to $s \leq K$. So there is a solution to the MINSAT instance. ■

Both of our hardness results relied on the constraint that the mechanism should be deterministic. In the next section, we show that the hardness of design disappears when we allow for randomization in the mechanism.

6 Randomized AMD for a self-interested designer is easy

We now show how allowing for randomization over the outcomes makes the problem of self-interested AMD tractable through linear programming, for any constant number of agents.

Theorem 3 *Self-interested randomized AMD with a constant number of agents is solvable in polynomial time by linear programming, both with and without payments, both for* ex post *and* ex interim *IR, and both for implementation in dominant strategies and for implementation in Bayes-Nash equilibrium—even if the types are correlated.*

Proof: Because linear programs can be solved in polynomial time, all we need to show is that the number of variables and equations in our program is polynomial for any constant number of agents—that is, exponential only in N. Throughout, for purposes of determining the size of the linear program, let $T = \max_i\{|\Theta_i|\}$. The variables of our linear program will be the probabilities $(p(\theta_1, \theta_2, \ldots, \theta_N))(o)$ (at most $T^N|O|$ variables) and the payments $\pi_i(\theta_1, \theta_2, \ldots, \theta_N)$ (at most NT^N variables). (We show the linear program for the case where payments are possible; the case without payments is easily obtained from this by simply omitting all the payment variables in the program, or by adding additional constraints forcing the payments to be 0.)

First, we show the IR constraints. For *ex post* IR, we add the following (at most NT^N) constraints to the LP:

- For every $i \in \{1, 2, \ldots, N\}$, and for every $(\theta_1, \theta_2, \ldots, \theta_N) \in \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N$, we add $(\sum_{o\in O} (p(\theta_1, \theta_2, \ldots, \theta_N))(o)u(\theta_i, o)) - \pi_i(\theta_1, \theta_2, \ldots, \theta_N) \geq 0$.

For *ex interim* IR, we add the following (at most NT) constraints to the LP:

- For every $i \in \{1, 2, \ldots, N\}$, for every $\theta_i \in \Theta_i$, we add $\sum_{\theta_1,\ldots,\theta_N} \gamma(\theta_1, \ldots, \theta_N|\theta_i)((\sum_{o\in O} (p(\theta_1, \theta_2, \ldots, \theta_N))(o)u(\theta_i, o)) - \pi_i(\theta_1, \theta_2, \ldots, \theta_N)) \geq 0$.

Now, we show the solution concept constraints. For implementation in dominant strategies, we add the following (at most NT^{N+1}) constraints to the LP:

- For every $i \in \{1, 2, \ldots, N\}$, for every $(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_N) \in \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N$, and for every alternative type report $\hat{\theta}_i \in \Theta_i$, we add the constraint $(\sum_{o\in O} (p(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_N))(o)u(\theta_i, o)) - \pi_i(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_N) \geq (\sum_{o\in O} (p(\theta_1, \theta_2, \ldots, \hat{\theta}_i, \ldots, \theta_N))(o)u(\theta_i, o)) - \pi_i(\theta_1, \theta_2, \ldots, \hat{\theta}_i, \ldots, \theta_N)$.

Finally, for implementation in Bayes-Nash equilibrium, we add the following (at most NT^2) constraints to the LP:

- For every $i \in \{1, 2, ..., N\}$, for every $\theta_i \in \Theta_i$, and for every alternative type report $\hat{\theta}_i \in \Theta_i$, we add the constraint $\sum_{\theta_1,\ldots,\theta_N} \gamma(\theta_1, ..., \theta_N|\theta_i)((\sum_{o\in O} (p(\theta_1, \theta_2, ..., \theta_i, ..., \theta_N))(o)u(\theta_i, o)) - \pi_i(\theta_1, \theta_2, ..., \theta_i, ..., \theta_N)) \geq \sum_{\theta_1,\ldots,\theta_N} \gamma(\theta_1, ..., \theta_N|\theta_i)((\sum_{o\in O} (p(\theta_1, \theta_2, ..., \hat{\theta}_i, ..., \theta_N))(o)u(\theta_i, o)) - \pi_i(\theta_1, \theta_2, ..., \hat{\theta}_i, ..., \theta_N))$.

All that is left to do is to give the expression the designer is seeking to maximize, which is:

- $\sum_{\theta_1,\ldots,\theta_N} \gamma(\theta_1, ..., \theta_N)((\sum_{o\in O} (p(\theta_1, \theta_2, ..., \theta_i, ..., \theta_N))(o)g(o)) + \sum_{i=1}^{N} \pi_i(\theta_1, \theta_2, ..., \theta_N))$.

As we indicated, the number of variables and constraints is exponential only in N, and hence the linear program is of polynomial size for constant numbers of agents. Thus the problem is solvable in polynomial time. ∎

7 Related research on computational complexity in mechanism design

There has been considerable recent interest in mechanism design in computer science. Some of it has focused on issues of computational complexity, but most of that work has strived toward designing mechanisms that are easy to *execute* (e.g. [13, 9]), rather than studying the complexity of *designing* the mechanism. The closest piece of earlier work studied the complexity of automated mechanism design *by a benevolent designer* [3]. Roughgarden has studied the complexity of designing a good network topology for agents that selfishly choose the links they use [14]. This is related to mechanism design, but differs significantly in that the designer only has restricted control over the rules of the game because there is no party that can impose the outcome (or side payments). Also, there is no explicit reporting of preferences.

8 Conclusions and future research

Often, an outcome must be chosen on the basis of the preferences reported by a group of agents. The key difficulty is that the agents may report their preferences insincerely to make the chosen outcome more favorable to themselves. *Mechanism design* is the art of designing the rules of the game so that the agents are motivated to report their preferences truthfully, and a desirable outcome is chosen. In a recently emerging approach—called *automated mechanism design*—a mechanism is computed for the specific preference aggregation setting at hand. This has several advantages, but the downside is that the mechanism design optimization problem needs to be solved anew each time. Unlike earlier work on automated mechanism design that studied a benevolent designer, in this paper we studied automated mechanism design problems *where the designer is self-interested*—a setting much more relevant for electronic commerce. In this setting, the center cares only about which outcome is chosen and what payments are made to it. The reason that the agents' preferences are relevant is that the center is constrained to making each agent at least as well off as the agent would have been had it not participated in the mechanism. In this setting, we showed that designing an optimal deterministic mechanism is $\mathcal{NP}$-complete in two important special cases: when the center is interested only in the payments made to it, and when payments are not possible and the center is interested only in the outcome chosen. These hardness results imply hardness in all more general automated mechanism design settings with a self-interested designer. The hardness results apply whether the individual rationality (participation) constraints are applied *ex interim* or *ex post*, and whether the solution concept is dominant strategies implementation or Bayes-Nash equilibrium implementation. Finally, we showed that allowing randomization in the mechanism makes the design problem in all these settings computationally easy.

Future research includes studying automated mechanism design with a self-interested designer in more restricted settings such as auctions (where the designer's objective may include preferences about which bidder should receive the good—as well as payments). We also want to study the complexity of automated mechanism design in settings where the outcome and type spaces have special structure so they can be represented more concisely. Finally, we plan to assemble a data set of real-world mechanism design problems—both historical and current—and apply automated mechanism design to those problems.

References

[1] Kenneth Arrow. The property rights doctrine and demand revelation under incomplete information. In M Boskin, editor, *Economics and human welfare*. New York Academic Press, 1979.

[2] E H Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[3] Vincent Conitzer and Tuomas Sandholm. Complexity of mechanism design. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 103–110, Edmonton, Canada, 2002.

[4] Vincent Conitzer and Tuomas Sandholm. Computational criticisms of the revelation principle, 2002. Draft.

[5] C d'Aspremont and L A Gérard-Varet. Incentives and incomplete information. *Journal of Public Economics*, 11:25–45, 1979.

[6] A Gibbard. Manipulation of voting schemes. *Econometrica*, 41:587–602, 1973.

[7] Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[8] R Kohli, R Krishnamurthi, and P Mirchandani. The minimum satisfiability problem. *SIAM Journal of Discrete Mathematics*, 7(2):275–283, 1994.

[9] Daniel Lehmann, Lidian Ita O'Callaghan, and Yoav Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. *Journal of the ACM*, 2003. To appear. Early version appeared in ACMEC-99.

[10] Andreu Mas-Colell, Michael Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[11] Eric S Maskin and John Riley. Optimal multi-unit auctions. In Frank Hahn, editor, *The Economics of Missing Markets, Information, and Games*, chapter 14, pages 312–335. Clarendon Press, Oxford, 1989.

[12] Roger B Myerson. Optimal auction design. *Mathematics of Operation Research*, 6:58–73, 1981.

[13] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001. Early version in STOC-99.

[14] Tim Roughgarden. Designing networks for selfish users is hard. In *FOCS*, 2001.

[15] Tuomas Sandholm. Issues in computational Vickrey auctions. *International Journal of Electronic Commerce*, 4(3):107–129, 2000. Special Issue on Applying Intelligent Agents for Electronic Commerce. A short, early version appeared at the Second International Conference on Multi–Agent Systems (ICMAS), pages 299–306, 1996.

[16] M A Satterthwaite. Strategy-proofness and Arrow's conditions: existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.

[17] W Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.