Computationally Feasible Automated Mechanism Design: General Approach and Case Studies*

Mingyu Guo and **Vincent Conitzer**
Duke University
Department of Computer Science
Durham, NC, USA
{mingyu, conitzer}@cs.duke.edu

Abstract

In many multiagent settings, a decision must be made based on the preferences of multiple agents, and agents may lie about their preferences if this is to their benefit. In mechanism design, the goal is to design procedures (mechanisms) for making the decision that work in spite of such strategic behavior, usually by making untruthful behavior suboptimal. In *automated mechanism design*, the idea is to computationally search through the space of feasible mechanisms, rather than to design them analytically by hand. Unfortunately, the most straightforward approach to automated mechanism design does not scale to large instances, because it requires searching over a very large space of possible functions. In this paper, we describe an approach to automated mechanism design that is computationally feasible. Instead of optimizing over all feasible mechanisms, we carefully choose a parameterized subfamily of mechanisms. Then we optimize over mechanisms within this family, and analyze whether and to what extent the resulting mechanism is suboptimal outside the subfamily. We demonstrate the usefulness of our approach with two case studies.

Mechanism Design Preliminaries

Mechanism design deals with making social decisions in systems involving multiple agents. For a given domain, let O be the set of all possible *outcomes* (social decisions). For example, in resource allocation problems, an outcome specifies who wins which resources, and, if monetary payments are allowed, how much each agent pays. We generally assume that the agents are rational in a game-theoretic sense, and that each agent's preferences are private information for that agent. Let Θ be the space of all possible *types* that agents may have, where agent i's type θ_i contains all i's private information. We generally focus on *direct-revelation* mechanisms, in which each agent makes a report $\hat{\theta}_i \in \Theta$ of her preferences to the mechanism, which then makes the decision based on these reported types. Hence, a mechanism is a function $f : \Theta^n \rightarrow O$, where n is the number of agents. (If randomized mechanisms are allowed, then a mechanism is a function $f : \Theta^n \rightarrow \Delta(O)$, where $\Delta(O)$ are the probability distributions over O.) Of course, each agent may lie so that $\hat{\theta}_i \neq \theta_i$. By a result known as the *revelation principle*, we can restrict attention to direct-revelation mechanisms that incentivize truthful reporting. A mechanism is *strategy-proof* if each agent is best off reporting her type truthfully, no matter what her type is and no matter what the other agents report. A mechanism is *(ex post) individually rational (IR)* if each agent's utility is always nonnegative (as long as she reports truthfully), so that participating is always optimal. The goal is to design a mechanism that satisfies all the desired properties, and performs well according to some objective. Example objectives include maximizing expected/worst-case *social welfare* (sum of the agents' utilities) as well as maximizing expected/worst-case *revenue* (total payment collected from the agents by the mechanism).

Automated Mechanism Design

Traditionally, economists have designed mechanisms manually, based on analytical techniques. This has resulted in many good mechanisms, such as the VCG mechanism. However, there are many different variants of the mechanism design problem, depending on the specific desired properties and objective, and many of these variants remain unsolved. Often, they correspond to very difficult optimization problems, and the solutions can take very complex forms. This is where automated mechanism design can be of help.

The first precise general formulation of the automated mechanism design problem was given by Conitzer and Sandholm (2002). More details on the general formulation can be found in a chapter by Conitzer (2006). Automated mechanism design (broadly interpreted) has also been used for optimizing over specific families of parameterized mechanisms, *e.g.*, (Cliff 2001; Phelps et al. 2002; Byde 2003; Likhodedov and Sandholm 2005; Vorobeychik, Kiekintveld, and Wellman 2006; Reeves, Soule, and Kasturi 2007). Our general approach also involves such an optimization step. The principle of automated mechanism design has also been applied to other settings (Jurca and Faltings 2006; Constantin and Parkes 2007).

The basic idea of automated mechanism design is to solve

[0]This paper gives a coherent perspective on a line of our research. No discussion of this perspective has previously appeared, but the prior research discussed appeared in the following papers: (Guo and Conitzer 2008a; 2008b; 2009; 2010). Guo and Conitzer are supported by NSF under award number IIS-0812113 and CAREER 0953756, and by an Alfred P. Sloan Research Fellowship.

for the function f as a constrained optimization problem, where the desired properties (e.g., strategy-proofness, IR) correspond to the constraints in the optimization, and the objective (e.g., social welfare, revenue) corresponds to the objective in the optimization.

To illustrate the framework, suppose that we have a prior distribution p for the true type vector of the agents, $p : \Theta^n \rightarrow [0,1]$. Let $u_i(\theta, o)$ be the utility that agent i obtains if she has true type θ and the outcome is o. Finally, suppose (for the sake of example) that strategy-proofness is the only desired property (no IR), we want the mechanism to be deterministic, and we aim to maximize expected social welfare. We obtain the following general formulation:

Variable function: $f : \Theta^n \rightarrow O$
Maximize
$\sum_{\vec{\theta} \in \Theta^n} p(\vec{\theta}) \sum_{i=1}^{n} u_i(\theta_i, f(\vec{\theta}))$ (expected social welfare)
Subject to:
$(\forall i \in \{1, 2, \ldots, n\}, \vec{\theta} \in \Theta^n, \theta_i' \in \Theta)$
$u_i(\theta_i, f(\vec{\theta})) \geq u_i(\theta_i, f((\theta_1, \ldots, \theta_{i-1}, \theta_i', \theta_{i+1}, \ldots, \theta_n))$
(strategy-proofness)

A key problem is that this is a problem of optimizing a *function*. The most basic approach to solving such problems is as follows (Conitzer and Sandholm 2002). If the type space Θ and the outcome space O are finite, and we allow for randomized mechanisms, then it is possible to write this as a linear program: this is done by defining a probability variable $p_f(o|\vec{\theta})$ for each $\vec{\theta} \in \Theta^n$ and $o \in O$. For the case where we require a deterministic mechanism, we can add the constraint that each of these probabilities must be in $\{0, 1\}$ to obtain an integer program (in the deterministic case, the problem is generally NP-hard even with one agent). Still, the scalability of this approach is very limited. One reason is that both the number of variables and the number of constraints are exponential in n. Another problem is that type spaces are generally not finite and require discretization for this approach to work. For small instances, this approach is feasible, and the solutions to these small instances will sometimes allow us to conjecture more general results, but generally the limitations on scalability are too constraining.

High-Level Description of Our Approach

Our general approach to addressing the scalability issue can be described as follows.

- For a specific setting, let Ω denote the set of all feasible mechanisms.
- Instead of optimizing over Ω directly, we first choose a suitable subfamily $\Omega' \subseteq \Omega$ based on analytical considerations, where the mechanisms in Ω' can be parameterized using k parameters, so that a vector $(c_1, \ldots, c_k)$ defines a mechanism $f_{c_1, c_2, \ldots, c_k} \in \Omega'$.
- We then optimize over $(c_1, \ldots, c_k)$, which is generally a much easier problem.
- Finally, we analytically study the suboptimality of the resulting mechanism in the full set Ω.

Unlike in the more basic automated mechanism design approach described earlier, this approach requires significant human input: we need to find a good subfamily Ω' as well as a formulation/algorithm for optimizing over this subfamily, and in the end we need to analyze the suboptimality of the resulting mechanism by hand. From the perspective of artificial intelligence, it may be disappointing that the approach requires so much domain-specific expertise from humans. On the other hand, we have been much more successful at contributing new results to microeconomic theory with this human-machine interactive approach, so we believe that at this point, this is the best way in which artificial intelligence can make contributions to the theory of mechanism design.

The key step is choosing Ω'. If Ω' is too restrictive, then even if we find the optimal mechanism in it, its performance will be too suboptimal. If Ω' is too general, then the optimization problem becomes too difficult again. That is, we want Ω' to be general enough that an (almost) optimal mechanism exists in Ω', and Ω' to be specific enough that we know how to optimize over its parameters. Of course, choosing a good Ω' is not as simple as doing a binary search on the specificity-generality spectrum: an unfortunate choice may result in an Ω' that is difficult to optimize over and still does not produce good mechanisms! The point is that there is some "art" involved in choosing a good Ω'.

In the remainder of this paper, we present two case studies corresponding to our previous work. The first studies resource allocation mechanisms that redistribute their revenue to the agents. Here, we found a good Ω' with a bottom-up approach, by finding a natural generalization of some good example mechanisms. The second studies resource allocation mechanisms that do not require payments at all. Here, we found a good Ω' with a top-down approach, by increasingly restricting the family of mechanisms until we found one that is easy to optimize over. In both cases, we obtained mechanisms that are optimal or nearly optimal even in the general family Ω.

Case Study 1: Redistributing Vickrey Revenue

Problem description. We study the problem of allocating one item among n agents. An example single-item allocation scenario would be a multi-user computer system in which a group of users need to decide who gets to use a certain system resource at a certain time. (Our full research papers study more general allocation problems; we focus on single-item allocation here for the purpose of presentation.) Each agent i has a true valuation v_i for the item; if she wins at a price of π_i, her utility is $v_i - \pi_i$ (quasilinear utilities). Without loss of generality, assume $v_1 \geq v_2 \geq \ldots \geq v_n$.

The standard strategy-proof mechanism for this is the *Vickrey auction*: the highest bidder (bidder 1) wins and pays the second-highest bid, v_2. (Because we are interested in strategy-proof mechanisms, here we do not distinguish between true and reported preferences.) While this is a natural mechanism if there is a seller who can collect the revenue v_2, this is not always the case. For example, the agents may be trying to allocate a jointly owned resource—for instance, who gets to use our jointly owned machine on the day of the

deadline? In this case, it makes sense for the auction's revenue to be redistributed to the agents, rather than to simply throw it away. Unfortunately, doing so naïvely will break the strategy-proofness property: the second-highest bidder will want to make the highest bidder pay more, if the former receives a fraction of this payment. In fact, it is not possible to redistribute *all* the revenue and maintain strategy-proofness. However, it is possible to redistribute a *large fraction* of the revenue, as the next two examples show.

Two example redistribution mechanisms. The first example redistribution mechanism is called the Bailey-Cavallo mechanism, which was proposed independently (from slightly different perspectives) in at least three papers (Bailey 1997; Porter, Shoham, and Tennenholtz 2004; Cavallo 2006). The idea is as follows. Giving each agent an equal share v_2/n of the revenue does not work because it introduces an incentive for the second-highest bidder to make the highest bidder pay more. Instead, under the BC mechanism, each agent i receives v_2^{-i}/n, where v_2^{-i} is the second-highest bid *among bids other than i's bid*. (Hence, $v_2^{-i} = v_3$ for $i \in \{1, 2\}$, and $v_2^{-i} = v_2$ for $i \in \{3, \ldots, n\}$.) Because it is impossible for an agent to affect her own redistribution with her bid, the incentives are the same as in the Vickrey auction, hence the mechanism is strategy-proof. The total amount redistributed is $2v_3/n+(n-2)v_2/n$, which is never more than the Vickrey revenue v_2 (so the redistribution scheme is feasible), and it is at least $(n-2)/n$ of the Vickrey revenue.

While this is a very natural mechanism, we found (by hand) a different mechanism that also redistributes a large amount of revenue, based on a similar idea. Under this mechanism, the amount redistributed to bidder i is $\frac{v_2^{-i}}{n-2} - \frac{2v_3^{-i}}{(n-2)(n-3)}$ (where v_3^{-i} is the third-highest bid among bids other than i's). The total amount redistributed by this mechanism is $v_2 - \frac{6v_4}{(n-2)(n-3)}$, which is never more than the Vickrey revenue v_2 and at least $1 - \frac{6}{(n-2)(n-3)}$ of the Vickrey revenue (we require that n is large enough that $1 - \frac{6}{(n-2)(n-3)} \geq 0$).

Which of the two redistribution mechanisms is better? Each redistributes more than the other in some cases. However, if n is large enough, the second mechanism redistributes a larger fraction of the Vickrey revenue in the worst case. This immediately leads to the question of whether there are even better redistribution mechanisms, for example, using v_4^{-i}.

Linear redistribution mechanisms. This also gives us a very natural subfamily Ω' of mechanisms to optimize over, namely, the mechanisms where the redistribution to agent i is $c_0 + c_1 v_1^{-i} + c_2 v_2^{-i} + \ldots + c_{n-1} v_{n-1}^{-i}$ for some vector of constants $(c_0, \ldots, c_{n-1})$. We call this the family of *linear redistribution mechanisms*, and each of its members is uniquely identified by its vector $(c_0, \ldots, c_{n-1})$, giving us a natural parameterization.

Results. To optimize for the worst-case redistribution fraction, it is possible to set up a linear program for finding the best values for the c_i. A few tricks are required to set up this linear program; the details can be found in (Guo and Conitzer 2009). This linear program gives us the optimal solution for any given n. This allowed us to obtain the optimal solution for specific values of n using a standard linear program solver. By inspection of these values, we were able to generalize these solutions to the case of general n, to obtain the following mechanism: $c_0 = c_1 = 0$ and

$$c_i = \frac{(-1)^i (n-1) \sum_{j=i}^{n-1} \binom{n-1}{j}}{i \sum_{j=1}^{n-1} \binom{n-1}{j}\binom{n-1}{i}} \text{ for } i = 2, \ldots, n-1.$$

The corresponding worst-case fraction of Vickrey revenue redistributed equals $1 - \binom{n-1}{1} / \sum_{j=1}^{n-1} \binom{n-1}{j}$.

Not only does this mechanism perform better than the existing mechanisms in the worst case, but (from simulations) it actually turns out that in many cases it also performs better *on average*.

The final step is to analyze how suboptimal this mechanism is in the family of all feasible mechanisms. In this case, it turns out that this mechanism is actually optimal among all mechanisms—the proof is in (Guo and Conitzer 2009). It should be noted that this mechanism is not uniquely optimal. Guo and Conitzer (2008c) and Apt *et al.* (2008) show that there are mechanisms outside of the linear family that redistribute at least as much in every case, and strictly more in some cases—though, of course, they redistribute the same amount in the worst case.

The mechanism described above was independently derived by Moulin (2009), as the solution to a slightly different optimization problem. (The solution in our paper (Guo and Conitzer 2009) is for a somewhat larger class of settings.) We used similar techniques to find mechanisms that maximize *expected* redistribution, when a prior distribution is available (Guo and Conitzer 2008b). Another insight is that we can sometimes decrease the loss from failing to redistribute by not allocating the items to the agents who value them most, with the net effect being an increase in welfare (Guo and Conitzer 2008a; de Clippel, Naroditskiy, and Greenwald 2009).

Case Study 2: Strategy-proof Allocation without Payments

Problem Description. We now study the problem of allocating m items between two agents, in settings where monetary payments are not feasible (*e.g.*, when we are dealing with software agents, or in other settings in which there is no established currency, such as in a peer-to-peer network). We allow for randomized mechanisms (equivalently, we can consider divisible items). We assume that the agents have additive utility functions, so that an agent who receives item j with probability a_j obtains a total utility of $\sum_j a_j v_j$, where v_j is her utility for item j. We can also think of a_j as the proportion of item j that the agent receives. In order to be able to compare agents' utilities to each other, we assume $\sum_j v_j = 1$ for every agent (when an agent wins all the items, she is 100% satisfied).

Given two mechanisms, if for all possible type vectors, the social welfare under mechanism one is at least α times

the social welfare under mechanism two, then we say mechanism one is α-*competitive* against mechanism two, where α is called the *competitive ratio*, and the maximal possible competitive ratio is called the *maximal competitive ratio*. Our goal is to design a strategy-proof mechanism that has high competitive ratio against the first-best allocation mechanism (the mechanism that always successfully picks the allocation that maximizes the social welfare—of course, the first-best allocation mechanism is not strategy-proof itself).

Linear increasing-price (LIP) family. We do not have a useful characterization of all strategy-proof mechanisms, nor do we know how to evaluate the maximal competitive ratio of an arbitrary strategy-proof mechanism (against the first-best allocation mechanism). We performed a sequence of restrictions on the family of mechanisms, starting with all strategy-proof mechanisms and eventually reaching the family of linear increasing-price (LIP) mechanisms (due to space constraint, we omit description of the intermediate families). Any LIP mechanism is characterized by a single parameter c ($c > 0$). $LIP(c)$ is defined as follows.

LIP$(\mathbf{c})$: *We flip a coin that determines which agent becomes the dictator. The dictator is endowed with* 1 *unit of* artificial *currency, which can be used to purchase items. At any moment, the instantaneous price charged to the dictator for buying another infinitesimal part of item* j *is* $cx_j + \frac{c}{e^c-1}$*, where* x_j *is the amount of artificial currency the dictator has already spent on* j *at that moment. That is, if the dictator spends* x *of her artificial currency on item* j*, then she receives that item with probability* $\int_0^x 1/(ct+\frac{c}{e^c-1})dt$*. After the dictator exhausts her artificial currency, the other agent gets what is remaining.*

Results. For at least two items, for any $c > 0$, we proposed a technique that computes the maximal competitive ratio of $LIP(c)$. (To be more precise, the technique computes a close lower bound on the maximal competitive ratio of $LIP(c)$.) Since the LIP family is characterized by only one parameter, we simply searched for the values of c that correspond to high competitive ratios. For the case of two items, we found that $LIP(2)$'s maximal competitive ratio is (at least) 0.828, which is very close to an upper bound of 0.841 that we derive for any strategy-proof mechanism. That is, for the case of two items, it is without much loss of optimality to focus on the LIP family. However, as the number of items increases, the competitive ratios become worse. That is, for large numbers of items, we may want to consider other families of mechanisms.

Conclusion

We believe that the approach to automated mechanism design that we have described in this paper, where the analytical capabilities of a human mechanism designer work in concert with algorithms that search through restricted families of possible mechanisms, is currently the most promising avenue for techniques from artificial intelligence to contribute to the theory of mechanism design and (perhaps) microeconomic theory in general. The human mechanism designer plays an essential role at several points in this process. For future research on automated mechanism design, it would be desirable to find ways to reduce the burden that is placed on the human designer.

References

Apt, K.; Conitzer, V.; Guo, M.; and Markakis, E. 2008. Welfare undominated Groves mechanisms. In WINE, 426–437.

Bailey, M. J. 1997. The demand revealing process: to distribute the surplus. *Public Choice* 91:107–126.

Byde, A. 2003. Applying evolutionary game theory to auction mechanism design. In EC, 192–193.

Cavallo, R. 2006. Optimal decision-making with minimal waste: Strategyproof redistribution of VCG payments. In AAMAS, 882–889.

Cliff, D. 2001. Evolution of market mechanism through a continuous space of auction-types. Technical Report HPL-2001-326, HP Labs.

Conitzer, V., and Sandholm, T. 2002. Complexity of mechanism design. In UAI, 103–110.

Conitzer, V. 2006. *Computational aspects of preference aggregation*. Ph.D. Dissertation, Carnegie Mellon University. Available as technical report CMU-CS-06-145.

Constantin, F., and Parkes, D. 2007. On revenue-optimal dynamic auctions for bidders with interdependent values. In AMEC, 1–15.

de Clippel, G.; Naroditskiy, V.; and Greenwald, A. 2009. Destroy to save. In EC, 207–214.

Guo, M., and Conitzer, V. 2008a. Better redistribution with inefficient allocation in multi-unit auctions with unit demand. In EC, 210–219.

Guo, M., and Conitzer, V. 2008b. Optimal-in-expectation redistribution mechanisms. In AAMAS, 1047–1054. Journal version to appear in AIJ.

Guo, M., and Conitzer, V. 2008c. Undominated VCG redistribution mechanisms. In AAMAS, 1039–1046.

Guo, M., and Conitzer, V. 2009. Worst-case optimal redistribution of VCG payments in multi-unit auctions. *Games and Economic Behavior* 67(1):69–98.

Guo, M., and Conitzer, V. 2010. Strategy-proof allocation of multiple items between two agents without payments or priors. To appear in AAMAS.

Jurca, R., and Faltings, B. 2006. Minimum payments that reward honest reputation feedback. In EC, 190–199.

Likhodedov, A., and Sandholm, T. 2005. Approximating revenue-maximizing combinatorial auctions. In AAAI, 267–273.

Moulin, H. 2009. Almost budget-balanced VCG mechanisms to assign multiple objects. *Journal of Economic Theory* 144(1):96–119.

Phelps, S.; McBurnley, P.; Parsons, S.; and Sklar, E. 2002. Co-evolutionary auction mechanism design. *Lecture Notes in AI* 2531.

Porter, R.; Shoham, Y.; and Tennenholtz, M. 2004. Fair imposition. *Journal of Economic Theory* 118:209–228.

Reeves, D. M.; Soule, B. M.; and Kasturi, T. 2007. Yootopia! *SIGecom Exchanges* 6(2):1–26.

Vorobeychik, Y.; Kiekintveld, C.; and Wellman, M. 2006. Empirical mechanism design: Methods, with application to a supply chain scenario. In EC, 306–315.