Automated Mechanism Design in Infinite Games of Incomplete Information: Framework and Applications

Yevgeniy Vorobeychik
University of Michigan
Computer Science & Engineering
Ann Arbor, MI 48109-2121 USA
yvorobey@umich.edu

Daniel M. Reeves
Yahoo! Research
45 W 18th St, 6th Floor
New York, NY 10011-4609 USA
dreeves@yahoo-inc.com

Abstract

We present a functional framework for automated mechanism design based on a two-stage game model of strategic interaction between the designer and the mechanism participants, and apply it to several classes of two-player infinite games of incomplete information. Our approach yields optimal or nearly optimal mechanisms in two application domains using various objective functions. By comparing our results with known optimal mechanisms, and in some cases improving on the best known mechanisms, we show that ours is a promising approach to parametric design of indirect mechanisms.

Motivation

While the field of Mechanism Design has been quite successful within a wide range of academic disciplines, much of its progress came as a series of arduous theoretical efforts. In its practical applications, however, successes have often been preceded by a series of setbacks, with the drama of auctioning radio spectrum licenses that unfolded in several countries providing a powerful example (Klemperer, 2004; McMillan, 1994). The following quote by Klemperer (2004) is, perhaps, especially telling: "Most of the extensive auction literature... is of second-order importance for practical auction design."

A difficulty in practical mechanism design that has been especially emphasized is the unique nature of most practical design problems. Often, this uniqueness is manifest in the idiosyncratic nature of objectives and constraints. For example, when the US government tried to set up a mechanism to sell the radio spectrum licenses, it identified among its objectives promotion of rapid deployment of new technologies. Additionally, it imposed a number of non-traditional constraints, such as ensuring that some licenses go to minority-owned and women-owned companies (McMillan, 1994).

Thus, a prime motivation for Conitzer and colleagues' automated mechanism design work (for example, Conitzer & Sandholm (2003a, 2004); Sandholm, Conitzer, & Boutilier (2007)) was to produce a framework for solving mechanism design problems computationally given arbitrary objectives and constraints. While we are similarly motivated, we recognize and tackle an additional problem: reliance on direct truthful mechanisms. This reliance has at its core the Revelation Principle (Myerson, 1981), which states that any outcome that can be achieved by an arbitrary mechanism can also be achieved if we restrict the design space to mechanisms that induce truthful revelation of agent preferences. While theoretically sound and general in settings without non-traditional constraints, there have been criticisms of the principle on computational grounds, for example, those leveled by Conitzer & Sandholm (2003b). It is also well recognized that if the design space is restricted in arbitrary ways, the revelation principle need not hold.[1] While the former set of criticisms can be addressed to some degree by multi-stage mechanisms that implement partial revelation of agent preferences in a series of steps (for example, an ascending auction), the latter criticisms are a property of the idiosyncratic constraints on the design problem at hand and offer a more difficult hurdle to overcome.

In this work, we introduce an approach to the design of indirect mechanisms[2] and apply it to several mechanism design problems. In many ways, our methods follow in the footsteps of the work on empirical mechanism design (Vorobeychik, Kiekintveld, & Wellman, 2006), although here we present a more systematic approach, albeit restricted to a particular class of infinite games of incomplete information. In practice, of course, we cannot possibly tackle an arbitrarily complex design space. Our simplification comes from assuming that the designer seeks to find the best setting for particular design parameters. In other words, we allow the designer to search for a mechanism in some subset of an n-dimensional Euclidean space, rather than in an arbitrary function space, as would be required in a completely general setting.

In the following sections, we present our framework for automated mechanism design and test it out in several application domains. Our results suggest that our approach has much promise: most of the designs that we discover auto-

[1] As a simple example, imagine that the designer's only choice is a first-price sealed-bid auction. Since this auction is not truthful, the revelation principle clearly fails in this restricted design space. Abstractly and generally, we can simply imagine eliminating truthful mechanisms from the design space.

[2] Indeed, several of our applications yield Bayes-Nash equilibrium strategies that do not correspond to any player type, although all equilibria we obtain are linear functions of the type parameter.

matically are nearly as good as or better than the best known hand-built designs in the literature.

Preliminaries

We restrict our attention to *one-shot games of incomplete information*, denoted by $[I, \{A_i\}, \{T_i\}, F(\cdot), \{u_i(a,t)\}]$, where I refers to the set of players and $m = |I|$ is the number of players. A_i is the set of actions available to player $i \in I$, with $A = A_1 \times \ldots \times A_m$ representing the set of joint actions of all players. T_i is the set of types (private information) of player i, with $T = T_1 \times \cdots \times T_m$ representing the joint type space, whereas $F(\cdot)$ is the joint type distribution. We define a strategy of a player i to be a function $s_i : T_i \rightarrow \mathbb{R}$, and will use $s(t)$ to denote the vector $(s_1(t_1), \cdots, s_m(t_m))$. It is often convenient to refer to a strategy of player i separately from that of the remaining players. To accommodate this, we use a_{-i} to denote the joint action of all players other than player i. Similarly, t_{-i} designates the joint type of all players other than i.

We define the payoff (utility) function of each player i by $u_i : A \times T \rightarrow \mathbb{R}$, where $u_i(a_i, t_i, a_{-i}, t_{-i})$ indicates the payoff to player i with type t_i for playing strategy a_i when the remaining players with joint types t_{-i} play a_{-i}.

Automated Mechanism Design Framework

General Framework

We can model the strategic interactions between the designer of the mechanism and its participants as a two-stage game (Vorobeychik, Kiekintveld, & Wellman, 2006). The designer moves first by selecting a value θ from a set of allowable mechanism settings, Θ. All the participant agents observe the mechanism parameter θ and move simultaneously thereafter. For example, the designer could be deciding between a first-price and a second-price sealed-bid auction mechanism, with the presumption that after the choice has been made, the bidders will participate with full awareness of the auction rules.

Since the participants know the mechanism parameter, we define a game between them in the second stage as

$$\Gamma_\theta = [I, \{A_i\}, \{T_i\}, F(\cdot), \{u_i(a,t,\theta)\}].$$

We refer to Γ_θ as the game *induced* by θ. As is common in mechanism design literature, we will evaluate mechanisms with respect to a sample Bayes-Nash equilibrium, $s(t,\theta)$. We say that given an outcome of play r, the designer's goal is to maximize a welfare function $W(r,t,\theta)$ with respect to the distribution of types. Thus, given that a Bayes-Nash equilibrium, $s(t,\theta)$, is the relevant outcome of play, the designer's problem is to maximize $W(s(\theta),\theta) = E_t[W(s(t,\theta),t,\theta)]$.[3]

Observe that if we *knew* $s(t,\theta)$ as a function of θ, the designer would simply be faced with an optimization problem. This insight is actually a consequence of the application of backwards induction, which would have us find $s(t,\theta)$ first for every θ and then compute an optimal mechanism with respect to these equilibria. If the design space were small, backwards induction applied to our model would thus yield an algorithm for optimal mechanism design. Indeed, if additionally the games Γ_θ featured small sets of players, strategies, and types, we would say little more about the subject. Our goal, however, is to develop a mechanism design tool for settings in which these assumptions do not hold. Specifically, we presume that it is infeasible to obtain a solution of Γ_θ for every $\theta \in \Theta$, either because the space of possible mechanisms is large, or because solving (or approximating solutions to) Γ_θ is very computationally intensive. Additionally, we try to avoid making assumptions about the objective function or constraints on the design problem or the agent type distributions. We do restrict the games to two players with piecewise linear utility functions, but allow them to have infinite strategy and type sets.

In short, we propose the following high-level procedure for finding optimal mechanisms:

1. Select a candidate mechanism, θ.
2. Find (approximate) solutions to Γ_θ.
3. Evaluate the objective and constraints given solutions to Γ_θ.
4. Repeat this procedure for a specified number of steps.
5. Return an approximately optimal design based on the resulting optimization path.

Designer's Optimization Problem

We begin by treating the designer's problem as black-box optimization, where the black box produces a noisy evaluation of an input design parameter, θ, with respect to the designer's objective, $W(s(\theta),\theta)$, given the game-theoretic predictions of play. Once we frame the problem as a black-box (simulation) optimization problem, we can draw on a wealth of literature devoted to developing methods to approximate optimal solutions (Olafsson & Kim, 2002). While we can in principle select a number of these, we have chosen simulated annealing, as it has proved quite effective for a great variety of simulation optimization problems in noisy settings, particularly in problems with many local optima (Corana *et al.*, 1987; Fleischer, 1995; Siarry *et al.*, 1997). We opt for a relatively simple adaptive implementation of simulated annealing, with normally distributed random perturbations applied to the solution candidate in every iteration (respecting the constraint set Θ).

As an application of black-box optimization, the mechanism design problem in our formulation is just one of many problems that can be addressed with one of a selection of methods. What makes it special is the subproblem of evaluating the objective function for a given mechanism choice, and the particular nature of typical mechanism design constraints.

Objective Evaluation

As implied by the backwards induction process, we must obtain the solutions (a Bayes-Nash equilibrium in our case) to a game induced by the design choice, θ, in order to evaluate

[3] Note the overloading of $W(\cdot)$.

the objective function. In general, this is simply not possible to do, since Bayes-Nash equilibria may not even exist in an arbitrary game, nor is there a general-purpose tool to find them. However, there are a number of tools that can find or approximate solutions in specific settings. For example, GAMBIT (McKelvey, McLennan, & Turocy, 2005) is a general-purpose toolbox of solvers that can find Nash equilibria in finite games, although it is often ineffective for even moderately sized games.

To the best of our knowledge, the only solver for infinite games of incomplete information was introduced by (Reeves & Wellman, 2004) (henceforth, RW). In fact, RW is a best response finder, which has successfully been used iteratively to obtain sample Bayes-Nash equilibria for a restricted class of infinite two-player games of incomplete information.

While RW is often effective in converging to a sample Bayes-Nash equilibrium, it does not do so always. Thus, we are presented with the first problem that makes automated mechanism design unique: how do we evaluate the objective if no solution can be obtained? There are a number of ways to approach this difficulty. For example, we can use a uniform distribution over pure strategy profiles in lieu of a Bayes-Nash equilibrium. However, this may not be possible, as the set of pure strategy profiles may be unbounded. Alternatively, since we are dealing with an iterative tool that will always (at least in principle) produce a best response to a given strategy profile, we can use the last best response in the non-converging finite series of iterations as the prediction of agent play. Finally, we may simply constrain the design to discard any choices for which our solver does not produce an answer. Here we employ the last alternative, which is the most conservative of the three.

Since the goal of automated mechanism design is to approximate solutions to design problems with arbitrary objectives and constraints and to handle games with arbitrary type distributions, we treat the probability distribution over player types as a black box from which we can sample joint player types. Thus, we use numerical integration (sample mean in our implementation) to evaluate the expectation of the objective with respect to player types, thereby introducing noise into objective evaluation.

Dealing with Constraints

Mechanism design can feature any of the following three classes of constraints: ex ante (constraints evaluated with respect to the joint distribution of types), ex interim (evaluated separately for each player and type with respect to the joint type distribution of other players), and ex post (evaluated for every joint type profile). When the type space is infinite we of course cannot numerically evaluate any expression for every type. We therefore replace these constraints with probabilistic constraints that must hold for all but a set of types (or joint type profiles) of small measure. For example, an ex post individual rationality (IR) constraint would only have to hold for type profiles that can occur with probability greater than 95%.

Besides a computational justification, there is a practical justification for weakening the requirement that constraints be satisfied for every possible player type (or joint type profile), at least as far as individual rationality is concerned. When we introduce constraints that hold with high probability, we may be excluding a small measure set of types from participation (this is the case for the individual rationality constraints). But by excluding a small portion of types, the expected objective function will change very little, and, similarly, such a change will introduce little incentive for other types to deviate. Indeed, by excluding a subset of types with low valuations for an object, the designer may raise its expected revenue (Krishna, 2002).

Even when we weaken traditional constraints to their probabilistic equivalents, we still need a way to verify that such constraints hold by sampling from the type distribution. Since we can only take a finite number of samples, we will in fact only verify a probabilistic constraint with some level of confidence. The question we want to ask, then, is how many samples do we need in order to say with confidence $1 - \alpha$ that a particular constraint holds for a type set with probability measure at least p? That is the subject of the following theorem.

Theorem 1. *Let B denote a set on which a probabilistic constraint is violated, and suppose that we have a uniform prior over the interval $[0, 1]$ on the probability measure of B. Then, we need at least $\frac{\log \alpha}{\log p} - 1$ samples to verify with confidence $1 - \alpha$ that the constraint holds on a set of types with measure at least p.*

The proofs of this and other results can be found in the appendix of the full version of this paper.

In practice, however, this is not the end of the story for the ex interim constraints. The reason is that the ex interim constraint will take expectation with respect to the joint distribution of types of players other than the player i for which it is verified. Since we must evaluate this expectation numerically, we cannot escape the presence of noise in constraint evaluation. Furthermore, if we are trying to verify the constraint for many type realizations, it is quite likely that in at least one of these instances we will get unlucky and the numerical expectation will violate the constraint, even though the actual expectation does not. We circumvent this problem in two ways. First, we introduce a slight tolerance for a constraint, so that it will not fail due to small evaluation noise. Second, we split the set of types for which the constraint is verified into smaller groups, and throw away a small proportion of types in each group with the worst constraint evaluation result. For example, if we are trying to ascertain that ex interim individual rationality holds, we would throw away several types with the lowest estimated ex interim utility value.

One final general note on constraints: Since constraints are evaluated in our framework as a part of the objective function evaluation process, if a constraint fails a value must still be returned for an objective function. Thus, we set the objective to negative infinity if any constraint fails.

We now describe three specific constraints that we consider in our applications.

Equilibrium Convergence Constraint The purpose of this constraint is to ensure that we have a reliable evalua-

tion of any mechanism that we determine to be optimal. For this to be the case, we must ascertain that it is indeed evaluated with a true equilibrium (or near-equilibrium) strategy profile (given our assumption that a Bayes-Nash equilibrium is a relevant predictor of agent play). For example, best response dynamics using RW need not converge at all. We formally define this constraint as follows:

Definition 1. *Let $s(t)$ be the last strategy profile produced in a sequence of solver iterations, and let $s'(t)$ immediately precede $s(t)$ in this sequence. Then the equilibrium convergence constraint is satisfied if for every joint type profile of players,*

$$|s(t) - s'(t)| < \delta$$

for some a priori fixed tolerance level δ.

The problem that we cannot in practice evaluate this constraint for every joint type profile is resolved by making this constraint probabilistic, as described above. Thus, we define a *p-strong equilibrium convergence constraint*:

Definition 2. *Let $s(t)$ be the last strategy profile produced in a sequence of solver iterations, and let $s'(t)$ immediately precede $s(t)$ in this sequence. Then the p-strong equilibrium convergence constraint is satisfied if for a set of type profiles t with probability measure no less than p,*

$$|s(t) - s'(t)| < \delta$$

for some a priori fixed tolerance level δ.

Ex Interim Individual Rationality This constraint (henceforth, Ex-Interim-IR) specifies that for every agent and for every possible agent's type, that agent's expected utility conditional on its type is greater than its opportunity cost of participating in the mechanism. Formally, it is defined as follows:

Definition 3. *The Ex-Interim-IR constraint is satisfied when for every agent $i \in I$, and for every type $t_i \in T_i$,*

$$E_{t_{-i}} u_i(t, s(t)|t_i) \geq C_i(t_i),$$

where $C_i(t_i)$ is the opportunity cost to agent i with type t_i of participating in the mechanism.

Again, in the automated mechanism design framework, we must change this to a probabilistic constraint as described above.

Definition 4. *The p-strong Ex-Interim-IR constraint is satisfied when for every agent $i \in I$, and for a set of types $t_i \in T_i$ with probability measure no less than p,*

$$E_{t_{-i}} u_i(t, s(t)|t_i) \geq C_i(t_i) - \delta,$$

where $C_i(t_i)$ is the opportunity cost of agent i with type t_i of participating in the mechanism, and δ is some a priori fixed tolerance level.

Commonly in the mechanism design literature the opportunity cost of participation, $C_i(t_i)$, is assumed to be zero but this assumption may not hold, for example, in an auction where not participating would be a give-away to competitors and entail negative utility.

Minimum Revenue Constraint The final constraint that we consider ensures that the designer will obtain some minimal amount of revenue (or bound its loss) in attaining a non-revenue-related objective.

Definition 5. *The minimum revenue constraint is satisfied if*

$$E_t k(s(t), t) \geq C,$$

where $k(s(t), t)$ is the total payment made to the designer by agents with joint strategy $s(t)$ and joint type profile t, and C is the lower bound on revenue.

Applications

In this section we present results from several applications of our automated mechanism design framework to specific two-player problems. One of these problems, finding auctions that yield maximum revenue to the designer, has been studied in a seminal paper by Myerson (1981) in a much more general setting than ours. Another one, which seeks to find auctions that maximize social welfare, has also been studied more generally. Additionally, in several instances we were able to derive optima analytically. For all of these we have a known benchmark to strive for. Others have no known optimal design.

An important consideration in any optimization routine is the choice of a starting point, as it will generally have important implications for the quality of results. This could be especially relevant in practical applications of automated mechanism design, for example, if it is used as a tool to enhance an already working mechanism through parametrized search. Thus, we would already have a reasonable starting point and optimization could be far more effective as a result. We explore this possibility in several of our applications, using a previously studied design as a starting point. Additionally, we apply our framework to every application with completely randomly seeded optimization runs, taking the best result of five randomly seeded runs in order to alleviate the problem posed by local optima. Furthermore, we enhance the optimization procedure by using a *guided* restart, that is, by running the optimization procedure once using the current best mechanism as a new starting point.

In all of our applications player types are independently distributed with uniform distribution on the unit interval. Finally, we used 50 samples from the type distribution to verify Ex-Interim-IR. This gives us 95% confidence that 94% of types lose no more than the opportunity cost plus our specified tolerance which we add to ensure that the presence of noise does not overconstrain the problem. It turns out that every application that we consider produces a mechanism that is individually rational for all types *with respect to the tolerance level that was set*.

Shared-Good Auction (SGA)

Consider the problem of two people trying to decide between two options. Unless both players prefer the same option, no standard voting mechanism (with either straight votes or a ranking of the alternatives) can help with this problem. VCG is a nonstarter with no one to provide a

subsidy and third-party payments are tantamount to a pure efficiency loss.

Instead we propose a simple auction: each player submits a bid and the player with the higher bid wins, paying some function of the bids to the loser in compensation. Reeves (2005) considered a special case of this auction and gave the example of two roommates using it to decide who should get the bigger bedroom and for how much more rent. We sometimes refer to this mechanism as an *un-sharing auction*—it allows one agent to sell its half of a good to the other joint owner (or pay the other to take on its half of a "bad").

We define a space of mechanisms for this problem that are all budget balanced, individually rational, and (assuming monotone strategies) socially efficient. We then search the mechanism space for games that satisfy additional properties. The following is a payoff function defining a space of games parametrized by the function f.

$$u(t,a,t',a') = \begin{cases} t - f(a,a') & \text{if } a > a' \\ \dfrac{t - f(a,a') + f(a',a)}{2} & \text{if } a = a' \\ f(a',a) & \text{if } a < a', \end{cases} \quad (1)$$

where $u()$ gives the utility for an agent who has a value t for winning and chooses to bid a against an agent who has value t' and bids a'. The t's are the agents' types and the a's their actions. Finally, $f()$ is some function of the two bids.[4] In the tie-breaking case (which occurs with probability zero for many classes of strategies) the payoff is the average of the two other cases, i.e., the winner is chosen by the flip of a fair coin.

We now consider a restriction of the class of mechanisms defined above.

Definition 6. $\mathrm{SGA}(h,k)$ *is the mechanism defined by Equation 1 with* $f(a,a') = ha + ka'$.

For example, in $\mathrm{SGA}(1/2,0)$ the winner pays half its own bid to the loser; in $\mathrm{SGA}(0,1)$ the winner pays the loser's bid to the loser. We now give Bayes-Nash equilibria for such games when types are uniform.

Theorem 2. *For* $h,k \geq 0$ *and types* $U[A,B]$ *with* $B \geq A+1$ *the following is a symmetric Bayes-Nash equilibrium of* $\mathrm{SGA}(h,k)$*:*

$$s(t) = \frac{t}{3(h+k)} + \frac{hA+kB}{6(h+k)^2}$$

For the following discussion, we need to define the notion of truthfulness, or Bayes-Nash incentive compatibility.

Definition 7 (BNIC)**.** *A mechanism is Bayes-Nash incentive compatible (truthful) if bidding* $s(t) = t$ *constitutes a Bayes-Nash equilibrium of the game induced by the mechanism.*

The Revelation Principle (Mas-Colell, Whinston, & Green, 1995) guarantees that for any mechanism, involving arbitrarily complicated sequences of messages between participants, there exists a BNIC mechanism that is equivalent in terms of how it maps preferences to outcomes. This is demonstrated by construction: consider a meta-mechanism that consists of the original mechanism with proxies inserted that take reported preferences from the agents and play a Nash equilibrium on the agents' behalf in the original game.

We can now characterize the truthful mechanisms in this space. According to Theorem 2, $\mathrm{SGA}(1/3,0)$ is truthful for $U[0,B]$ types. We now show that this is the *only* truthful design in this design space.

Theorem 3. *With* $U[0,B]$ *types (*$B > 0$*),* $\mathrm{SGA}(h,k)$ *is BNIC if and only if* $h = 1/3$ *and* $k = 0$.

Of course, by the revelation principle, it is straightforward to construct a mechanism that is BNIC for any $U[A,B]$ types. However, to be a proper auction (Krishna, 2002) the mechanism should not depend on the types of the participants. In other words, the mechanism should not be parametrized by A and B. With this restriction, the revelation principle fails to yield a BNIC mechanism for arbitrary uniform types. Below, we will show concrete examples of the failure of the revelation principle for several sensible designer objectives.

From now on we will restrict ourselves to the case of $U[0,1]$ types. Since $\mathrm{SGA}(1/3,0)$ is the only truthful mechanism in our design space, we can directly compare the objective value obtained from this mechanism and the best indirect mechanism in the sections that follow.

Minimize Difference in Expected Utility First, we consider as our objective *fairness*, or negative differences between the expected utility of winner and loser. Alternatively, our goal is to minimize

$$\begin{aligned} &|E_{t,t'}[u(winner) - u(loser)]| = \\ &|E_{t,t'}[u(t,s(t),t',s(t'),k,h \mid a > a') - \\ &\quad - u(t,s(t),t',s(t'),k,h \mid a < a')]|. \end{aligned} \quad (2)$$

We first use the equilibrium bid derived above to analytically characterize optimal mechanisms.

Theorem 4. *The objective value in (2) for* $\mathrm{SGA}(h,k)$ *is*

$$\frac{2h+k}{9(h+k)}.$$

Furthermore, $\mathrm{SGA}(0,k)$*, for any* $k > 0$*, minimizes the objective, and the optimum is 1/9.*

By comparison, the objective value for the truthful mechanism, $\mathrm{SGA}(1/3,0)$, is 2/9, twice as high as the minimum produced by an untruthful mechanism. Thus, the revelation principle does not hold for this objective function in our design space. We can use Theorem 4 to find that the objective value for $\mathrm{SGA}(1/2,0)$, the mechanism described by Reeves (2005), is 2/9.

Now, to test our framework, we imagine we don't know about the above analytic derivations (including the derivation of the Bayes-Nash equilibrium) and run the automated mechanism design procedure in black-box mode. Table 1 presents results when we start the search at random values of h and k (taking the best outcome from 5 random restarts), and at the starting values of $h = 0.5$ and $k = 0$. Since the

[4] Reeves (2005) considered the case $f(a,a') = a/2$.

Parameters	Initial Design	Approx. Optimal Design
h	0.5	0
k	0	1
objective	2/9	1/9
h	random	0
k	random	1
objective	N/A	1/9

Table 1: Design that approximately maximizes fairness (minimizes difference in expected utility between utility of winner and loser) when the optimization search starts at a fixed starting point, and the best mechanism from five random restarts.

objective function turns out to be fairly simple, it is not surprising that we obtain the optimal mechanism for specific and random starting points (indeed, the optimal design was produced from every random starting point we generated).

Minimize Expected (Ex-Ante) Difference in Utility Here we modify the objective function slightly as compared to the previous section, and instead aim to minimize the expected ex ante difference in utility:

$$\begin{aligned} &E|u(winner) - u(loser)| = \\ &E|u(t, s(t), t', s(t'), k, h|a > a') \\ &\quad - u(u(t, s(t), t', s(t'), k, h|a < a')|. \end{aligned} \tag{3}$$

While the only difference from the previous section is the placement of the absolute value sign inside the expectation, this difference complicates the analytic derivation of the optimal design considerably. Therefore, we do not present the actual optimum design values.

Parameters	Initial Design	Approx. Optimal Design
h	0.5	0.49
k	0	1
objective	0.22	0.176
h	random	0.29
k	random	0.83
objective	N/A	0.176

Table 2: Design that approximately minimizes expected ex ante difference between utility of winner and loser when the optimization search starts at a random and a fixed starting points.

The results of application of our AMD framework are presented in Table 2. While the objective function in this example appears somewhat complex, it turns out (as we discovered through additional exploration) that there are many mechanisms that yield nearly optimal objective values.[5] Thus, both random restarts as well as a fixed starting point produced essentially the same near-optima. By comparison, the truthful design yields the objective value of about 0.22, which is considerably worse.

Maximize Expected Utility of the Winner Yet another objective in the shared-good-auction domain is to maximize the expected utility of the winner.[6] Formally, the designer is maximizing $E[u(t, s(t), t', s(t'), k, h \mid a > a')]$.

We first analytically derive the characterization of optimal mechanisms.

Theorem 5. *The problem is equivalent to finding h and k that maximize $4/9 - \frac{k}{18(h+k)}$. Thus, $k = 0$ and $h > 0$ maximize the objective, and the optimum is $4/9$.*

Parameters	Initial Design	Approx. Optimal Design
h	0.5	0.21
k	0	0
objective	4/9	4/9
h	random	0.91
k	random	0.03
objective	N/A	0.443

Table 3: Design that approximately maximizes the winner's expected utility.

Here again our results in Table 3 are optimal or very nearly optimal, unsurprisingly for this relatively simple application.

Maximize Expected Utility of the Loser Finally, we try to maximize the expected utility of the loser.[7] Formally, the designer is maximizing

$$E[u(t, s(t), t', s(t'), k, h|a < a')].$$

We again first analytically derive the optimal mechanism.

Theorem 6. *The problem is equivalent to finding h and k that maximize $2/9 + \frac{k}{18(h+k)}$. Thus, $h = 0$ and $k > 0$ maximize the objective, and the optimum is $5/18$.*

Table 4 shows the results of running AMD in black-box mode in this setting. We can observe that our results are again either actually optimal when the search used a fixed starting point, or close to optimal when random starting points were used. While this design problem is relatively easy and the answer can be analytically derived, the objective function is non-linear, which, along with the presence of noise, adds sufficient complexity to blind optimization to suggest that our success here is at least somewhat interesting.

[5] Particularly, we carried out a far more intensive exploration of the search space given the analytic expression for the Bayes-Nash equilibrium to ascertain that the values reported are close to actual optima. Indeed, we failed to improve on these.

[6] For example, the designer may be interested in minimizing the amount of money which changes hands, which is, by construction, an equivalent problem.

[7] For example, the designer may be interested in maximizing the amount of money which changes hands, which is, by construction, an equivalent problem.

Parameters	Initial Design	Approx. Optimal Design
h	0.5	0
k	0	0.4
objective	2/9	5/18
h	random	0.13
k	random	1
objective	N/A	0.271

Table 4: Design that approximately maximizes the loser's expected utility.

Of the two examples we considered so far, the first turned out to be analytic and only the second we could only approach numerically. Nevertheless, even in the analytic cases, the objective function forms were not trivial, particularly from a blind optimization perspective. Furthermore, we must take into account that even the simple cases are somewhat complicated by the presence of noise, and thus we need not arrive at global optima even in the simplest of settings so long as the number of samples is not very large.

Having found success in the simple shared-good auction setting, we now turn our attention to a series of considerably more difficult problems.

Myerson Auctions

The seminal paper by Myerson (1981) presented a theoretical derivation of revenue maximizing auctions in a relatively general setting. Here, our aim is to find a mechanism with a nearly-optimal value of some given objective function, of which revenue is an example. However, we restrict ourselves to a considerably less general setting than did Myerson, constraining our design space to that described by the parameters $q, k_1, k_2, K_1, k_3, k_4, K_2$ in (4).

$$u(t,a,t',a') = \begin{cases} U_1 & \text{if } a > a' \\ 0.5(U_1+U_2) & \text{if } a = a' \\ U_2 & \text{if } a < a', \end{cases} \quad (4)$$

where $U_1 = qt - k_1 a - k_2 a' - K_1$ and $U_2 = (1-q)t - k_3 a - k_4 a' - K_2$. We further constrain all the design parameters to be in the interval [0,1]. In standard terminology, our design space allows the designer to choose an allocation parameter, q, which determines the probability that the winner (i.e., agent with the winning bid) gets the good, and transfers, which we constrain to be linear in agents' bids.

While our automated mechanism design framework assures us that p-strong individual rationality will hold with the desired confidence, we can actually verify it by hand in this application. Furthermore, we can adjust the mechanism to account for lapses in individual rationality guarantees for subsets of agent types by giving to each agent the amount of the expected loss of the least fortunate type.[8] Similarly, if we do find a mechanism that is Ex-Interim-IR, we may still have an opportunity to increase expected revenue as long as the minimum expected gain of any type is strictly greater than zero.[9]

Maximize Revenue In this section, we are interested in finding approximately revenue-maximizing designs in our constrained design space. Based on Myerson's feasibility constraints, we derive in the following theorem that an optimal incentive compatible mechanism in our design space yields revenue of 1/3 to the designer,[10] as compared to 0.425 in the general two-player case.[11]

Lemma 7. *The mechanism in the design space described by the parameters in equation 4 is BNIC and Ex-Interim-IR if and only if* $k_3 = k_4 = K_1 = K_2 = 0$ *and* $q - k_1 - 0.5k_2 = 0.5$.

Theorem 8. *Optimal incentive compatible mechanism in our setting yields the revenue of 1/3, which can be achieved by selecting* $k_1 \in [0, 0.5]$ *and* $k_2 \in [0, 1]$*, respecting the constraint that* $k_1 + 0.5k_2 = 0.5$.

Parameters	Initial Design	Approx. Optimal Design
q	random	0.96
k_1	random	0.95
k_2	random	0.84
K_1	random	0.78
k_3	random	0.73
k_4	random	0
K_2	random	0.53
objective	N/A	0.3

Table 5: Design that approximately maximizes the designer's revenue.

In addition to performing five restarts from random starting points, we repeated the simulated annealing procedure starting with the best design produced via the random restarts. This procedure yielded the design in Table 5. We used the RW solver to find a symmetric equilibrium of this design, under which the bids are $s(t) = 0.72t - 0.73$.[12] We now verify the Ex-Interim-IR and revenue properties of this design.

Proposition 9. *The design described in Table 5 is Ex-Interim-IR and yields the expected revenue of approximately 0.3. Furthermore, the designer could gain an additional 0.0058 in expected revenue without effect on incentives while maintaining the individual rationality constraint.*

We have already shown that the best known design, which is also the optimal incentive compatible mechanism in this

[8] Observe that such constant transfers will not affect agent incentives.

[9] We do not explore computational techniques for either fixing individual rationality or exploiting strictly positive player surplus in this work. We imagine, however, that given a working design many approaches that perform extensive exploration of player surplus in the players' type spaces could be adequate.

[10] For example, Vickrey auction will yield this revenue.

[11] The optimal mechanism prescribed by Myerson is not implementable in our design space.

[12] This equilibrium is approximate in the sense that we rounded the parameters to the nearest hundredth.

setting, yields a revenue of 1/3 to the designer. Thus, our AMD framework produced a design near to the best known. It is an open question what the actual global optimum is.

Maximize Welfare It is well known that the Vickrey auction is welfare-optimal. Thus, we know that the welfare optimum is attainable in our design space. Before proceeding with search, however, we must make one observation. While we are interested in welfare, it would be inadvisable in general to completely ignore the designer's revenue, since the designer is unlikely to be persuaded to run a mechanism at a disproportionate loss. To illustrate, take the same Vickrey auction, but afford each agent one billion dollars for participating. This mechanism is still welfare-optimal, but seems a senseless waste if optimality could be achieved without such spending (and, indeed, at some profit to the auctioneer). To remedy this problem, we use a minimum revenue constraint, ensuring that no mechanism that is too costly will be selected as optimal.

First, we present a general result that characterizes welfare-optimal mechanisms in our setting.

Theorem 10. *Welfare is maximized if either the equilibrium bid function is strictly increasing and $q = 1$ or the equilibrium bid function is strictly decreasing and $q = 0$. Furthermore, the maximum expected welfare in our design space is 2/3.*

Thus, for example, both first- and second-price sealed bid auctions are welfare-optimizing (as is well known).

In Table 6 we present the result of our search for optimal design with 5 random restarts, followed by another run of simulated annealing that uses the best outcome of 5 restarts as the starting point. We verified using the RW solver that

Parameters	Initial Design	Approx. Optimal Design
q	random	1
k_1	random	0.88
k_2	random	0.23
K_1	random	0.28
k_3	random	0.06
k_4	random	0.32
K_2	random	0
objective	N/A	2/3

Table 6: Design that approximately maximizes welfare.

the bid function $s(t) = 0.645t - 0.44$ is an equilibrium given this design. Since it is strictly increasing in t, we can conclude based on Theorem 10 that this design is welfare-optimal. We only need to verify then that both the minimum revenue and the individual rationality constraints hold.

Proposition 11. *The design described in Table 6 is Ex-Interim-IR, welfare optimal, and yields the revenue of approximately 0.2. Furthermore, the designer could gain an additional 0.128 in revenue (for a total of about 0.33) without affecting agent incentives or compromising individual rationality and optimality.*

It is interesting that this auction, besides being welfare-optimal, also yields a slightly higher revenue to the designer than our mechanism in the previous section if we implement the modification proposed in Proposition 11. Thus, there appears to be some synergy between optimal welfare and optimal revenue in our design setting.

Conclusion

We presented a framework for automated indirect mechanism design using the Bayes-Nash equilibrium solver for infinite games developed by Reeves & Wellman (2004); Reeves (2005). Results from applying this framework to several design domains demonstrate the value of our approach for practical mechanism design. The mechanisms that we found were typically either close to the best known mechanisms, or better.

While in principle it is not at all surprising that we can find mechanisms by searching the design space—as long as we have an equilibrium finding tool—it was not at all clear that any such system will have practical merit. We presented evidence that indirect mechanism design can indeed be effectively automated on somewhat realistic design problems. Undoubtedly, real design problems are vastly more complicated than any that we considered (or any that can be considered theoretically). In such cases, we believe that our approach could offer considerable benefit if used in conjunction with other techniques, either to provide a starting point for design, or to tune a mechanism produced via theoretical analysis and computational experiments.

References

Brandt, F., and Weiß, G. 2001. Antisocial agents and Vickrey auctions. In *Eighth International Workshop on Agent Theories, Architectures, and Languages*, volume 2333 of *Lecture Notes in Computer Science*, 335–347. Seattle: Springer.

Brandt, F.; Sandholm, T.; and Shoham, Y. 2005. Spiteful bidding in sealed-bid auctions. Technical report, Stanford University.

Conitzer, V., and Sandholm, T. 2003a. Applications of automated mechanism design. In *UAI-03 Bayesian Modeling Applications Workshop*.

Conitzer, V., and Sandholm, T. 2003b. Computational criticisms of the revelation principle. In *Workshop on Agent Mediated Electronic Commerce-V*.

Conitzer, V., and Sandholm, T. 2004. An algorithm for automatically designing deterministic mechanisms without payments. In *Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 128–135.

Corana, A.; Marchesi, M.; Martini, C.; and Ridella, S. 1987. Minimizing multimodal functions of continuous variables with simulated annealing algorithm. *ACM Transactions on Mathematical Software* 13(3):262–280.

Fleischer, M. 1995. Simulated Annealing: Past, present, and future. In *Winter Simulation Conference*, 155–161.

Klemperer, P. 2004. *Auctions: Theory and Practice*. Princeton University Press.

Krishna, V. 2002. *Auction Theory*. Academic Press, 1st edition.

Mas-Colell, A.; Whinston, M. D.; and Green, J. R. 1995. *Microeconomic Theory*. New York: Oxford University Press.

McKelvey, R. D.; McLennan, A. M.; and Turocy, T. L. 2005. Gambit: Software tools for game theory, version 0.2005.06.13.

McMillan, J. 1994. Selling spectrum rights. *The Journal of Economic Perspectives* 8(3):145–162.

Morgan, J.; Steiglitz, K.; and Reis, G. 2003. The spite motive and equilibrium behavior in auctions. *Contributions to Economic Analysis and Policy* 2(1).

Myerson, R. B. 1981. Optimal auction design. *Mathematics of Operations Research* 6(1):58–73.

Olafsson, S., and Kim, J. 2002. Simulation optimization. In Yucesan, E.; Chen, C.-H.; Snowdon, J.; and Charnes, J., eds., *2002 Winter Simulation Conference*.

Reeves, D. M., and Wellman, M. P. 2004. Computing best-response strategies in infinite games of incomplete information. In *Twentieth Conference on Uncertainty in Artificial Intelligence*, 470–478.

Reeves, D. M. 2005. *Generating Trading Agent Strategies: Analytic and Empirical Methods for Infinite and Large Games*. Ph.D. Dissertation, University of Michigan.

Sandholm, T.; Conitzer, V.; and Boutilier, C. 2007. Automated design of multistage mechanisms. In *International Joint Conference on Artificial Intelligence*.

Siarry, P.; Berthiau, G.; Durbin, F.; and Haussy, J. 1997. Enhanced simulated annealing for globally minimizing functions of many continuous variables. *ACM Transactions on Mathematical Software* 23(2):209–228.

Vorobeychik, Y.; Kiekintveld, C.; and Wellman, M. P. 2006. Empirical mechanism design: Methods, with an application to a supply chain scenario. In *ACM E-Commerce*, 306–315.