An Algorithm for Automatically Designing Deterministic Mechanisms without Payments[*]

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

Abstract

Mechanism design is the art of designing the rules of the game so that a desirable outcome is reached even though the agents in the game behave selfishly. This is a difficult problem because the designer is uncertain about the agents' preferences and the agents may lie about their preferences. Traditionally, the focus in mechanism design has been on designing mechanisms that are appropriate for a range of settings. While this approach has produced a number of famous mechanisms, much of the space of possible settings is still left uncovered. In contrast, in automated mechanism design (AMD)*, a mechanism is* computed *on the fly for the setting at hand—a universally applicable approach.*

In this paper we present (to our knowledge) the first algorithm *designed specifically for AMD. It is designed for the special case where there is only one type-reporting agent, the mechanism must be deterministic, and payments are not possible. The algorithm relies on an association of a particular (easy to compute) mechanism to each subset of outcomes, and a proof that one such mechanism is an optimal one—which allows us to reduce the search space to one of size $2^{|O|}$. We propose an admissible heuristic to use in searching over this space, and show how it can be updated efficiently from node to node. We show how to apply branch and bound DFS as well as IDA* to this search space, and show that this approach outperforms CPLEX 8.0, a general-purpose solver, solidly on unstructured instances, both with and without an IR constraint. However, on our third type of instance, a bartering problem, CPLEX almost achieves the performance of our algorithm by exploiting the structure inherent in the domain.*

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

1. Introduction

In many multiagent settings, an *outcome* must be chosen on the basis of the preferences reported by a group of one or more 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 her 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.

Mechanism design is the art of designing the rules of the game so that a desirable outcome is reached even though the agents in the game behave selfishly. Traditionally, the focus in mechanism design has been on designing mechanisms that are appropriate for a range of settings. While this approach has produced a number of famous mechanisms (for example, the VCG mechanism [15, 3, 9], the dAGVA mechanism [7, 1], and the Myerson optimal auction [11]), much of the space of possible settings is still left uncovered. In many (arguably most) cases where a mechanism is needed, the classical mechanisms are not satisfactory because they make assumptions on what the agents can do (for example, side payments are required for the mechanism to work); they pursue the wrong objective (for example, social welfare is pursued instead of maximal revenue); or they do not make use of all available information (such as probability distributions over the agents' preferences, or *types*), at the cost of the objective pursued.

In contrast, in an approach we call *automated mechanism design (AMD)* [4, 5, 6], a mechanism is *computed* on the fly for the setting at hand—a universally applicable approach. In our previous work on automated mechanism design, we studied the worst-case theoretical complexity of some versions of the abstract problem. In this paper we present (to our knowledge) the first *algorithm* designed specifically for

AMD. It is not designed for the general case of AMD, but rather only for the special case where there is only one type-reporting agent, the mechanism must be deterministic, and payments are not possible. While this is a restricted setting, it is nevertheless one of significant importance, both theoretically and practically, for the following reasons:

• Good (polynomial time) algorithms for designing randomized mechanisms are available (using linear programming), but not for deterministic mechanisms: designing deterministic mechanisms is typically NP-complete.

• Much of classical mechanism design has focused on settings where payments are possible, so many mechanisms are available for such settings; this is not the case for settings where payments are not possible.

• In the single type-reporting agent setting, all incentive-compatibility and individual-rationality notions coincide. Thus, any results obtained in this setting apply to all combinations of these notions, rather than just to a particular combination, making these results particularly fundamental for AMD.

• All NP-hardness results for deterministic AMD known to date hold even in the single-agent setting, indicating that at least part, and perhaps all, of any hardness result in AMD derives from the fact that the single-agent setting is a special case of the problem. In other words, good algorithms for solving the single-agent case are likely to help us significantly in more general cases.

• Many important mechanism design problems fit this setting exactly. One such problem that we will study is creating a one-on-one bartering mechanism.[1]

Of course, we anticipate that future research will create special-purpose algorithms for cases of AMD not covered by this algorithm. Special-purpose optimization algorithms have been tremendously successful in other multiagent settings in the intersection of AI and economics. Most notably, *combinatorial auctions* [13, 14, 8, 12] have been a topic of plentiful and valuable research in this area. We believe there is a similar potential for such research on automated mechanism design.

2. Automated mechanism design

We first discuss automated mechanism design in the general case informally; then we provide a formal definition for the case where there is only one type-reporting agent and the mechanism must be deterministic, without payments—the setting where our algorithm applies.

1 Mechanism design with only one reporting agent has also been widely studied in decision analysis. For example, how should one motivate a meteorologist to state exactly his subjective probability that it will rain tomorrow (rather than to under- or overstate it)? [17]

2.1. The general setting

In the general case, an automated mechanism design problem is given by the following.

• A finite set of outcomes O.

• A finite set of N agents.

• For each agent i, 1. a finite set of types Θ_i; 2. a probability distribution p_i over Θ_i (in the case of correlated types, there is a single joint distribution p over $\Theta_1 \times \ldots \times \Theta_N$); 3. a utility function $u_i : \Theta_i \times O \to \mathbb{R}$.[2]

• An objective function g whose expectation the designer wishes to maximize. Possible objective functions include social welfare (the sum of all agents' utilities), total payments extracted from the agents, the designer's own agenda for the outcome, and any mix of these.

• A specification of the tools available to the designer—that is, what types of mechanisms are allowed. For instance, are payments possible? Is randomization possible?

The mechanism designer has to construct a game for the agents to play; how this game is played will determine the outcome chosen. (Additionally, it determines any side payments.) In designing the game, the mechanism designer seeks to maximize the expected value of the objective g, under the assumption that the agents will play the game strategically. A useful result called the *revelation principle* states that the mechanism designer can restrict his attention to *truthful direct revelation mechanisms*, where the agents report their types directly and where they never have any incentive to report them falsely. Thus, a *deterministic* mechanism is given by a function from reported type vectors to outcomes (and possibly to payment vectors). A *randomized* mechanism is given by a function from reported type vectors to probability distributions over outcomes (and possibly to payment vectors—but the agents will only care about the expected payment they have to make as long as they are risk-neutral.)

Furthermore, we need a definition of a *truthful* mechanism (one in which agents do not have incentives to lie about their types). The corresponding constraint is known as an *incentive compatibility (IC)* constraint. The two best-known such constraints are the following. In *implementation in dominant strategies (DS)*, an agent never has an incentive to misreport her type even if she knows what all the other agents reported. In *implementation in Bayes-Nash equilibrium (BNE)*, an agent never has an incentive to misreport her type presuming that she knows nothing more about

2 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.

the other agents' types than the commonly known prior, and that she believes that all other agents will report truthfully.

Finally, in many cases, the designer needs to make sure that the agents do not incur a loss as a result of participating in the mechanism (because the agent may then choose not to participate). This is known as an *individual rationality (IR)* constraint. Two well-known notions are *ex interim* IR, where it always makes sense for the agent to participate if she knows her own type but only the priors for the other agents; and *ex post* IR, where it always makes sense for the agent to participate even if she knows everyone's type.

2.2. Automatically designing deterministic mechanisms without payments with one type-reporting agent

The definitions from the previous subsection simplify significantly when applied to the setting where a deterministic mechanism without payments must be designed, with a single type-reporting agent. For one, the different possible IC (truthfulness) constraints differ only in what a type-reporting agent is assumed to know about other type-reporting agents' preferences and reports. Because in this setting, there are no other type-reporting agents, the different IC constraints coincide. The same is true for the IR (participation) constraints. We also do not need distributions over outcomes, or payment functions. The result is the following formal definition for our special case.

(In a slight abuse of notation, we use the variable o (possibly with subscripts) both for elements of the outcome set O and for functions from types to outcomes ($o : \Theta \to O$); it should be clear from context what is meant.)

Definition 1 (AMD) *We are given a set of outcomes O, and a set of types Θ for the agent together with a probability distribution p over these types. Additionally we are given a utility function for the agent, $u : \Theta \times O \to \mathbb{R}$, and an objective function for the designer, $g : \Theta \times O \to \mathbb{R}$. We are asked to find an outcome function $o : \Theta \to O$ (a deterministic mechanism without payments) such that:*

1. *For every $\theta, \hat{\theta} \in \Theta$, $u(\theta, o(\theta)) \geq u(\theta, o(\hat{\theta}))$ (the IC constraint).*
2. *If there is an IR constraint, for every $\theta \in \Theta$, $u(\theta, o(\theta)) \geq 0$. (In this case there typically also is a* default outcome *o_0 with $u(\theta, o_0) = 0$ for all $\theta \in \Theta$.[3])*
3. *Subject to the previous constraints, the mechanism maximizes $\sum_{\theta \in \Theta} p(\theta) g(\theta, o(\theta))$.*

3 We can set the utility of the default outcome to 0 without loss of generality, by normalizing the utility function. (From a decision-theoretic point of view it does not matter how utilities compare across types, because the agent always knows her own type and will not take utilities for other types into account in making any decision.)

In our previous work on AMD, we have already shown that this problem is NP-complete by a reduction from MINSAT (even without the IR constraint, and even when the objective function is a social welfare function including another agent outside of the mechanism).

3. Application: one-on-one bartering

As an interlude, we first present an application. Consider the situation where two agents each have an initial endowment of goods. Each agent has a valuation for every subset of the n goods that the agents have together. It is possible that both agents can become better off as a result of trade. Suppose, however, that the agents cannot make any form of payment; all they can do is swap goods. This is known as *bartering*. Additionally, suppose that one agent (agent 1) is in the position of dictating the rules of the bartering process. Agent 1 can credibly say to agent 2, "we will barter by my rules, or not at all". This places agent 1 in the position of the mechanism designer, and corresponds to the following AMD problem. The set of outcomes is the set of all allocations of the goods (there are 2^n of them). Agent 2 is to report his preferences over the goods (the valuation that agent has for each subset), and on the basis of this report an outcome is chosen. This outcome function, which is selected by agent 1 beforehand, must be incentive compatible so that agent 2 has no incentive to misreport. Also, it must be individually rational, or agent 2 simply will not trade.[4] Under these constraints, agent 1 wishes to make the expected value of her own allocation under the mechanism as large as possible. The revelation principle justifies that restricting agent 1 to this approach comes at no loss to that agent.

Automatically generated mechanisms for this setting are likely to be useful in barter-based electronic marketplaces, such as Recipco, firstbarter.com, BarterOne, and Intagio.

We now return to computational aspects, but we will readdress the bartering problem in our experiments. We will postpone dealing with IR constraints for a few sections, and then return to it.

4. Search over subsets of outcomes

In this section, we associate with each subset of outcomes a truthful mechanism for that set of outcomes; we then show that for some subset of outcomes, the truthful mechanism associated with that subset of outcomes is an optimal mechanism for the setting. Because the mechanism associated with a subset of outcomes is easy to compute,

4 If agent 1 actually wants to make the rules so that there is no trade for a certain type report, she can simply make the original allocation the outcome for this type report; so there is no loss to agent 1 in designing the outcome function in such a way that agent 2 always wishes to participate in the mechanism.

we can search over subsets of outcomes (of which there are $2^{|O|}$) rather than over all possible outcome functions (of which there are $|O|^{|\Theta|}$).[5]

We first define the outcome function (mechanism) o_X associated with a particular subset of the outcomes.

Definition 2 *For a given subset $X \subseteq O$, let $o_X(\theta)$ be (the lowest-indexed element of) $\arg\max_{\{o \in X : (\forall o' \in X) u(\theta,o) \geq u(\theta,o')\}} g(\theta, o)$. Let $v(X)$ be given by $\sum_{\theta \in \Theta} p(\theta) g(\theta, o_X(\theta))$.*

Intuitively, $o_X(\theta)$ is the outcome we wish to pick for type θ, if we (somehow) know that the set of other outcomes used in the mechanism is exactly X, and we wish to pick an outcome from X as well. $v(X)$ is the expected value of the objective function for the mechanism o_X, presuming that the agent reports truthfully. The next lemma shows that indeed, the agent has no incentive to report falsely.

Lemma 1 *For all $X \subseteq O$, o_X is truthful. (Thus $v(X)$ is indeed the expected value of the objective function for it.)*

Proof: For any pair of types θ_1, θ_2, we have that $o_X(\theta_2) \in X$ because all outcomes ever chosen by o_X are in X; and thus that $u(\theta_1, o_X(\theta_1)) \geq u(\theta_1, o_X(\theta_2))$, because for any θ, $o_X(\theta)$ maximizes $u(\theta, \cdot)$ among outcomes $o \in X$. ■

The next lemma shows that for any subset X, the mechanism o_X dominates all mechanisms that use exactly the outcomes in X.

Lemma 2 *For any $X \subseteq O$, suppose that $o : \Theta \rightarrow X$ is a truthful mechanism making use only of outcomes in X, but using each outcome in X at least once—that is, $o(\Theta) = X$. Let its expected value of the objective function be $v_o = \sum_{\theta \in \Theta} p(\theta) g(\theta, o(\theta))$. Then $v(X) \geq v_o$.*

Proof: For any $\theta \in \Theta$, we must have that for any $o \in X$, $u(\theta, o(\theta)) \geq u(\theta, o)$—because there exists some $\theta' \in \Theta$ such that $o(\theta') = o$, and thus the agent can guarantee herself at least utility $u(\theta, o)$ by reporting θ'. But $o_X(\theta)$ maximizes $g(\theta, \cdot)$ among such outcomes. Thus, $g(\theta, o_X(\theta)) \geq g(\theta, o(\theta))$. It follows that $v(X) = \sum_{\theta \in \Theta} p(\theta) g(\theta, o_X(\theta)) \geq \sum_{\theta \in \Theta} p(\theta) g(\theta, o(\theta)) = v_o$. ■

It is not necessarily the case that $v(X) = v_o$ for some truthful o making use of all outcomes in X; for instance, there could be some outcome in X that has both a very low utility value and a very low objective value. Then o_X will not use this outcome, and thereby have a higher expected value of the objective function than any mechanism that does use it.

We are now ready to present the main theorem of this section, which states that the best o_X is indeed an optimal mechanism.

Theorem 1 $\max_{X \subseteq O} v(X)$ *is the maximum expected value of the objective over* all *mechanisms (that are deterministic and use no payments). o_X is an optimal mechanism (among mechanisms that are deterministic and use no payments) if $X \in \arg\max_{X \subseteq O} v(X)$.*

Proof: Consider an optimal truthful mechanism o,[6] and let X be the set of all outcomes it uses ($X = o(\Theta)$). By Lemma 1, o_X is truthful and $v(X)$ is the expected value of the objective function for it. By Lemma 2, we have $v(X) \geq v_o$ where v_o is the expected value of the objective function for o. ■

5. A heuristic and its admissibility

We now proceed to define an outcome function that is associated with two disjoint subsets X and Y of the outcomes; we will use this outcome function to compute an admissible heuristic for our search problem. The interpretation is as follows. In the process of constructing a mechanism of the kind described in the previous section, we successively label each outcome as "in" or "out", depending on whether we wish to include this outcome in the set that the eventual mechanism is associated with. X consists of the outcomes that we have already decided are "in"; Y consists of the outcomes that we have already decided are "out". To get an optimistic view of the mechanisms we may eventually arrive at from here, we assign to each type the outcome in $O - Y$ that gives us the highest objective value for that type (the mechanisms certainly will not use any outcome in Y), under the constraint that this outcome will make that type at least as well off as any outcome in X (because we have already decided that these are certainly "in", so we know this constraint must apply).

Definition 3 *For given subsets $X, Y \subseteq O$, let $o_{X,Y}(\theta)$ be (the lowest-indexed element of) $\arg\max_{o \in O-Y : (\forall o' \in X) u(\theta,o) \geq u(\theta,o')} g(\theta, o)$. Let $v(X, Y)$ be given by $\sum_{\theta \in \Theta} p(\theta) g(\theta, o_{X,Y}(\theta))$*

Outcome functions of this type do *not* necessarily constitute truthful mechanisms. (For instance, if X and Y are both the empty set, then $o_{X,Y}$ will simply choose the objective-maximizing outcome for each type.) Nevertheless, because we are merely trying to obtain an optimistic estimate, we

5 In the case where $|O|$ is bigger than $|\Theta|$, we can restrict ourselves to outcome subsets of size at most $|\Theta|$, making our approach still more efficient than the straightforward brute search approach. For simplicity of presentation, in this paper we will focus on settings where $|\Theta| > |O|$ (as is commonly the case).

6 Which, by the revelation principle, is an optimal mechanism, period.

compute $v(X,Y)$ as before, presuming the agents will report truthfully. The following theorem shows that $v(X,Y)$ is indeed admissible.

Theorem 2 *For any subsets $X, Y \subseteq O$, for any $Z \subseteq O - X - Y$, for any $\theta \in \Theta$, we have $g(\theta, o_{X,Y}(\theta)) \geq g(\theta, o_{X \cup Z}(\theta))$; and $v(X,Y) \geq v(X \cup Z)$.*

Proof: Using the facts that $X \subseteq X \cup Z$ and $X \cup Z \subseteq O - Y$, we can conclude that $\{o \in X \cup Z : (\forall o' \in X \cup Z) u(\theta,o) \geq u(\theta,o')\} \subseteq \{o \in X \cup Z : (\forall o' \in X) u(\theta,o) \geq u(\theta,o')\} \subseteq \{o \in O - Y : (\forall o' \in X) u(\theta,o) \geq u(\theta,o)\}$. It follows that $g(\theta, o_{X,Y}(\theta)) = \max_{o \in O-Y:(\forall o' \in X) u(\theta,o) \geq u(\theta,o')} g(\theta,o) \geq \max_{o \in X \cup Z:(\forall o' \in X \cup Z) u(\theta,o) \geq u(\theta,o')} g(\theta,o) = g(\theta, o_{X \cup Z}(\theta))$. Thus $v(X,Y) = \sum\limits_{\theta \in \Theta} p(\theta) g(\theta, o_{X,Y}(\theta)) \geq \sum\limits_{\theta \in \Theta} p(\theta) g(\theta, o_{X \cup Z}(\theta)) = v(X \cup Z)$. ■

The following theorem shows that conveniently, at a leaf node, where we have decided for every outcome whether it is in or out, the heuristic value coincides with the value of that outcome set.

Theorem 3 *For any $X \subseteq O, \theta \in \Theta$, we have $o_{X,O-X}(\theta) = o_X(\theta)$ and $v(X, O-X) = v(X)$.*

Proof: Immediate using $O - (O - X) = X$. ■

6. The algorithm

6.1. Basic structure

We now summarize the backbone of our algorithm. A node in our search space is defined by a set of outcomes that are definitely "in" (X)[7] and a set of outcomes that are definitely out (Y). For a node at depth n, $X \cup Y$ always constitutes the first n outcomes for some fixed order of the outcomes; thus, a node has two children, one where the next outcome is added to X, and one where it is added to Y. The expansion order is fixed at putting the node in X first, and then in Y. The heuristic value (bound) of a node is given by $v(X,Y)$, as described above.

We can now simply apply A*; this, however, quickly fills up the available memory, so we resort to more space-efficient methods. We first present branch and bound depth-first search for our setting, and then IDA*.

In the following, v is the heuristic for the current node. d is the depth of the current node. n (a global variable) is the number of outcomes. CB (another global) is the outcome set corresponding to the best mechanism found so far. L (another global) is the expected value of the objective function for the best mechanism we have found so far. o_i is outcome i. The other variables are as described above.

```
SEARCH1($X$, $Y$, $v$, $d$)
 **if** $d = n + 1$
  $CB = X$
  $L = v$
 **else**
  **if** $v(X \cup \{o_d\}, Y) > L$
   SEARCH1($X \cup \{o_d\}$, $Y$, $v(X \cup \{o_d\}, Y)$, $d + 1$)
  **if** $v(X, Y \cup \{o_d\}) > L$
   SEARCH1($X$, $Y \cup \{o_d\}$, $v(X, Y \cup \{o_d\})$, $d + 1$)

BRANCH-AND-BOUND-DFS()
 $CB$ := *NULL*
 $L$ := $-\infty$
 SEARCH1({}, {}, 0, 1)
 **return** $CB$
```

Our implementation of IDA* is similar, except we do not initialize L to $-\infty$. Rather, we initialize it to some high value, and decrease it every time we fail to find a solution—either to a fraction of itself, or to the highest value that is still feasible (whichever is *less*). This also requires us to keep track of the highest value still feasible (given by HF, another global variable), so that we have to modify the search call slightly.

```
SEARCH2($X$, $Y$, $v$, $d$)
 **if** $d = n + 1$
  $CB = X$
  $L = v$
 **else**
  **if** $v(X \cup \{o_d\}, Y) > L$
   SEARCH2($X \cup \{o_d\}$, $Y$, $v(X \cup \{o_d\}, Y)$, $d + 1$)
  **else if** $v(X \cup \{o_d\}, Y) > HF$
   $HF$ := $v(X \cup \{o_d\}, Y)$
  **if** $v(X, Y \cup \{o_d\}) > L$
   SEARCH2($X$, $Y \cup \{o_d\}$, $v(X, Y \cup \{o_d\})$, $d + 1$)
  **else if** $v(X, Y \cup \{o_d\}) > HF$
   $HF$ := $v(X, Y \cup \{o_d\})$

IDA*()
 $CB$ := *NULL*
 $L$ := *initial-limit*
 **while** $CB$ = *NULL*
  $HF$ := $-\infty$
  SEARCH2({}, {}, 0, 1)
  $L$ := $\min\{HF$, *fraction*$\cdot L\}$
 **return** $CB$
```

7 We emphasize that this does *not* mean that the outcome will definitely be used by the mechanism corresponding to any descendant leaf node; rather, this outcome *may* be used by any descendant leaf node; and for any descendant leaf node, in the mechanism associated with this node, any type must receive an outcome at least as good to it as this one.

6.2. Efficiently updating the heuristic

Rather than computing the heuristic anew each time, it can be computed much more quickly from information used for computing the heuristic at the parent node. For instance, when adding an outcome o to X, we will not have to change $o_{X,Y}(\theta)$ unless $u(\theta, o) > u(\theta, o_{X,Y}(\theta))$. As another example, when adding an outcome o to Y, we will not have to change $o_{X,Y}(\theta)$ unless $o_{X,Y}(\theta) = o$. In addition to this, maintaining appropriate data structures (such as a list of the outcomes sorted by objective value for a given type) allows us to quickly find the new outcome when we do need to make a change.

7. Individual rationality

We now show how to deal with an individual rationality constraint in this setting.

Theorem 4 *o_X is individually rational if and only if for every $\theta \in \Theta$, there is some $o \in X$ such that $u(\theta, o) \geq 0$.*

Proof: If for some $\theta \in \Theta$, there is no $o \in X$ such that $u(\theta, o) \geq 0$, o_X cannot give the agent nonnegative utility for type θ because o_X uses only outcomes from X; so it is not individually rational. On the other hand, if for every $\theta \in \Theta$, there is some $o \in X$ such that $u(\theta, o) \geq 0$, then o_X will give the agent nonnegative utility for that type θ, because o_X is constrained to choose an outcome that maximizes $u(\theta, \cdot)$ among outcomes from X, and at least one of the outcomes in X gives nonnegative utility. So it is individually rational. ■

It follows that when we have an individual rationality constraint, in our search procedures, we do not need to expand nodes where for some type θ, there are no outcomes left in $O - Y$ that give the agent a nonnegative utility for θ.

8. Experimental results

In this section, we compare the performances of branch and bound DFS and IDA* over our search space with the performance of CPLEX 8.0,[8] on random instances drawn from three different distributions. In each case, we investigate both scalability in the number of types and in the number of outcomes.

8 CPLEX is a state of the art general-purpose solver that typically performs very well even on special purpose domains—for instance, its performance in combinatorial auction clearing is at least competitive with many algorithms designed specifically for this purpose. It is almost a *de facto* standard as the first cut at solving optimization problems. To apply CPLEX to our problem, we use the mixed integer program formulation obtained by taking the standard LP formulation for automatically designing randomized mechanisms [4, 5, 6], and forcing all probabilities to be 0 or 1.

8.1. Uniform distribution, no IR

For this distribution, each value $u(\theta, o)$ and each value $g(\theta, o)$ is independently and uniformly drawn from $[0, 100]$. No IR constraint applies (all utilities are nonnegative).

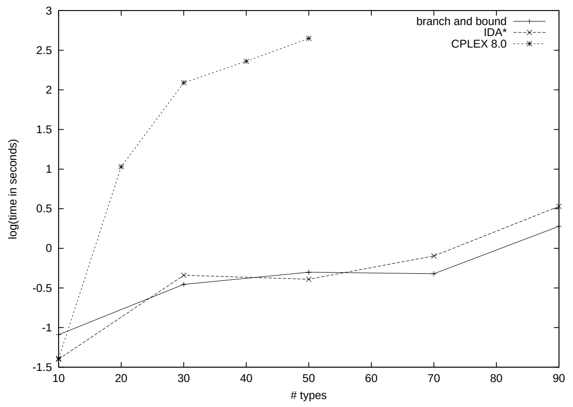

Figure 1. Performance vs. types for the uniform, no IR case with 20 outcomes

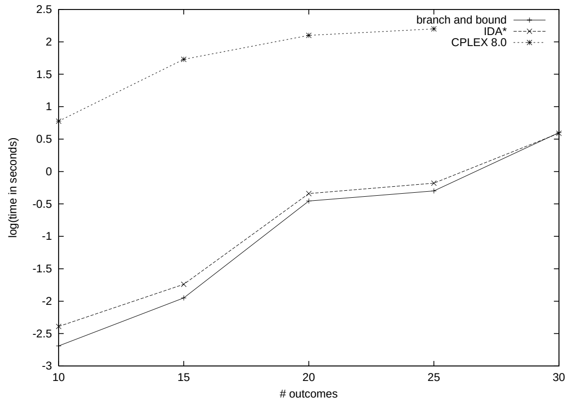

Figure 2. Performance vs. outcomes for the uniform, no IR case with 30 types

Both versions of our algorithm outperform CPLEX soundly; our approach is especially more scalable in the number of types.

8.2. Uniform distribution, with IR

Now, each value $u(\theta, o)$ and each value $g(\theta, o)$ is independently and uniformly drawn from $[-50, 50]$. We apply an IR constraint (the agent can never get negative utility).

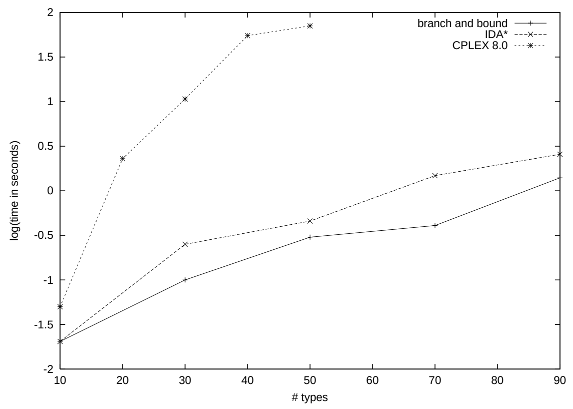

Figure 3. Performance vs. types for the uniform, with IR case with 20 outcomes

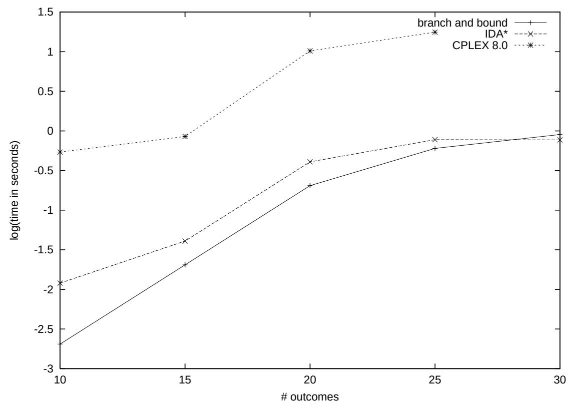

Figure 4. Performance vs. outcomes for the uniform, with IR case with 30 types

Both versions of our algorithm still solidly outperform CPLEX, but the gaps are a little tighter; CPLEX manages to get a greater speedup factor out of the IR constraint.

8.3. Bartering

The final distribution corresponds to the bartering problem described earlier. The designer and the agent each have k goods (for 2^{2k} outcomes—each good can end up with either agent); the designer has a randomly drawn value (from $[0, 10]$) for each individual good (constituting g, which does not depend on θ in this case), and the agent has a randomly drawn value (from $[0, 10]$) for each individual good for each type (constituting u). The value of a bundle to an agent is the sum of the values of the individual goods.[9] If the total number of goods is odd, the agent gets one more good than the designer.

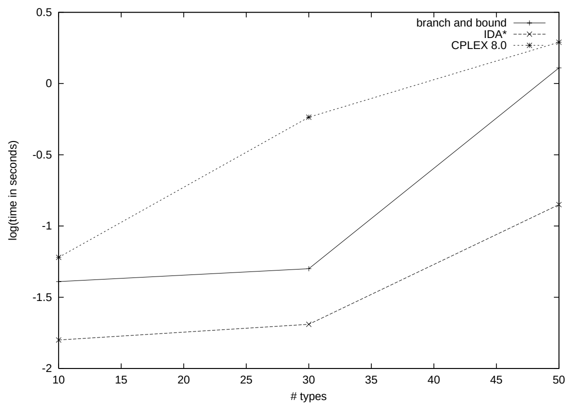

Figure 5. Performance vs. types for the bartering case with 32 outcomes

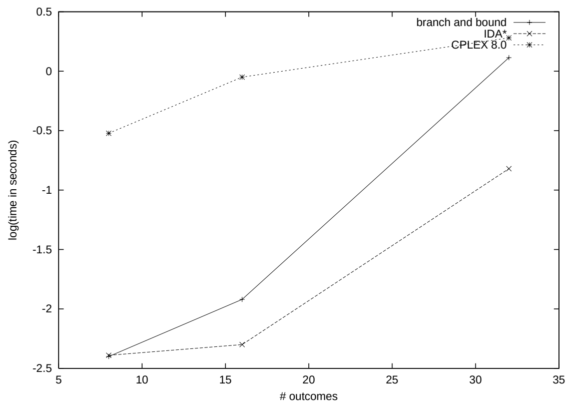

Figure 6. Performance vs. outcomes for the bartering case with 50 types

The gaps here are much tighter, and it appears that CPLEX may in fact get the upper hand on even larger instances. (Space limitations prevented us from taking the experiments further.) CPLEX apparently makes very good use

9 There is nothing preventing our approach from having more complicated values over bundles; we simply felt it was nice to present the simplest example.

of the additional structure in this domain, whereas our algorithm is not geared towards exploiting this structure. Also, IDA* seems to outperform branch and bound DFS now.

9. Conclusions

In this paper, we have proposed (to our knowledge) the first algorithm designed specifically for automated mechanism design (AMD). It solves AMD in the setting where there is only one type-reporting agent, the mechanism must be deterministic, and payments are not possible. This setting is important for theoretical reasons (the hardness in this setting is at the core of hardness in more general settings), as well as for practical reasons (there are few classical mechanisms, and no general poly-time algorithms, available for this setting; many important applications (such as bartering) fit this setting). The algorithm relied on an association of a particular (easy to compute) mechanism to each subset of outcomes, and a proof that one such mechanism is an optimal one—which allows us to reduce the search space to one of size $2^{|O|}$. We proposed an admissible heuristic to use in searching over this space, and showed how it can be updated efficiently from node to node. We showed how to apply branch and bound DFS as well as IDA* to this search space, and showed that this approach outperformed CPLEX 8.0, a general-purpose solver, solidly on unstructured instances, both with and without an IR constraint. However, on our third example, a bartering problem, CPLEX almost achieved the performance of our algorithm by exploiting the structure inherent in the domain.

10. Future research

The most immediate future research is to expand the techniques discussed here to other, possibly more general cases of automated mechanism design. The general case of automated mechanism design, with an arbitrary number of agents, any IC and any IR constraint, with or without payments and with or without randomization, can already be solved with a general purpose solver such as CPLEX, but there is likely to be room for improvement over CPLEX even in the general AMD case. Attempting to generalize the optimization techniques used here to the general AMD setting appears to be a good approach to this.

Of course, even algorithms that are applicable only to other special cases of AMD would be of great interest. One exciting application area of AMD for which better algorithms are needed is *optimal combinatorial auction design*, where the auctioneer designs the rules of a combinatorial auction to maximize expected revenue. This is a different problem than *clearing* combinatorial auctions, where the auctioneer attempts to maximize the sum of the values of the accepted bids; in this problem there is typically no discussion of why the agents are motivated to bid truthfullly (or it is assumed that a VCG payment scheme is used, which makes the agents bid truthfully—but this does not maximize expected revenue). Optimal combinatorial auction design is a well-recognized open problem [2, 16]. Other exciting application areas include public goods problems and dispute settlements. It would also be interesting to see if an algorithm can be tailored to the bartering problem proposed in this paper, as our algorithm apparently does not do as well as CPLEX in exploiting the specific structure of the bartering problem.

References

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

[2] C. Avery and T. Hendershott. Bundling and optimal auctions of multiple products. *Review of Economic Studies*, 67:483–497, 2000.

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

[4] V. Conitzer and T. Sandholm. Complexity of mechanism design. *UAI*, pages 103–110, Edmonton, Canada, 2002.

[5] V. Conitzer and T. Sandholm. Automated mechanism design: Complexity results stemming from the single-agent setting. In *Proceedings of the 5th International Conference on Electronic Commerce (ICEC-03)*, pages 17–24, 2003.

[6] V. Conitzer and T. Sandholm. Self-interested automated mechanism design and implications for optimal combinatorial auctions. *ACM-EC*, New York, NY, 2004.

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

[8] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. *IJCAI*, pages 548–553, Stockholm, Sweden, Aug. 1999.

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

[10] A. Mas-Colell, M. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

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

[12] N. Nisan. Bidding and allocation in combinatorial auctions. *ACM-EC*, pages 1–12, Minneapolis, MN, 2000.

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

[14] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54.

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

[16] R. V. Vohra. Research problems in combinatorial auctions. Mimeo, version Oct. 29, 2001.

[17] D. von Winterfeldt and W. Edwards. *Decision analysis and behavioral research*. Cambridge University Press, 1986.