# Automated Mechanism Design: A New Application Area for Search Algorithms*

Tuomas Sandholm

Computer Science Department
Carnegie Mellon University
Pittsburgh PA 15213
sandholm@cs.cmu.edu

**Abstract.** *Mechanism design* is the art of designing the rules of the game (aka. *mechanism*) so that a desirable outcome (according to a given objective) is reached despite the fact that each agent acts in his own self-interest. Examples include the design of auctions, voting protocols, and divorce settlement procedures. Mechanisms have traditionally been designed manually for classes of problems. In 2002, Conitzer and Sandholm introduced the *automated mechanism design* approach, where the mechanism is computationally created for the specific problem instance at hand. This approach has several advantages: 1) it can yield better mechanisms than the ones known to date, 2) it applies beyond the problem classes studied manually to date, 3) it can circumvent seminal economic impossibility results, and 4) it shifts the burden of design from man to machine. In this write-up I overview the approach, focusing on problem representations, computational complexity, and initial applications. I also lay out an agenda for future research in this area.

## 1 Introduction

In multiagent settings, agents generally have conflicting preferences, yet it is crucial to be able to aggregate the preferences, that is, to choose a socially desirable *outcome*, for example a president, resource allocation, or task allocation. This problem prevails among any self-interested agents: humans, companies, *etc.*—and software agents representing such parties.

Preference aggregation mechanisms include voting protocols, auctions, divorce settlement procedures, and collaborative rating systems, to name just a few. Unfortunately, most naive preference aggregation mechanisms suffer from *manipulability*. 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. Manipulation is an undesirable phenomenon because preference aggregation mechanisms are tailored to aggregate preferences in a socially desirable way, and if the agents reveal their preferences insincerely, a socially undesirable outcome may be chosen. Manipulability is a pervasive problem across preference aggregation mechanisms. A seminal negative result, the

* This material is based upon work supported by the National Science Foundation under CAREER Award IRI-9703122, Grant IIS-9800994, ITR IIS-0081246, and ITR IIS-0121678.

*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 [25, 47]. (A preference aggregation scheme is called dictatorial if one of the agents dictates the outcome no matter how the others vote.)

*Mechanism design* is the art of designing the *mechanism* (*i.e.*, rules of the game) so that the agents are motivated to report their preferences truthfully and a desirable (according to a given objective) outcome is chosen. The objective can be, for example, social welfare (*i.e.*, sum of the agents' utilities), seller's revenue, fairness, or some combination of these.

## 1.1 Manual mechanism design

Mechanism design has traditionally been a manual endeavor. The designer uses experience and intuition to hypothesize that a certain rule set is desirable in some ways, and then tries to prove that this is the case. Alternatively, the designer formulates the mechanism design problem mathematically and characterizes desirable mechanisms analytically in that framework. These approaches have yielded a small number of canonical mechanisms over the last 40 years, each of which is designed for a class of settings and a specific 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 [48, 9, 27]), or they can be easily applied to any probability distribution over the preferences (*e.g.*, the dAGVA mechanism [24, 2], the Myerson auction [39], and the Maskin-Riley multi-unit auction [38]). However, these 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. For example, Myerson's expected revenue maximizing auction is for selling a single item, and Maskin and Riley's expected revenue maximizing auction is 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.

– The most common mechanisms (*e.g.*, VCG, dAGVA, Myerson auction, and the Maskin-Riley auction) assume that the agents have quasilinear preferences. This means that the utility function of each agent $i \in \{1, \ldots, N\}$ can be written as $u_i(o, \pi_1, \ldots, \pi_N) = v_i(o) - \pi_i$, where $o$ is the outcome and $\pi_i$ is the amount that agent $i$ has to pay. So, very restrictively, it is assumed that 1) the agent's valuation, $v_i$, of outcomes is independent of money, 2) the agent does not care about other agents' payments, and 3) the agent is risk neutral.

In addition to mechanisms, mechanism design research has yielded impossibility results that state that no mechanism works across a class of settings (for varying definitions of "works" and varying classes). For example, the Gibbard-Satterthwaite theorem, discussed above, states that for the class of general preferences, no mechanism works in the sense that 1) the mechanism's outcome can be any one of at least three candidates, 2) the mechanism is nondictatorial, and 3) every agent's dominant strategy is to reveal his preferences truthfully.

## 1.2 Automated mechanism design

In sharp contrast to manual mechanism design, Conitzer and Sandholm in 2002 introduced a systematic approach—called *automated mechanism design*—where the mechanism is automatically created for the setting and objective at hand [16].[1] This has at least four important advantages:

– It can be used in settings beyond the classes of problems that have been successfully studied in (manual) mechanism design to date.
– It can allow one to circumvent the impossibility results: when the mechanism is designed for the setting (instance) at hand, it does not matter that it would not work on preferences beyond those in that setting (*e.g.*, for a class of settings). Even when the optimal mechanism—created automatically—does not circumvent the impossibility, it always minimizes the pain entailed by impossibility.
– It can yield better mechanisms (in terms of better outcomes and/or stronger nonmanipulability guarantees[2]) than the canonical mechanisms because the mechanism capitalizes on the particulars of the setting (the probabilistic (or other) information that the mechanism designer has about the agents' preferences).
  Given the vast amount of information that parties have about each other today, it is astonishing that the canonical mechanisms (such as first-price reverse auctions), which ignore that information, have prevailed thus far. I foresee an imminent revolution, where future mechanisms will be created

[1] Note that automated mechanism design is completely different from *algorithmic mechanism design* [41]. In the latter, the mechanism is designed manually with the goal that *executing* the mechanism is computationally tractable. On the other hand, in automated mechanism design, the mechanism itself is designed automatically.

[2] For example, satisfaction of *ex post* IC and/or IR constraints rather than their *ex interim* variants. These are discussed in Section 2.

automatically. For example, imagine a Fortune 1000 company automatically creating its procurement mechanism based on its statistical knowledge about its suppliers (and potentially also the public prices of the suppliers' inputs, *etc.*). Initial work like this is already being conducted at CombineNet, Inc.
– It shifts the burden of mechanism design from humans to a machine.

## 2 The computational problem

As a first step toward fulfilling the vision of automated mechanism design, we modeled mechanism design as a computational optimization problem [16, 20]. This section reviews that model.

First, the automated mechanism design setting is defined as follows.

**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 $\Theta_i$,*
- *a probability distribution $\gamma_i$ over $\Theta_i$ (in the case of correlated types, there is a single joint distribution $\gamma$ over $\Theta_1 \times \ldots \times \Theta_N$),*
- *a utility function $u_i : \Theta_i \times O \to \mathbb{R}$;*[3]

*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. On the other hand, 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. Specifically, a self-interested designer has an objective function $g(o) + \sum_{i=1}^{N} \pi_i$, where $g : O \to \mathbb{R}$ indicates the designer's own preference over the outcomes, and $\pi_i$ 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.

[3] Though this follows standard game theory notation [37], the fact that the agent has both a utility function and a type is perhaps confusing. It simply means that the agent has a utility function from a finite set of utility functions. If agent is of type 1, then it has the first utility function, an agent of type 2 has the second utility function, and so on. 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.

**Definition 2.** *A* deterministic mechanism without payments *consists of an outcome selection function $o : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N \rightarrow O$. A* randomized mechanism without payments *consists of a distribution selection function $p : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N \rightarrow \mathcal{P}(O)$, where $\mathcal{P}(O)$ is the set of probability distributions over $O$. A* deterministic mechanism with payments *consists of an outcome selection function $o : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N \rightarrow O$ and for each agent $i$, a payment selection function $\pi_i : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N \rightarrow \mathbb{R}$, 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 $\pi_i : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_N \rightarrow \mathbb{R}$.*[4]

There are two types of constraint on the designer in building the mechanism: individual rationality constraints and incentive compatibility constraints. The following subsections will define them, respectively.

## 2.1 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 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 (aka. participation 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 [37].

**Definition 3.** *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, $g(o_0) = 0$.)*

[4] 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.

We can now to define the notions of individual rationality.

**Definition 4.** Individual rationality (IR) *is defined as follows.*

- *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 are left out if payments are not possible.*

## 2.2 Incentive compatibility (IC) constraints

The second type of constraint states that the agents should never have an incentive to misreport their type. For this type of constraint, the two most common variants (or *solution concepts*) are *implementation in dominant strategies*, and *implementation in Bayesian Nash equilibrium.*

**Definition 5.** *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 are left out if 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 *Bayesian Nash equilibrium.*

**Definition 6.** *Given an automated mechanism design setting, a mechanism is said to* implement its outcome and payment functions in Bayesian 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 are left out if payments are not possible.*

### 2.3 The optimization problem

We can now define the computational problem of automated mechanism design.

**Definition 7.** *(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 Bayesian Nash equilibrium). Also, we are told whether payments are possible, and whether randomization is 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.*[5]

## 3 Complexity results

This section discusses the complexity of AMD. An interesting special case is the setting where there is only one agent. In this case, the 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 Bayesian Nash equilibrium only differ on what an agent is assumed to know about other agents' types, the two solution concepts coincide here. This observation is a useful tool in proving hardness results: we proved computational hardness in the single-agent setting, which immediately implies hardness for both IR concepts, for both solution concepts, and for any constant number of agents.

Now we are ready to review the hardness results. It turns out that in settings without side payments, such as voting, designing an optimal (*e.g.*, expected social welfare maximizing) deterministic mechanism is $\mathcal{NP}$-complete. This holds whether the designer is benevolent [16, 14] or self-interested [20]. If side payments are allowed, designing a deterministic mechanism is easy if the designer's objective is social welfare (the VCG mechanism suffices), but $\mathcal{NP}$-complete more generally (for example, if the objective is to maximize the expected revenue collected from the bidders [20]—as is the objective in some auctions). All of these hardness results apply even with a uniform prior over types.

Interestingly, if one allows randomized mechanisms, the mechanism design problem becomes solvable in polynomial time using linear programming (LP)

[5] For studying computational complexity, we phrase AMD as a decision problem, but the corresponding optimization problem is clear.

(for any constant number of agents) [16, 20].[6] A decision variable in the LP is the probability that a given outcome is chosen given that a certain type revelation vector (each agent reveals a type) occurs. For any constant number of agents, the number of decision variables is polynomial in the number of types and in the number of outcomes. Furthermore, the number of constraints (IC and IR) is polynomial. The LP can then be solved in polynomial time.[7]

## 4 A tiny example: Divorce settlement

We built a basic automated mechanism design system to test the approach in practice. This section illustrates a small example (from [19]). For each setting below, our system found the optimal mechanism. The system used CPLEX, a general-purpose optimization package, to solve the underlying mixed integer/linear program.

Consider a couple getting a divorce. They jointly own a painting and the arbitrator has to decide what happens to the painting. There are 4 options to decide between: (1) the husband gets the painting, (2) the wife gets the painting, (3) the painting remains in joint ownership and is hung in a museum, and (4) the painting is burned. The husband and wife each have two possible types: one that implies not caring for the painting too much (low), and one that implies being strongly attached to the painting (high). (low) is had with probability .8, (high) with .2, by each party. To maximize social welfare, the arbitrator would like to give the painting to whoever cares for it more, but even someone who does not care much for it would prefer having it over not having it, making the arbitrator's job in ascertaining the preferences nontrivial. Specifically, the utility function is (for either party):

```
u(low,get the painting)=2
u(low,other gets the painting)=0
u(low,joint ownership)=1
u(low,burn the painting)=-10 (both consider burning it bad from an art history perspective)

u(high,get the painting)=100
u(high,other gets the painting)=0
u(high,joint ownership)=50
u(high,burn the painting)=-10
```

First, let us assume that side payments are not possible, randomization is not possible, and that implementation in dominant strategies is required. Our system generated the following optimal mechanism for this setting:

```
           husband_low            husband_high
wife_low   husband gets painting  husband gets painting
wife_high  husband gets painting  husband gets painting
```

[6] This holds for any mechanism design objective that is linear in the outcome probabilities.

[7] Randomized automated mechanism design can be solved in polynomial time even if the types are correlated, that is, the agents' types are drawn from a joint distribution, not from separate distributions.

So, we cannot do better than always giving the painting to the husband (or always giving it to the wife). (The solver does not look for the "fairest" mechanism because fairness is not part of the objective we specified.) Now let us change the problem slightly, by requiring only implementation in Bayesian Nash equilibrium. For this instance, our system generated the following optimal mechanism:

```
          husband_low        husband_high
wife_low  joint ownership    husband gets painting
wife_high wife gets painting painting is burned
```

Thus, when we relax the incentive compatibility constraint to Bayesian Nash equilibrium, we can do better by sometimes burning the painting! The burning of the painting (with which nobody is happy *per se*) is sufficiently helpful in tailoring the incentives that it becomes a key part of the mechanism.

It turns out that we can do better by also allowing for randomization in the mechanism. The optimal randomized mechanism generated by the system is the following:

```
          husband_low              husband_high
wife_low  .57: husband, .43: wife  1: husband
wife_high 1: wife                  .45: burn; .55: husband
```

The randomization helps because the threat of burning the painting *with some probability* when both report high is enough to obtain the incentive effect that allows us to give the painting to the right party for other type vectors. The mechanism now chooses to randomize over the party that receives the painting rather than awarding joint ownership in the setting where both report low.

We also studied this divorce scenario when the benevolent mechanism designer can use side payments, and when the mechanism designer is self-interested (wants to maximize the amounts paid to him by the divorcees) [19].

## 5 Initial applications

The automated mechanism design approach is new, and so far we have only done preliminary experiments. In addition to solving the small divorce scenarios discussed above, our system has yielded the following highlights [19]:

- It reinvented the celebrated Myerson auction [39], which maximizes the seller's expected revenue in a 1-object auction.
- It created expected revenue maximizing combinatorial auctions. This has been a long-standing recognized open research problem in (manual) mechanism design [4, 49]. The general form for such an auction is still unknown, but automated mechanism design created prior-specific optimal mechanisms. (In the manual mechanism design literature, even the problem with only two objects for sale is open; only a case with very special form of complementarity and no substitutability has been solved [1].)
- It created optimal mechanisms for a public good problem (deciding whether or not to build a bridge). The VCG mechanism could be used in this setting as long as each agent's utility function is quasilinear. However, in the

VCG, nonnegative payments are collected from the voters (intuitively, the payments are collected in order to avoid the free rider problem), and those payments have to be burned. According to a seminal impossibility result, this problem plagues *any* mechanism that applies to general quasilinear utility functions, yields a social welfare maximizing decision, and makes truthful reporting of utility functions a dominant strategy [26]. The automated mechanism design approach allowed us to incorporate money burning as a loss in the social welfare objective, and maximize that revised objective. We had automated mechanism design create an optimal mechanism for the bridge building scenario under each variant of the incentive compatibility (IC) constraint discussed above (with the *ex post* IR constraint). In neither variant was money ever burned. Under the *ex interim* IC constraint, the bridge was always built if and only if that was best for the agents. (Under the *ex post* IC constraint this was not always the case.) For the *ex interim* IC constraint, the general-purpose *dAGVA* mechanism could be used to yield the social welfare maximizing choice without burning money [24, 2]. However, a seminal economic impossibility result shows that no mechanism for general quasilinear utility functions yields the social welfare maximizing choice, maintains budget balance, *and satisfies the IR constraint* (even the *ex interim* variant) [40]. As the experiment above showed, automated mechanism design can circumvent this impossibility! It constructed a mechanism that satisfies all these desiderata, and actually the *ex post* (*i.e.*, stronger) variant of the IR constraint.
- It created optimal mechanisms for public goods problems with multiple goods. This is the public goods analog of combinatorial auctions.

## 6 Structured preferences

If the agents' utility functions are additively decomposable into independent issues, the input to automated mechanism design can be represented (potentially exponentially) more concisely. (An example of this is a multi-item auction where for each bidder, the value of the bundle of items that she wins is simply the sum of the values that she assigns to the individual items in the bundle.)

In that representation it is $\mathcal{NP}$-complete (even under strong restrictions) 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 revenue collected from the agents [21]. However, again, a randomized mechanism can be designed in polynomial time. So, the complexity as a function of the input length is the same in the concise representation as it is in the flat representation. In other words, due to its potentially exponentially shorter input length, the structured representation allows potentially exponentially faster automated mechanism design.

## 7 Conclusions and perspective

*Mechanism design* is the art of designing the rules of the game (aka. *mechanism*) so that a desirable outcome (according to a given objective) is reached despite the fact that each agent acts in his own self-interest. Mechanisms have traditionally been designed manually for classes of problems. In 2002, Conitzer and Sandholm introduced the *automated mechanism design* approach, where the mechanism is computationally created for the specific problem instance at hand. As illustrated in this write-up, this approach has several advantages: 1) it can yield better mechanisms than the ones known to date, 2) it applies beyond the problem classes studied manually to date, 3) it can circumvent seminal economic impossibility results, and 4) it shifts the burden of design from man to machine.

In most variants of the problem, designing a deterministic mechanism is $\mathcal{N}P$-complete (even with just one agent), while a randomized mechanism can be designed in polynomial time using linear programming (for any constant number of agents). Put in perspective, the designer faces uncertainty about the agents' private information, which leads to the need for mechanism design and introduces the associated computational complexity. Interestingly, the designer can remove this complexity *by making the agents face additional uncertainty* (randomization in the mechanism). This comes at no loss, and in some cases at a gain, in the designer's objective because deterministic mechanisms are a subset of randomized ones (the outcome probabilities are 0/1 for each type vector).

Applications overviewed included different types of divorce settlement settings, optimal auctions, optimal combinatorial auctions (a recognized open research problem in manual mechanism design), optimal public goods problems, and optimal combinatorial public goods problems. If the agents' utility functions can be additively decomposed into independent issues, the input to the mechanism design problem can become exponentially shorter, and it turns out that this allows for exponentially faster solving of the design problem.

One potential objection to automatically designed mechanisms is that they can be complex (the mapping from type revelation vectors to outcomes can be long to list), and unintuitive (because they are designed anew for each setting, the agents will likely not have had any previous experience with the mechanism). I would argue that neither of these objections is fundamental: it suffices for each agent to know his best way of *behaving* in the mechanism (which, by the IR and IC constraints, is participating and revealing his type truthfully).

## 8 Current and future research directions

Automated mechanism design is a brand new area of research, and holds significant promise for enormous theoretical and practical impact. In this section I lay out an agenda for current and future research in this area.

### 8.1 Real-world applications

In the short term, the automated mechanism work with greatest practical impact will undoubtedly be the application of the methodology to real-world problems. While we introduced automated mechanism design only recently (in 2002), it is already being adopted in applications. In addition to our own explorations of applications [19] reviewed above, our approach and methodology is being used by others, for example to design auctions [28] and nonmanipulable collaborative rating systems [30]. Promising application areas for the future include other auction settings, voting settings, and a variety of mediation settings.

## 8.2 Partial priors and input representation

One potential criticism is that in automated mechanism design, the prior distribution of types is used. This runs directly against the *Wilson doctrine* of mechanism design that states that the mechanism should be prior-independent because in many settings the designer does not know the prior. I would argue that in many settings the designer does know a lot about the prior, and it would be silly to ignore it. Consider, for example, a Fortune 1000 company procuring materials from its established supplier base. The company certainly has significant statistical information about the suppliers' capacities and production costs. Secondly, in many settings, good mechanisms *must* use the prior. This is necessary, for example, in revenue-maximizing auctions. Therefore also many of the manually designed mechanisms (*e.g.*, the dAGVA mechanism, the Myerson auction, and the Maskin-Riley auction) do use the prior.

A related potential objection arises from the fact that in some settings the type space can be so large that the input to mechanism design is prohibitive. For example, in a combinatorial auction, the number of bundles is exponential in items, and even if every bundle can have a small number of alternative values (for a given agent), the agent's type space is doubly exponential in items.

To make automated mechanism design practical in these settings, it would be desirable to develop ways to use only partial information about the prior (and the type space), and yet design mechanisms that are provably (or experimentally) close to optimal. Furthermore, could the mechanism design software selectively and incrementally elicit partial information about the input *on an as-needed basis* from the human designer who uses the software—and yet design a (close to) optimal mechanism?

Related research questions include the following. How should the type space be discretized if the actual type space is continuous? Can the discretization be avoided entirely in some settings? Can the input be represented more effectively in some settings? (Section 6 showed that if the agents' utility functions are additively decomposable, the answer to the last question is affirmative.)

## 8.3 Special-purpose algorithms and characterization results

While optimal randomized mechanisms are quick to design automatically, optimal deterministic ones tend to be $\mathcal{NP}$-complete to design. CPLEX tends to be able to create optimal deterministic mechanisms with tens of types and tens of outcomes in less than a minute [19]. Future research should improve this scalability through algorithms specially crafted for automated mechanism design.

One interesting approach along that line is to use game-theoretic characterization results (that state features that all mechanisms with certain desirable properties for a class of settings have, *e.g.*, [35, 51]) to prune down the search. One example along these lines is a general search algorithm for designing deterministic mechanisms for one agent [18]. It tends to outperform CPLEX. Another example is the recent design of a 1-object optimal auction mechanism when the objective is a combination of the bidders' welfare and the seller's expected utility [36]. The main piece of that work is an analytical characterization, but at the end, a binary search algorithm is used to set the key parameter—which depends on the prior and for which an analytical solution does not exist. Yet another example is a recent paper on automated determination of an optimal sequence of take-it-or-leave-it-offers (*e.g.*, by a seller to a set of buyers, one buyer at a time) [46]. It has a simple characterization result that makes modeling of the optimization problem viable, thus enabling computational solving. The latter two examples also serve as examples of techniques that are for a special kind of automated mechanism design setting rather than for the general case.

## 8.4 Handling collusion

The mechanism design setting discussed in this write-up so far considers deviations by individual agents. Even if a mechanism is robust against such deviations, a coalition of agents may be better off by reporting their types insincerely. Several solution concepts have been proposed that require robustness against coalitional deviations [3, 7]. Future research should study automated mechanism design under those solution concepts as well.

## 8.5 Inducing general mechanisms and mechanism design principles

Another future use of automated mechanism design is to solve for mechanisms for a variety of settings (real or artificially generated), and to see whether new canonical mechanisms (that work across a *class* of settings) and/or mechanism design principles can be inferred.

## 8.6 Nonstandard mechanism types

Perhaps most fundamentally, automated mechanism design could be used while at the same time relaxing some of the core assumptions of mechanism design. In this section I will discuss some important avenues along this line.

**Multi-stage mechanisms** Often in practice only a portion of the type information is needed to determine the outcome. What information is needed from an agent generally depends on what types the other agents have. There has been significant recent work on selective incremental preference elicitation from bidders in combinatorial auctions [10, 12, 11, 29, 52, 8] [8] and from voters in elections [17]. It turns out that in some settings, exponentially less information is communicated in a multi-stage mechanism than in the most communication-efficient single-step mechanism [22].

So, in practice, multi-stage mechanisms may be desirable in order to reduce communication, enhance privacy, and to reduce the agents' effort in settings where they need to expend effort to determine their own types (for example, in many auctions, a bidder does not know the value of the goods before constructing a plan of what he would do with the goods if he won). Future work includes ***automatically*** designing multi-stage mechanisms.

[8] Ascending (combinatorial) auctions (*e.g.*, [42, 50]) are a special cases of the elicitation model.

**Mechanisms with insincere equilibrium play** The mechanism design framework presented in this write-up creates mechanisms in which truthful type reporting is each agent's best strategy. This is justified by a central design principle in mechanism design, the *revelation principle* [37]. It states that anything that can be accomplished with a mechanism where some agents' best strategies involve insincere reporting, can be accomplished with a mechanism where each agent's best strategy is to reveal his type truthfully.[9]

However, the revelation principle falls apart in practice when computational complexity is an issue. A recent paper [22] shows that there are settings where 1) the optimal truthful mechanism is $\mathcal{N}P$-complete to execute (for the center, *e.g.*, auctioneer, who is running the mechanism), 2) by moving to insincere mechanisms, one can shift the burden of having to solve the $\mathcal{N}P$-complete problem from the center to one of the agents, 3) the insincere mechanism is equally good as the optimal truthful mechanism in the presence of unlimited computation, and most interestingly, 4) whereas being unable to carry out the complex computation would have hurt the center in achieving his objective in the truthful setting, if the agent is unable to carry out the complex computation, the value of the designer's objective *strictly improves.* This shows that there are at least theoretical settings where it is beneficial to use insincere mechanisms. So, is there an advantage in practical settings? What would such mechanisms look like? Are there principles for constructing them? Could they be automatically designed?

**Mechanisms that take into account the agents' bounded rationality** Sometimes economic mechanism design falls short: it can hit one of the impossibility results. One way to try to circumvent impossibility results is to relax the incentive compatibility constraint (and hopefully the *other* desirable properties are obtainable). Then, the agents may have incentive to manipulate. A novel way around this is to design mechanisms where *finding* a beneficial insincere type revelation is provably hard computationally. There has been work characterizing the complexity of manipulating known voting protocols [6, 5, 15, 13], and recent work on designing small changes to voting protocols so that manipulation becomes hard [15, 23]. Future research includes *automatically* designing mechanisms that are provably hard to manipulate.

Another significant issue is that an agent may not know his preferences up front, but can refine them by computing or information gathering. For example, when a trucking company bids for a trucking task, this involves solving (at least)

[9] The proof is remarkably simple: given any mechanism, we can construct a truth-promoting mechanism whose performance is identical, as follows. We build an interface layer between the agents and the original mechanism. The agents report their 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. 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.

two $\mathcal{N}P$-complete planning problems: the vehicle routing problem with the new task and the problem without it [43, 44]. The difference in the costs of those two local plans is the cost of taking on the new task. Should a bidder evaluate the object he is bidding on if there is a cost to doing so? It turns out that the celebrated Vickrey auction loses its dominant-strategy property if the bidder has the option to evaluate the object or not: Whether or not the bidder should pay the evaluation cost depends on the other bidders' valuations [45].

The issues run even deeper. If a bidder has the opportunity to approximate its valuation to different degrees, how much computing time should the bidder spend on refining its valuation? If there are multiple items for sale, how much computing time should the bidder allocate on different bundles of items? A bidder may even allocate some computing time to evaluate other bidders' valuations (*e.g.*, how much it would cost for a competing trucking company to take on a given set of tasks) so as to be able to bid more strategically; this is called *strategic computing* [33, 32, 31, 34].

To answer these questions, we developed a deliberation control method called a *performance profile tree* for projecting how an anytime algorithm (a black box from the perspective of the deliberation controller) will change the valuation if additional computing is allocated toward refining (or improving) it [33, 32, 31, 34]. Unlike earlier deliberation control methods for anytime algorithms, the performance profile tree is a fully normative model of bounded rationality: it takes into account all the information that an agent can use to make its deliberation control decisions. (This is necessary in the game-theoretic context; otherwise a self-interested agent could take into account some information that the model does not, which could lead to strategic instability and much worse results.)

Using this deliberation control method, the computing actions can be made part of the (auction) game. At every point, each agent can decide on which bundle to allocate its next step of computing as a function of the agent's computing results so far (and in open-cry auction format also the others' bids observed so far). At every point, the agent can also decide to submit bids. One can then solve this model for the Bayesian Nash equilibrium, where each agent's (deliberation and bidding) strategy is a best-response to the others' strategies. This is called a *deliberation equilibrium*. With this model, it has been determined under what conditions strategic computing does (not) occur [33, 32, 31, 34].

Our *performance profile tree* based deliberation control method together with the idea of *deliberation equilibrium* provide a normative model of bounded rationality in multiagent systems, which is needed to determine how computationally constrained self-interested agents would behave in a given mechanism. This allows one to evaluate mechanisms for computationally constrained agents, and hopefully paves the way to designing such mechanisms (automatically). This methodology could also be used to design mechanisms that are computationally hard to manipulate, where hardness is measured not in terms of worst-case complexity, but informed by game-theoretic deliberation control. This methodology could even yield new mechanism design principles. As discussed, the central design principle in mechanism design, the *revelation principle*, ceases to meaningfully hold under computational or communication constraints.

## References

1. Mark Armstrong. Optimal multi-object auctions. *Review of Economic Studies*, 67:455–481, 2000.
2. 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.
3. R Aumann. Acceptable points in general cooperative n-person games. volume IV of *Contributions to the Theory of Games*. Princeton University Press, 1959.
4. Christopher Avery and Terrence Hendershott. Bundling and optimal auctions of multiple products. *Review of Economic Studies*, 67:483–497, 2000.
5. John J. Bartholdi, III and James B. Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.
6. John J. Bartholdi, III, Craig A. Tovey, and Michael A. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989.
7. B Douglas Bernheim, Bezalel Peleg, and Michael D Whinston. Coalition-proof Nash equilibria: I concepts. *Journal of Economic Theory*, 42(1):1–12, June 1987.
8. Avrim Blum, Jeffrey Jackson, Tuomas Sandholm, and Martin Zinkevich. Preference elicitation and query learning. In *Conference on Learning Theory (COLT)*, Washington, D.C., 2003.
9. E H Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.
10. Wolfram Conen and Tuomas Sandholm. Preference elicitation in combinatorial auctions: Extended abstract. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 256–259, Tampa, FL, October 2001. A more detailed description of the algorithmic aspects appeared in the IJCAI-2001 Workshop on Economic Agents, Models, and Mechanisms, pp. 71–80.
11. Wolfram Conen and Tuomas Sandholm. Differential-revelation VCG mechanisms for combinatorial auctions. In *AAMAS-02 workshop on Agent-Mediated Electronic Commerce (AMEC)*, Bologna, Italy, 2002.
12. Wolfram Conen and Tuomas Sandholm. Partial-revelation VCG mechanism for combinatorial auctions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 367–372, Edmonton, Canada, 2002.
13. Vincent Conitzer, Jerome Lang, and Tuomas Sandholm. How many candidates are needed to make elections hard to manipulate? In *Theoretical Aspects of Rationality and Knowledge (TARK IX)*, Bloomington, Indiana, USA, 2003.
14. Vincent Conitzer and Tuomas Sandholm. Automated mechanism design: Complexity results stemming from the single-agent setting, 2002. Draft.
15. Vincent Conitzer and Tuomas Sandholm. Complexity of manipulating elections with few candidates. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 314–319, Edmonton, Canada, 2002.
16. 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.
17. Vincent Conitzer and Tuomas Sandholm. Vote elicitation: Complexity and strategy-proofness. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 392–397, Edmonton, Canada, 2002.
18. Vincent Conitzer and Tuomas Sandholm. An algorithm for single-agent deterministic automated mechanism design without payments. In *IJCAI-03 workshop on Distributed Constraint Reasoning (DCR)*, Acapulco, Mexico, 2003.

19. Vincent Conitzer and Tuomas Sandholm. Applications of automated mechanism design. In *UAI-03 workshop on Bayesian Modeling Applications*, Acapulco, Mexico, 2003.
20. Vincent Conitzer and Tuomas Sandholm. Automated mechanism design for a self-interested designer. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 232–233, San Diego, CA, 2003. Poster paper. Full-length draft available at www.cs.cmu.edu/ ˜ sandholm/.
21. Vincent Conitzer and Tuomas Sandholm. Automated mechanism design with a structured outcome space, 2003.
22. Vincent Conitzer and Tuomas Sandholm. Computational criticisms of the revelation principle. In *AAMAS-03 workshop on Agent-Mediated Electronic Commerce (AMEC)*, Melbourne, Australia, 2003. Poster paper.
23. Vincent Conitzer and Tuomas Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.
24. C d'Aspremont and L A Gérard-Varet. Incentives and incomplete information. *Journal of Public Economics*, 11:25–45, 1979.
25. A Gibbard. Manipulation of voting schemes. *Econometrica*, 41:587–602, 1973.
26. J Green and J-J Laffont. *Incentives in Public Decision Making.* Amsterdam: North-Holland, 1979.
27. Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.
28. Eric Hsu. Automated mechanism design: Type space and exponential auction. In *AAMAS-03 workshop on Evolutionary Game Theory for Learning in MAS*, Melbourne, Australia, 2003.
29. Benoit Hudson and Tuomas Sandholm. Effectiveness of preference elicitation in combinatorial auctions. In *AAMAS-02 workshop on Agent-Mediated Electronic Commerce (AMEC)*, Bologna, Italy, 2002. Extended version: Carnegie Mellon University, Computer Science Department, CMU-CS-02-124, March. Also: Stanford Institute for Theoretical Economics workshop (SITE-02).
30. Anthony Jameson, Christopher Hackl, and Thomas Kleinbauer. Evaluation of automatically designed mechanisms. In *UAI-03 workshop on Bayesian Modeling Applications*, Acapulco, Mexico, 2003.
31. Kate Larson and Tuomas Sandholm. Bargaining with limited computation: Deliberation equilibrium. *Artificial Intelligence*, 132(2):183–217, 2001. Short early version appeared in the Proceedings of the National Conference on Artificial Intelligence (AAAI), pp. 48–55, Austin, TX, 2000.
32. Kate Larson and Tuomas Sandholm. Computationally limited agents in auctions. In *AGENTS-01 Workshop of Agents for B2B*, pages 27–34, Montreal, Canada, May 2001.
33. Kate Larson and Tuomas Sandholm. Costly valuation computation in auctions. In *Theoretical Aspects of Rationality and Knowledge (TARK VIII)*, pages 169–182, Sienna, Italy, July 2001.
34. Kate Larson and Tuomas Sandholm. An alternating offers bargaining model for computationally limited agents. In *International Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002.
35. Ron Lavi, Ahuva Mu'Alem, and Noam Nisan. Towards a Characterization of Truthful Combinatorial Auctions, 2003. Draft, April 8th.
36. Anton Likhodedov and Tuomas Sandholm. Auction mechanism for optimally trading off eficiency and revenue. In *AAMAS workshop on Agent-Mediated Electronic Commerce (AMEC V)*, Melbourne, Australia, 2003.

37. Andreu Mas-Colell, Michael Whinston, and Jerry R. Green. *Microeconomic Theory.* Oxford University Press, 1995.
38. 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.
39. Roger Myerson. Optimal auction design. *Mathematics of Operation Research*, 6:58–73, 1981.
40. Roger Myerson and Mark Satterthwaite. Efficient mechanisms for bilateral exchange. *Journal of Economic Theory*, 28:265–281, 1983.
41. Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001. Early version in STOC-99.
42. David C Parkes and Lyle Ungar. Iterative combinatorial auctions: Theory and practice. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 74–81, Austin, TX, August 2000.
43. Tuomas Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 256–262, Washington, D.C., July 1993.
44. Tuomas Sandholm. *Negotiation among Self-Interested Computationally Limited Agents.* PhD thesis, University of Massachusetts, Amherst, 1996. Available at http:// www.cs.cmu.edu/ ˜sandholm/ dissertation.ps.
45. 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.
46. Tuomas Sandholm and Andrew Gilpin. Sequences of take-it-or-leave-it offers: Near-optimal auctions without full valuation revelation. In *AAMAS workshop on Agent-Mediated Electronic Commerce (AMEC V)*, Melbourne, Australia, 2003.
47. 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.
48. W Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.
49. Rakesh V. Vohra. Research problems in combinatorial auctions. Mimeo, version Oct. 29, 2001.
50. Peter R Wurman and Michael P Wellman. AkBA: A progressive, anonymous-price combinatorial auction. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 21–29, Minneapolis, MN, October 2000.
51. Makoto Yokoo. The characterization of strategy/false-name proof combinatorial auction protocols: Price-oriented, rationing-free protocol. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, August 2003.
52. Martin Zinkevich, Avrim Blum, and Tuomas Sandholm. On polynomial-time preference elicitation with value queries. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 176–185, San Diego, CA, 2003.