Automated mechanism design with a structured outcome space

Vincent Conitzer
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Previous research on automated mechanism design (proposed in UAI-02) assumed that the outcome space was flatly represented, which makes that work inapplicable if the outcome space is exponential, as it is, for example, in multi-item auctions. This paper introduces (to our knowledge) the first more concise representation for the problem, which relies on decomposing the outcome space into distinct *issues*. While the decomposability makes the input to the design problem polynomial, we show that it is NP-complete (even with a single agent with only two types) to design a mechanism that maximizes one of the following objectives: 1) expected social welfare when payments are not possible, 2) a general objective function even when payments are possible, and 3) expected payments to the center (designer). We show that the NP-hardness is only weak by developing a pseudopolynomial algorithm for the former two single-agent mechanism design problems with any constant number of types. Finally, we show that when designing randomized mechanisms, we can exploit the structure of the representation and even solve the general problem in polynomial time for any constant number of agents for any growing number of types.

1 Introduction

In multiagent settings, often an *outcome* must be chosen based on the preferences of 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 aggregator has to elicit the agents' preferences from them. Unfortunately, an agent may 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—causing the outcome to be socially less desirable. *Mechanism design* is the art of designing the rules of the game 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. Mechanism design has traditionally been a manual endeavor in economics [12] as well as computer science (e.g. [16, 17]).

Recently (UAI-02), we proposed the idea of *automated* mechanism design, where an algorithm creates a customized mechanism for the setting at hand [3, 4, 5]. This approach has several advantages. First, it can be used even in settings that do not satisfy the assumptions of the classical mechanisms (such as availability of side payments and/or the objective being social welfare [21, 2, 10, 6, 1, 7]). Second, it can allow one to circumvent impossibility results (such as the Gibbard-Satterthwaite theorem [9, 20]) 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 better outcomes or stronger nonmanipulability guarantees) than classical mechanisms because the mechanism capitalizes on the particulars of the setting (the probabilistic information that the mechanism designer has about the agents' types).[1] Fourth, the burden of design is shifted from humans to a machine.

The prior paper on automated mechanism design assumed that the outcome space was flatly represented, that is, all the possible outcomes were enumerated. This makes that work inapplicable if the outcome space is exponential, as it is, for example, in multi-item auctions. In contrast, this paper (to our knowledge) is the first to examine automated mechanism design when the outcome space can be more con-

[1]Some of the manual mechanism design work has resulted in mechanisms that also capitalize on this probabilistic information in specific settings, e.g., in restricted auction settings where the objective is the seller's revenue [14, 13].

cisely represented. Specifically, we allow for the decomposition of the outcome space into distinct *issues*, across which there are no interacting effects with regards to either the agents' utility functions or the designer's objective. (A subtle point is that this does *not* mean that we can simply solve each issue independently. We will discuss this later.) We can represent any outcome space in our representation, and in many realistic settings, our representation is exponentially more concise than the flat representation.

This type of situation occurs across the AI literature, and especially in the literature modeling uncertainty. For instance, specifying a joint probability distribution over a number of variables, in general, requires exponential space. Nevertheless, there are models such as Bayesian networks that can represent many real distributions concisely. As another example, in a combinatorial auction, each bidder in the general case has a valuation for each of the exponentially many bundles of goods, but there are models that allow for concise preference representation in many real settings (e.g., [19, 8, 15, 11]).

Computational complexity is usually measured as a function of the input size. Therefore, for concise representations, the problem is often in a higher complexity class. For example, probabilistic inference is in P for the flat representation, but NP-complete for Bayesian networks; optimal winner determination is in P (using dynamic programming) for flatly represented combinatorial auctions [18], but NP-complete [18] (and inapproximable [19]) for concise representations.

In this paper we first show that for our concise representation, the three most important problems in designing deterministic mechanisms are NP-complete *even when there is only one agent—with only two types*.[2] This is in contrast with flatly represented outcome spaces, where these problems are solvable in polynomial time with a simple brute force algorithm for any constant number of agents with any constant number of types [3]. We show that the NP-hardness is only weak in two of the three problems, by developing a pseudopolynomial algorithm for those single-agent mechanism design problems, for any constant number of types. We then consider randomized mechanisms. In flatly represented outcome spaces, such problems can typically be solved for any constant number of agents and any growing number of types in polynomial time with a straightforward linear program. However, with our structured representation, this program has an exponential number of variables. Nevertheless, we show how to use the structure of our representation to reduce the variables to a polynomial number. The resulting linear program has polynomial size and can be used to solve problems with any constant number of agents and any growing number of types in polynomial time—*in the size of our concise representation.*

[2]Problems with small type spaces derive their importance from situations where preferences are highly clustered, e.g. when a voter's stand on issues is entirely along the lines of the Democratic (or Republican) party, or situations in which too detailed preference revelation introduces privacy issues.

2 The model and definitions

We first review the automated mechanism design definitions. (These are the same as in our previous work [3, 4, 5].) After that, we present our concise problem representation.

2.1 Review of automated mechanism design

Definition 1 *In a* mechanism design setting, *we are given 1. a finite set of outcomes O and a finite set of N agents; 2. for each agent i, a) a finite set of types Θ_i, b) a probability distribution γ_i over Θ_i (in the case of correlated types, there is a single joint distribution γ over $\Theta_1 \times \ldots \times \Theta_N$), c) a utility function $u_i : \Theta_i \times O \to \mathbb{N}$; 3. the designer's objective function $g : O \times \Theta_1 \times \ldots \times \Theta_N \to \mathbb{N}$; 4. a linear factor, a, such that the designer attempts to maximize the expectation of $g + a \sum_{1 \leq i \leq N} \pi_i$, where π_i is the payment by agent i.*

Thus, the designer can aim to maximize complex combinations of interests regarding how the outcome function relates to the agents' utilities, and payments made to the designer. We show hardness arises already in the following special cases:

- The designer's objective is *social welfare*, and the designer does not care about payments. (Social welfare is the sum of all agents' utilities, plus an additional term u_0 representing the interest of parties "outside" the mechanism that do not report a type.) Thus, $g = \sum\limits_{1 \leq i \leq N} u_i(\theta_i, o) + u_0(o)$ and $a = 0$.
- The designer's objective is a general function g, but the designer still does not care about payments ($a = 0$).
- The designer is only concerned with maximizing the payments ($g = 0$ everywhere, and $a = 1$).

We now define the kinds of mechanisms that we consider.

Definition 2 *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 $p : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N \to \mathcal{P}(O)$, where $\mathcal{P}(O)$ 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 $p : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N \rightarrow \mathcal{P}(O)$, and for each agent i, a payment selection function $\pi_i : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N \rightarrow \mathbb{R}$.*[3]

The mechanism designer is constrained in choosing a mechanism. The first type of constraint states that the utility of each agent has to be at least as great as the agent's fallback utility, that is, the utility that the agent would receive if it did not participate in the mechanism. (We assume that each agent's fallback utility is zero for each one of its types.[4]) This type of constraint is called an *individual rationality (IR)* constraint. *Ex interim* IR means that for every one of its types, each agent's expected utility (averaged over the other agents' possible types) is nonnegative. In other words, each agent is happy to participate when it is still uncertain about the other agents' private information. *Ex post* IR means that for every one of its types and all types of the other agents, each agent's utility is nonnegative. In other words, each agent is always happy to participate even if it knows the other agents' type revelations.

The second type of constraint states that for each agent, the agent should never have an incentive to misreport its type. The two most common variants of this are *implementation in dominant strategies*, and *implementation in Bayes-Nash equilibrium* [12]. A mechanism is said to *implement its outcome and payment functions in dominant strategies* if, for each agent, truthtelling is always optimal even if the agent knows the types reported by the other agents. 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 the other agents' type revelations (but the probabilistic information about all agents' types is common knowledge among the agents and the designer) and the other agents are telling the truth.

Now we can define the computational problem of automated mechanism design.

Definition 3 *(AUTOMATED-MECHANISM-DESIGN (AMD)) We are given a mechanism design setting, an IR notion (*ex interim, ex post*), and an IC notion (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 IC notion, and gives an expected value of at least G for the objective function.*

[3]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.

[4]This is without loss of generality because we can simply 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 [12].

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 notions 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, these two IC notions coincide here. This observation will prove to be a useful tool in proving hardness results: *we prove computational hardness in the single-agent setting, immediately implying hardness for both IR notions, for both IC notions, for any number of agents.*

2.2 Decomposition of the outcome space

We are now ready to present the first contribution of this paper: our concise representation based on decomposition of the outcome space.

Definition 4 *$O = O_1 \times O_2 \times \ldots \times O_r$ is a* valid decomposition *of O (where r is the number of* issues*) if the following two conditions hold:*

- *For each agent i, for each $1 \leq k \leq r$ there exists a function $u_i^k : \Theta_i \times O_k \rightarrow \mathbb{N}$ such that $u_i(\theta_i, (o^1, \ldots, o^r)) = \sum_{1 \leq k \leq r} u_i^k(\theta_i, o^k)$;*
- *For each $1 \leq k \leq r$ there exists a function $g^k : \Theta_1 \times \ldots \times \Theta_n \times O_k \rightarrow \mathbb{N}$ such that $g(\theta_1, \ldots, \theta_n, (o^1, \ldots, o^r)) = \sum_{1 \leq k \leq r} g^k(\theta_1, \ldots, \theta_n, o^k)$.*

We observe that when g is a social welfare function, the first condition implies the second, because if the first condition holds, $g(\theta_1, \ldots, \theta_n, (o^1, \ldots, o^r)) = \sum_{1 \leq i \leq n} u_i(\theta_i, (o^1, \ldots, o^r)) = \sum_{1 \leq i \leq n} \sum_{1 \leq k \leq r} u_i^k(\theta_i, o^k) = \sum_{1 \leq k \leq r} \sum_{1 \leq i \leq n} u_i^k(\theta_i, o^k)$, so that we can define $g^k(\theta_1, \ldots, \theta_n, o^k) = \sum_{1 \leq i \leq n} u_i^k(\theta_i, o^k)$.

We call automated mechanism design with a valid decomposition *multi-issue automated mechanism design*. It may seem that we can solve a multi-issue AMD instance simply by solving the AMD problem for each individual issue separately. However, doing so will in general not give the optimal mechanism. The reason is that in general, the designer may use one issue to tailor the incentives to get better results on another issue. For example, in an auction setting, one could think of the allocation as one issue, and the payments as another issue. Even when the designer is only concerned with bringing about the optimal allocation, the payments are still a useful instrument to give the bidders an incentive to bid truthfully. (We caution the reader that apart from this minor deviation, we do not consider the payments to be part of the outcome space O.) The hardness results later in this paper will also imply that solving the AMD problem separately for each issue does not give the optimal solution. (The multi-issue AMD problem is NP-complete even in settings where the corresponding single-issue AMD problem is in P, so if the approach of solving the problem separately for each issue were optimal, we would have shown that P=NP.)

3 Example: Multi-item auctions

Consider auctioning a set of distinguishable items. If each of the n items can be allocated to any of N agents (or to no agent at all), the outcome space O has size $(N+1)^n$ (one for each possible allocation). If, for every bidder, the bidder's valuation for any bundle of items is the sum of the bidder's valuations of the individual items in the bundle, then we can decompose the outcome space as $O = O_1 \times O_2 \times \ldots \times O_n$, where $O_j = \{0, 1, 2, \ldots, N\}$ is the set of all possible allocations for item j (0 indicating it goes to no agent). Agent i's utility function can be written as $u_i((o^1, o^2, \ldots, o^n)) = \sum_{j \in \{1,\ldots,n\}} u_i^j(o^j)$ where u_i^j is given by $u_i^j(i) = v_i(j)$ and $u_i^j(m) = 0$ for $m \neq i$, where $v_i(j)$ is agent i's valuation for item j.

Two extensions of this that also allow for decomposable outcome spaces are the following:

• An agent, if it does not receive an item, still cares which agent (if any) receives that item. Here we no longer always have $u_i^j(m) = 0$ for $m \neq i$. For example, John may prefer it if the museum wins the painting rather than a private collector, because in the former case he can still see the painting.

• Some items exhibit substitutability or complementarity (so an agent's valuation for a bundle is not the sum of its valuations of the individual items in the bundle), but the items can be partitioned into subsets so that there are no substitution or complementarity effects across subsets in the partition. In this case, we can still decompose the outcome space over these subsets. For example, a plane trip, a hotel stay, a cell phone and a pager are all for sale. The plane trip and the hotel stay are each worthless without the other: they are perfect complements. The cell phone and the pager each reduce the value of having the other: they are (partial) substitutes. But the value of the plane trip or the hotel stay has nothing to do with whether one also gets the cell phone or the pager. Thus, we decompose the outcome space into two issues, one indicating the winners of the plane trip and hotel stay, and one indicating the winners of the cell phone and the pager.

In each of these settings, the approach of this paper can be used directly to maximize any objective the designer has. (This requires that the vaulations lie in a finite interval and are discretized.)

4 Hardness results

In this section we show that for the concise representation, the three most important variants of the problem of designing a deterministic mechanism are NP-complete. (Membership in NP is guaranteed only if the number of agents is constant.) While hardness results have already been established for some of these problems in the general (single-issue) representation [3, 4, 5], under the multi-issue representation, hardness occurs even in much more restricted settings (with small type spaces and a small outcome space for each issue). Of course, hardness in a restricted setting implies hardness in any more general setting.

Theorem 1 *Multi-issue automated deterministic mechanism design without payments to maximize social welfare is NP-complete even when there is only a single type-reporting agent (but there is also another interested agent who does not report a type), even when there are only two possible types and $|O_i| = 2$ for all i. (Membership in NP is guaranteed only if the number of agents is constant.)*

Proof: The problem is in NP because we can nondeterministically generate the full outcome selection function o (as long as the number of agents is constant, because otherwise there are exponentially many type vectors). To show NP-hardness, we reduce an arbitrary KNAPSACK instance (given by a set of pairs $\{(c_j, v_j)\}_{j \in \{1,\ldots,m\}}$, a cost limit C, and a value goal V) to the following single-agent deterministic multi-issue AMD instance, where payments are not allowed and the objective is social welfare. Let the number of issues be $r = m + 1$. For every $j \in \{1, \ldots, m+1\}$, we have $O_j = \{t, f\}$. The agent's type set, over which there is a uniform distribution, is $\Theta = \{\theta^1, \theta^2\}$, and the agent's utility function $u = \sum_{k \in \{1,\ldots,r\}} u^k$ is given by:

- For all $k \in \{1, \ldots, m\}$, $u^k(\theta^1, t) = AB$ where $A = 2 \sum\limits_{j \in \{1,\ldots,m\}} c_j$ and $B = 2 \sum\limits_{j \in \{1,\ldots,m\}} v_j$; and $u^k(\theta^1, f) = 0$.
- $u^{m+1}(\theta^1, t) = u^{m+1}(\theta^1, f) = 0$.
- For all $k \in \{1, \ldots, m\}$, $u^k(\theta^2, t) = c_k$, and $u^k(\theta^2, f) = 0$.
- $u^{m+1}(\theta^2, t) = C$, and $u^{m+1}(\theta^2, f) = 0$.

The part of the social welfare that does not correspond to any agent in the game is given by $u_0 = \sum_{k \in \{1,\dots,r\}} u_0^k$ where

- For all $k \in \{1,\dots,m\}$, $u_0^k(t) = 0$, and $u^k(f) = v_k A$.
- $u_0^{m+1}(t) = u_0^{m+1}(f) = 0$.

The goal social welfare is given by $G = \frac{A(mB+V)}{2}$. We show the two instances are equivalent. First suppose there is a solution to the KNAPSACK instance, that is, a subset S of $\{1,\dots,m\}$ such that $\sum_{j \in S} c_j \leq C$ and $\sum_{j \in S} v_j \geq V$. Then consider the following mechanism:

- For all $k \in \{1,\dots,m\}$, $o^k(\theta^1) = t$.
- For $k \in \{1,\dots,m\}$, $o^k(\theta^2) = f$ if $k \in S$, $o^k(\theta^2) = t$ otherwise.
- $o^{m+1}(\theta^1) = f$, and $o^{m+1}(\theta^2) = t$.

First we show there is no incentive for the agent to misreport. If the agent has type θ^1, then it is getting the best possible outcome for each issue by reporting truthfully, so there is certainly no incentive to misreport. If the agent has type θ^2, reporting truthfully gives it a utility of $C + \sum_{j \in \{1,\dots,m\}, \notin S} c_j$, whereas reporting θ^1 instead gives it a utility of $\sum_{j \in \{1,\dots,m\}} c_j$; so the marginal utility of misreporting is $-C + \sum_{j \in S} c_j \leq -C + C = 0$. Hence there is no incentive to misreport. Now we show that the goal social welfare is reached. If the agent has type θ^1, the social welfare will be mAB. If it has type θ^2, it will be $\sum_{j \in S} v_j A + \sum_{j \in \{1,\dots,m\}, \notin S} c_j + C \geq \sum_{j \in S} v_j A \geq VA$. Hence the expected social welfare is at least $\frac{mAB+VA}{2} = G$. So there is a solution to the AMD instance. Now suppose there is a solution to the AMD instance. If it were the case that, for some $j \in \{1,\dots,m\}$, $o^j(\theta^1) = f$, then the maximum social welfare that could possibly be obtained (even if we did not worry about misreporting) would be $\frac{(m-1)AB+v_jA}{2} + \frac{\sum_{j \in \{1,\dots,m\}} v_jA+C}{2} = \frac{(m-1)AB+\frac{AB}{2}+v_jA+C}{2} < \frac{mAB+VA}{2} = G$. Thus, for all $j \in \{1,\dots,m\}$, $o^k(\theta^1) = t$. Now, let $S = \{j \in \{1,\dots,m\} : o^j(\theta^2) = f\}$. Then, if the agent has type θ^2 and reports truthfully, it will get utility at most $C + \sum_{j \in \{1,\dots,m\}, \notin S} c_j$, as opposed to the at least $\sum_{j \in \{1,\dots,m\}} c_j$ that it could get for this type by reporting θ^1 instead. Because there is no incentive to misreport in the mechanism, it follows that $\sum_{j \in S} c_j \leq C$. Also, the total social welfare obtained by the mechanism is at most $\frac{mAB+\sum_{j \in S} v_jA + \sum_{j \in \{1,\dots,m\}, \notin S} c_j + C}{2}$. Because $\sum_{j \in \{1,\dots,m\}, \notin S} c_j + C < A$, and all the other terms in the numerator are some integer times A, it follows that this fraction is greater than or equal to the goal $\frac{mAB+VA}{2}$ (where the numerator is also an integer times A) if and only if $\sum_{j \in S} v_j \geq V$—and this must be the case because by assumption, the mechanism is a solution to the AMD instance. Thus S is a solution to the KNAPSACK instance. ■

Theorem 2 *Multi-issue automated deterministic mechanism design with payments for general objective functions g (even with no interest in the payments, i.e., $a = 0$) is NP-complete even for a single agent, even when there are only two possible types and $|O_i| = 2$ for all i. (Membership in NP is guaranteed only if the number of agents is constant.)*

Proof: We omit this proof because of space constraint. ■

Theorem 3 *Multi-issue automated deterministic mechanism design where the objective is to maximize the payments (from the agents to the designer) is NP-complete even for a single agent, even when there are only two possible types and $|O_i| = 2$ for all i. (Membership in NP is guaranteed only if the number of agents is constant.)*

Proof: It is easy to see that the problem is in NP. (We can nondeterministically guess an outcome function, after which setting the payments is a linear programming problem and can hence be done in polynomial time.) To show NP-hardness, we reduce an arbitrary KNAPSACK instance (given by a set of pairs $\{(c_j, v_j)\}_{j \in \{1,\dots,m+1\}}$, a cost limit C, and a value goal V) to the following single-agent deterministic multi-issue AMD instance, where we seek to maximize the expected payments from the agent. Let the number of issues be $r = m+1$. For every $j \in \{1,\dots,m\}$, we have $O_j = \{t, f\}$. The agent's type set, over which there is a uniform distribution, is $\Theta = \{\theta^1, \theta^2\}$, and the agent's utility function $u = \sum_{k \in \{1,\dots,r\}} u^k$ is given by:

- For all $k \in \{1,\dots,m\}$, $u^k(\theta^1, t) = c_k A$ where $A = 4 \sum_{j \in \{1,\dots,m\}} v_j$; and $u^k(\theta^1, f) = 0$.
- $u^{m+1}(\theta^1, t) = 0$, and $u^{m+1}(\theta^1, f) = -CA$.
- For all $k \in \{1,\dots,m\}$, $u^k(\theta^2, t) = v_k$, and $u^k(\theta^2, f) = 0$.
- $u^{m+1}(\theta^2, t) = 0$, and $u^{m+1}(\theta^2, f) = 0$.

The goal expected revenue is given by $G = \frac{AB+V}{2}$, where $B = \sum_{j \in \{1,\dots,m\}} c_j$. We show the two instances are equivalent. First suppose there is a solution to the KNAPSACK instance, that is, a subset S of $\{1,\dots,m\}$ such that

$\sum_{j \in S} c_j \leq C$ and $\sum_{j \in S} v_j \geq V$. Then consider the following mechanism. Let the outcome function be

- For all $k \in \{1, \ldots, m\}$, $o^k(\theta^1) = t$.
- For $k \in \{1, \ldots, m\}$, $o^k(\theta^2) = t$ if $k \in S$, $o^k(\theta^2) = f$ otherwise.
- $o^{m+1}(\theta^1) = t,\ o^{m+1}(\theta^2) = f$.

Let the payment function be $\pi(\theta^1) = AB$, $\pi(\theta^2) = \sum_{j \in S} v_j$. First, to see that the IR constraint is satisfied, observe that for each type, the mechanism extracts exactly the agent's utility obtained from the outcome function. Second, we show there is no incentive for the agent to misreport. If the agent has type θ^1, reporting θ^2 instead gives a utility (including payments) of $-CA + \sum_{j \in S} c_j A - \sum_{j \in S} v_j \leq -CA + CA - \sum_{j \in S} v_j < 0$, which is what the agent would have got for reporting truthfully. If the agent has type θ^2, reporting θ^1 instead gives a utility (including payments) of $\frac{A}{4} - AB < 0$, which is what the agent would have got for reporting truthfully. Hence there is no incentive to misreport. Third, the goal expected payment is reached because $\frac{AB + \sum_{j \in S} v_j}{2} \geq \frac{AB+V}{2} = G$. So there is a solution to the AMD instance. Now suppose there is a solution to the AMD instance. The maximum utility that the agent can get from the outcome function if it has type θ^2 is $\frac{A}{4}$, and by the IR constraint this is the maximum payment we may extract from the agent when the reported type is θ^2. Because the goal is greater than $\frac{AB}{2}$, it follows that the payment the mechanism should extract from the agent when the reported type is θ^1 is at least $AB - \frac{A}{4}$. Because the maximum utility the agent can derive from the outcome function in this case is AB, it follows that the agent's utility (including payments) for type θ^1 can be at most $\frac{A}{4}$. Now consider the set $S = \{j \in \{1, \ldots, m\} : o^j(\theta^2) = t\}$. Then, if the agent falsely reports type θ^2 when the true type is θ^1, the utility of doing so (including payments) is at least $\sum_{j \in S} c_j A - CA - \frac{A}{4}$. This is to be at most the agent's utility for reporting truthfully in this case, which is at most $\frac{A}{4}$. It follows that $\sum_{j \in S} c_j A - CA - \frac{A}{4} \leq \frac{A}{4}$, which is equivalent to $\sum_{j \in S} c_j \leq C + \frac{1}{2}$. Because the c_j and C are integers, this implies $\sum_{j \in S} c_j \leq C$. Finally, because we need to extract at least a payment of V from the agent when type θ^2 is reported, but the utility that the agent gets from the outcome function in this case is at most $\sum_{j \in S} v_j$ and we can extract at most this by the IR constraint, it follows that $\sum_{j \in S} v_j \geq V$. Thus, S is a solution to the KNAPSACK instance. ■

5 Pseudopolynomial algorithm for one agent

In this section we develop a pseudopolynomial algorithm that shows that the first two multi-issue AMD problems discussed in the previous section are only *weakly* NP-complete. (A problem is only weakly NP-complete if it is NP-complete, but there exists an algorithm that would be polynomial if the numbers in the instance were given in unary, rather than binary—a *pseudopolynomial* algorithm.) This algorithm only works when there is only one type-reporting agent. While this is still a significant problem because of the conflict of interest between the designer and the agent, it is an interesting open problem to see if the algorithm can be generalized to settings with multiple agents.

Theorem 4 *Multi-issue automated deterministic mechanism design in the setting where there is only one agent, there is no interest in the payments ($a = 0$), and the number of types is a constant, can be solved in pseudopolynomial time using dynamic programming. This holds both with and without payments.*

Proof: The dynamic program adds in the issues one at a time. For each $k \in \{0, 1, \ldots, r\}$, it builds a matrix which concerns a reduced version of the problem instance where only the first k issues are included. Let $r(\theta^i, \theta^j) = u(\theta^i, o(\theta^j)) - u(\theta^i, o(\theta^i))$, that is, the regret that the agent has for reporting its true type θ^i rather than submitting the false report θ^j. (These regrets may be negative.) Any outcome function mapping the reported types to outcomes defines a vector of $|\Theta|(|\Theta| - 1)$ such regrets, one for each pair (θ^i, θ^j). Then, our matrix for the first k issues contains, for each possible regret vector v, a number indicating the highest expected value of the objective function that can be obtained with an outcome function over the first k issues whose regret vector is dominated by v. (One vector is said to be dominated by another if all entries of the former are less than or equal to the corresponding ones of the latter.) This entry is denoted $M^k[v]$. We observe that if v_1 dominates v_2, then $M^k[v_1] \geq M^k[v_2]$. If the absolute value of the regret between any two types is bounded by R, it suffices to consider $(2R+1)^{|\Theta|(|\Theta|-1)}$ regret vectors (each entry taking on values in $\{-R, -R+1, \ldots, 0, \ldots, R-1, R\}$). The matrix for $k = 0$ (i.e., when no issues have yet been added) is 0 everywhere. We then successively build up the next matrix as follows. When we add in issue k, there are $|O^k|^{|\Theta|}$ possibilities for setting the outcome function o^k from types to elements of O^k. Denoting a possible setting of o^k by a vector $w = (o^k(\theta^1), o^k(\theta^2), \ldots, o^k(\theta^{|\Theta|}))$, letting $g^k(w) = \sum_{\theta \in \Theta} g^k(\theta, o^k(\theta))$ be the total value gained in the objective function as a result of this vector, and letting $r(w) = (u^k(\theta^i, o^k(\theta^j)) - u^k(\theta^i, o^k(\theta^i))_{\{\theta^i \neq \theta^j\}}$ be the regret vector over this issue alone, we have the following recursive identity for $k > 0$: $M^k[v] = \max_w \{g^k(w) + M^{k-1}[v - r(w)]\}$. It is possible that, when we use this identity to fill in the matrices, the identity refers to an entry ”outside” the previous matrix, that is, one of the entries of $v - r(w)$ has absolute value greater than R. If this occurs, one of the following two cases applies:

- One of the entries is greater than R. This means that the regret allowed for one of the pairs (θ^i, θ^j) is greater than the maximum it could be. We may reduce the value of this entry to R, without causing a reduction in the highest value of the objective function that can be obtained.
- One of the entries is smaller than $-R$. This means that the regret allowed for one of the pairs (θ^i, θ^j) is smaller than the minimum it could be. Hence, it is impossible to construct an outcome function that satisfies this, and hence we simply say $M^{k-1}[v - r(w)] = -\infty$.

Once we have constructed M^r, we can use this matrix to solve any of our deterministic automated mechanism design problems. If payments are not allowed, we simply look at the entry $M^r[(0, 0, \dots, 0)]$, because this is the highest possible expected value of the objective function that we can obtain without the agent having positive regret anywhere. If payments are allowed, then we look at all the entries $M^r[v]$ where the regret vector v is such that we can set the payments so as to make every regret disappear—that is, where we can set π_θ such that for any θ^i, θ^j, we have $r(\theta^i, \theta^j) + \pi(\theta^j) - \pi(\theta^i) \leq 0$. (This is a simple linear program and can hence be solved in polynomial time.) Of all these entries, we choose the one with the highest value of the objective function. If we want to know not only the highest possible expected value of the objective function, but also a mechanism that achieves it, we need to store at each step not only the highest possible expected value for each matrix entry, but also a partial outcome function that achieves it. ■

6 Polynomial algorithm for designing an optimal randomized mechanism

Allowing randomization in the mechanism itself can never hurt the designer’s objective, because the set of deterministic mechanisms is a strict subset of randomized mechanisms. Interestingly, when we allow randomization in the mechanism, an optimal mechanism can be designed in polynomial time in the length of the concise representation. (This is already known for the single-issue problem [3, 4, 5], but, as argued earlier, this does not directly imply that this is also the case in the multi-issue problem, because the problem does not decompose over issues.)

Theorem 5 *Multi-issue automated randomized mechanism design can be solved in polynomial time using linear programming, for any constant number of agents, any IR notion, any IC notion, any objective function g and any weight, a, on payments collected. This holds even for a growing number of types, with and without payments.*

Proof: We will solve this by casting it as a linear program. One method of solving this problem is to construct a linear program with, for each type report vector, for each outcome, a variable representing the probability that this outcome will be chosen (in addition to variables representing the payments). While this technique is effective for flatly represented outcome spaces [3, 4, 5], here it is not because the outcome space is exponential in the size of the representation. However, we can reduce the number of variables to a polynomial number. To this end, we observe:

- For all i, $E(u_i|(\hat{\theta_1}, \dots, \hat{\theta_N}), \theta_i) = \sum_{(o^1,\dots,o^r)\in O} P((o^1, \dots, o^r)|(\hat{\theta_1}, \dots, \hat{\theta_N})) \sum_{1\leq k\leq r} u_i^k(\theta_i, o^k) = \sum_{1\leq k\leq r} \sum_{(o^1,\dots,o^r)\in O} P((o^1, \dots, o^r)|(\hat{\theta_1}, \dots, \hat{\theta_N}))u_i^k(\theta_i, o^k) =$ $\sum_{1\leq k\leq r} \sum_{o_*^k\in O^k} u_i^k(\theta_i, o_*^k) \sum_{(o^1,\dots,o^r):o^k=o_*^k} P((o^1, \dots, o^r)|(\hat{\theta_1}, \dots, \hat{\theta_N}))$ $= \sum_{1\leq k\leq r} \sum_{o_*^k\in O^k} P(o^k = o_*^k|(\hat{\theta_1}, \dots, \hat{\theta_N}))u_i^k(\theta_i, o_*^k)$.
- Similarly, $E(g|(\hat{\theta_1}, \dots, \hat{\theta_N})) = \sum_{1\leq k\leq r} \sum_{o_*^k\in O^k} P(o^k = o_*^k|(\hat{\theta_1}, \dots, \hat{\theta_N}))g^k((\hat{\theta_1}, \dots, \hat{\theta_N}), o_*^k)$.

It follows that for the purposes at hand, we care only about the quantities $P(o^k = o_*^k|(\hat{\theta_1}, \dots, \hat{\theta_N}))$, rather than about the entire distribution. There are precisely $\sum_{1\leq k\leq r} |O^k| \prod_{1\leq i\leq N} |\Theta^i|$ such probabilities, which is a polynomial number when the number of agents, N, is a constant. Additionally, only $N \prod_{1\leq i\leq N} |\Theta^i|$ variables are needed to represent the payments made by the agents in each case (or none if payments are not possible).

The linear program, which contains constraints for the IC notions and IR notion in question, and attempts to optimize some linear function of the expected value of the objective function and payments made, is now straightforward to construct. (We do not give it explicitly due to limited space, but we have given the analogous linear program for the flat representation [3, 4, 5].) Because linear programs can be solved in polynomial time, and the number of variables and equations in our program is polynomial for any constant number of agents, the problem is in P. ■

7 Conclusions

Previous research on automated mechanism design assumed that the outcome space was flatly represented, which makes that work inapplicable if the outcome space is exponential, as it is, for example, in combinatorial auctions. This paper introduced (to our knowledge) the first more concise representation for the problem, which relies on decomposing the outcome space into distinct *issues*. While the decomposability makes the input to the design problem polynomial, we showed that it is NP-complete (even with a single agent with only two types) to design a mechanism that maximizes one of the following objectives: 1) expected social welfare when payments are not possible, 2) a general objective function even when payments are possible, and 3) expected payments to the center (designer). We showed that the NP-hardness is only weak by developing a pseudopolynomial algorithm for the former two single-agent mechanism design problems with any constant number of types. Finally, we showed that when designing randomized mechanisms, we can exploit the structure of the representation and solve the general problem in polynomial time for any constant number of agents for any growing number of types.[5]

[5]The pseudopolynomial algorithm and the algorithm for designing randomized mechanisms can be trivially extended to designing optimal ϵ-*incentive compatible* mechanisms as well, i.e., where the agents' regrets, defined above, can go up to ϵ.

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 *The 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 103–110, Edmonton, Canada, 2002.

[4] Vincent Conitzer and Tuomas Sandholm. Automated mechanism design: Complexity results stemming from the single-agent setting. In *The 5th International Conference on Electronic Commerce (ICEC-03)*, pages 17–24, Pittsburgh, PA, USA, 2003.

[5] Vincent Conitzer and Tuomas Sandholm. Self-interested automated mechanism design and implications for optimal combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, New York, NY, 2004.

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

[7] Eithan Ephrati and Jeffrey S Rosenschein. The Clarke tax as a consensus mechanism among automated agents. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 173–178, 1991.

[8] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *The Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 548–553, Stockholm, Sweden, August 1999.

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

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

[11] Holger Hoos and Craig Boutilier. Bidding languages for combinatorial auctions. In *The Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1211–1217, Seattle, WA, 2001.

[12] Andreu Mas-Colell, Michael Whinston, and Jerry R. Green. *Microeconomic Theory*. 1995.

[13] 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. 1989.

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

[15] Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 1–12, 2000.

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

[17] Ryan Porter, Yoav Shoham, and Moshe Tennenholtz. Fair imposition. *Journal of Economic Theory*, 2004. To appear. Early version appeared in IJCAI-01.

[18] M. Rothkopf, A. Pekeč, and R. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[19] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, January 2002.

[20] 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.

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