Statement of Interest for Bayesian Modeling Applications Workshop: Applications of Automated Mechanism Design*

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

1 Topic Overview

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 (for example, lie about their preferences). The measure of desirability can be, for instance, social welfare, fairness, seller's revenue, or any combination of these. Thus, mechanism design covers a wide spectrum of protocol design problems, including the design of auctions, elections, public goods mechanisms, and dispute settlement rules. Mechanism design is receiving growing attention in the general AI literature, and in the UAI community in particular [5, 12].

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 [13, 4, 9], the dAGVA mechanism [7, 2], 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)* (introduced at UAI-02 [5]), a mechanism is *computed* on the fly for the setting at hand—a universally applicable approach. The ability to design the mechanism specifically for the setting at hand has numerous advantages. First, it allows one to create mechanisms for settings where none of the classical mechanisms are applicable. Second, it may allow one to circumvent impossibility results that show that there is no mechanism that works across a family of settings (if the mechanism is designed for the setting at hand, it does not matter that it would not work in other settings). Even if the impossibility result applies to the setting at hand, the pain of the impossibility will be minimized by the automated mechanism design approach. Third, the automated mechanism design approach can lead to better mechanisms than the canonical ones, both in terms of stronger nonmanipulability guarantees and in terms of more desirable outcomes, by capitalizing on knowledge about the situation at hand.

To turn the automated mechanism design problem into a concrete computational problem, in our initial research we have modeled the problem as follows. The mechanism designer knows the set of possible outcomes, the set of agents that will participate in the mechanism, and all the preferences (types) that they can possibly have. (We focus on the case where there are only finitely many types, but this can be used to approximate real-valued preferences.) The designer also has a belief about which types are likely, given by a probability distribution over the agents' type vectors. Finally, the designer has a set of constraints on the mechanism (can it be randomized, are side payments possible, can the agents be made worse off by the mechanism, how strong is the nonmanipulability requirement), and an objective.

2 Results

In our prior research on automated mechanism design [5], we studied the computational complexity of the general problem, and showed that typically, the problem is NP-complete when randomized mechanisms are not allowed, but solvable in polynomial time (by linear programming) otherwise. In contrast, in our recent research we have created software that can solve arbitrary automated mechanism design instances using a mixed integer/linear program solver (CPLEX 8.0). We then used this software to solve some interesting instances:

* 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-item optimal auctions.** In an optimal auction, the auctioneer seeks to maximize its (expected) revenue from the auction. In the 1-item setting, the mechanisms produced by our software coincide with the celebrated Myerson auction [11] for optimally auctioning off a single good.

• **optimal combinatorial auctions.** The design of auctions that maximize the seller's expected revenue when there are multiple items (that is, a *combinatorial* auction) is a recognized open research problem [3, 14]. The problem is open even if there are only two goods for sale. (The two-good case with a very special form of complementarity and no substitutability has been solved recently [1].) Our software produced optimal combinatorial auctions for specific settings, which led to some interesting general observations.

• **divorce settlements.** Our software also produced optimal mechanisms for divorce settlement (both with the objective of maximizing the sum of the divorcees' utilities and with the objective of maximizing revenue for the arbitrator).

• **public good problems.** In public good problems, a decision has to be made on whether to build a good which would benefit multiple parties (a bridge, for example). The good has to be financed by agents who will benefit from it, but care has to be taken to avoid the *free rider* problem, where agents pretend to not care for the good much in order to shift the burden of paying for it to other agents. There are no completely satisfactory general mechanisms for solving public goods problems (that always lead to the social welfare maximizing decision), unless making unnecessary payments (which will have to be burned) is not considered a loss [8, 10]. For a specific setting, our software produced optimal public good mechanisms that took burning money into account as a loss. The mechanisms never burned any money, and some of them (depending on the nonmanipulability constraint) always made the social welfare maximizing decision.

• **combinatorial public good problems.** In the case where multiple goods can be built, we face a *combinatorial* public goods problem. (Just like an auction with multiple items is a combinatorial auction.) Our software also produced optimal mechanisms for combinatorial public goods problems.

We also considered the scalability of our software by testing it on randomly generated, unstructured instances. The experiments show that the runtime is heavily dependent upon 1. the number of agents (the more the harder), 2. the number of types per agent (the more the harder), 3. the strength of the nonmanipulability concept (the stronger the harder), 4. whether randomization is allowed (deterministic mechanism design is harder), and 5. which objective is pursued (from social welfare with payments that are not taken into account (easy) to payment maximization (hard)). Other variables, such as the number of outcomes and which (if any) participation constraint is used, turned out to affect the required runtime much less.

3 Future research

AMD in a very new fertile research area with numerous promising directions for future research. First, there would be great value in designing AMD algorithms that are faster than the general-purpose MIP solver used here (CPLEX 8.0) on the general AMD problem or special cases of it. (We have already created a special purpose algorithm for the case where there is only one agent, the mechanism is deterministic, and no payments are allowed; this algorithm typically outperforms CPLEX 8.0 significantly [6].)

Second, one could apply AMD to large-scale real-world problems. This will be the goal of the first author's internship at CombineNet, Inc. this summer.

Third, it would be interesting to use AMD as a tool for finding mechanisms for *classes* of mechanism design settings; this would most likely entail solving several instances using AMD, and trying to identify the commonality of the resulting mechanisms.

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] Christopher Avery and Terrence Hendershott. Bundling and optimal auctions of multiple products. *Review of Economic Studies*, 67:483–497, 2000.

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

[5] 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.

[6] Vincent Conitzer and Tuomas Sandholm. An algorithm for single-agent deterministic automated mechanism design without payments, 2003.

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

[8] J Green and J-J Laffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica*, 45:427–438, 1977.

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

[10] Roger Myerson and Mark Satterthwaite. Efficient mechanisms for bilateral exchange. *Journal of Economic Theory*, 28:265–281, 1983.

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

[12] R. Porter, A. Ronen, Y. Shoham, and M. Tennenholtz. Fault tolerant mechanism design. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, Edmonton, Canada, 2002.

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

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