Searching for Stable Mechanisms : Automated Design for Imperfect Players

Andrew J. Blumberg
Department of Mathematics
University of Chicago
blumberg@math.uchicago.edu

abhi shelat
CSAIL
Massachusetts Institute of Technology
abhi@mit.edu

Abstract

Recently Conitzer and Sandholm (Conitzer & Sandholm 2002) introduced the concept of "automated mechanism design", whereby mechanism design problems are solved using constraint-satisfaction methods.

Traditionally, mechanism design has focused on producing games which yield the desired outcomes when played by ideal rational players. However actual players are never perfectly rational — human irrationality has been exhaustively studied and computational agents have both resource bounds and potentially implementation flaws. In this paper, we discuss extensions of the techniques of automated mechanism design to produce games which are robust in the face of player imperfections.

We model limited rationality by examining agents which converge on their strategy by using a simple variant of "fictitious play" (simulation of repeated play) (Singh, Kearns, & Mansour 2000). This model associates to each game a system of differential equations describing the trajectory of the agent's strategies. We describe additional constraints which guarantee that automated mechanism design search problems yield *stable* mechanisms. In particular, we present negative results for structural stability and positive results for asymptotic stability by considering strict Bayesian-Nash equilibria and by employing Lyapunov techniques.

Introduction

Mechanism design is the problem of designing the rules of a game such that individually selfish behavior by the players leads to a "globally optimal" outcome. Mechanism design is relevant whenever there are preferences to be aggregated — examples of such situations include auctions, elections, and public goods problems. Classical approaches to mechanism design in the economics literature focused on the construction of general mechanisms which would function for a wide class of possible preference relations for the players. For example, a seminal result due to Maskin (Osborne & Rubinstein 1994) demonstrates by explicit construction that the mechanism design problem is always solvable (in undominated Nash equilibria) when there are more than two players and the player's preferences satisfy certain fairly general conditions (e.g. monotonicity). Other such results include the celebrated Vickrey-Clark-Groves second-price auctions.

However, the classical approach is unsatisfactory on a number of grounds. For one thing, impossibility results such as the Gibbard-Satterthwaite theorem state that such general constructions cannot be obtained in many important areas of mechanism design. Moreover, the mechanisms used in the proofs of these results are extremely intricate and sometimes perverse (for example, many employ tie-breaking "integer games" which require the players to declare unbounded integers (Osborne & Rubinstein 1994)). To circumvent these problems, Conitzer and Sandholm (Conitzer & Sandholm 2002) introduced the notion of automated mechanism design. Their key insight is that in a particular instance, the preferences of the players are known and so the mechanism can be designed with these specific preferences in mind. This observation enables one to translate the problem of designing a mechanism to a numerical programming problem which can be solved using standard techniques.

Viewing mechanism design problems as constraint-satisfaction problems creates an opportunity to add more constraints to search for games with additional structure. Our work herein demonstrates that it is now possible to reasonably perform mechanism design for games with nuanced and complicated solution concepts. It is hard to imagine such problems being tractable from a general perspective.

The formulation of the traditional mechanism design problem presumes that the players are perfectly rational and error-free. This is an idealization which is simply false in practice. For example, human players are known to consistently deviate from ideal rationality in a complicated collection of ways. In the electronic setting, computational agents are likely to have limited computational power and potentially flawed algorithms or implementations. It is therefore desirable to attempt to construct mechanisms which are robust in the face of player failure. Larson and Sandholm considered a model of bounded computational resources for players which they call "strategic computation", in which players rationally seek to deploy their limited computation (Larson & Sandholm 2001). We study a different approach to the problem.

Repeated play is a common model of learning and bounded rationality. In such a setting, players are assumed to learn to play the game by repeatedly playing and improving their strategies between rounds. It may be the case that the game is actually played many times, or the players may

be individually simulating the repetition of the game in order to converge on a strategy to play in a one-shot game (this is called "fictitious play"). Such learning processes can be modeled by a system of differential equations describing the updates of the strategy profiles of the players. We study the stability of a potential mechanism by examining the stability of this associated system of differential equations. A robust mechanism is one for which the associated system is stable and robust in the face of noise both in the description of the game and in the specification of the strategies.

In the following sections, we explore several notions of stability. We examine structural stability (the robustness of qualitative behavior of the trajectories in the face of perturbation of the system of equations) and prove that for some classes of games, automated mechanism design will never yield structurally stable solutions. In general, we discuss how to determine if the associated system is structurally stable. Then we consider asymptotic stability of equilibrium points (the robustness of an equilibrium in the face of small perturbations of the state of a system). We describe how to write constraints which guarantee that solutions to the automated mechanism design search problem have asymptotically stable equilibria by searching for Lyapunov functions (Merkin 1997).

Background

A central solution concept in games of incomplete information is the Bayesian-Nash equilibrium, which we review here (see (Fudenberg & Tirole 1991) for a detailed presentation). In a Bayesian-Nash setting, each of the n players in a game know their own *type* θ_i (private information), but only have a probability distribution over the types of the other players. Each player also has a set of actions, A_i. An *outcome*, or action profile, is the tuple of actions chosen by all the players and is a member of $A = A_1 \times A_2 \times \cdots \times A_n$. Each player also has a utility function which maps their type and an outcome of the game to a particular payoff.

(Informal) Bayesian-Nash Equilibrium A set of strategies is a Bayesian-Nash equilibrium if for each player, and for each possible type of that player, the expected utility of the player's strategy (given the distribution of other players' types and that their strategies are fixed) is greater than or equal to the expected utility of any other strategy for that player.

Automated mechanism design

Now we review the formal setting for automated mechanism design problems.

Definition In an automated mechanism design setting, we are given :

1. A finite set of players N.
2. A finite set of outcomes O.
3. For each player i, a set of types θ_i, and a probability distribution over θ_i.
4. For each player i, a utility function $u_i : \theta_i \times O \rightarrow \mathbb{R}$ which maps types and outcomes to utilities.
5. A social objective function whose expectation the designer is to maximize.
6. A solution concept (e.g. Bayesian-Nash equilibrium).
7. Additional constraints — for example, whether the mechanism can employ randomness or side-payments.

The goal of a mechanism design problem is to produce a game G which is a map $G : S_1 \times S_2 \times \ldots \times S_n \rightarrow \chi(O)$, where $\chi(O)$ is the space of probability distribution over O and each S_i is a set of actions for player i to play. When randomized outcomes are not allowed, χ is restricted to distributions with all of their weight on a single outcome. Note that the set of pure strategies for each player in this game are the same as the set of types– in other words, in the game, each player is to announce a type, and the collective actions of all players determines the outcome.

The mechanism G is a *truth-telling* mechanism if the set of strategies in which all players announce their types honestly forms an equilibrium with respect to the solution concept.

We assume in the remainder of the paper that the solution concept is Bayesian-Nash equilibrium, randomized outcomes are permitted, and the utility function is the sum of expected utilities for all players. Conitzer and Sandholm (Conitzer & Sandholm 2002) present a procedure for expressing mechanism design problems of this type as linear programming problems.

LP For Mechanism Design Let $S = S_1 \times S_2 \times \cdots \times S_n$ represent the action profiles, which in this case coincides with the set of types, $\Theta = \theta_1 \times \cdots \times \theta_n$. The variables, $\{x_{s,o}\}$, are indexed on $S \times O$ and represent the probability that a particular outcome o occurs given a specified action profile $s = (s_1, \ldots, s_n)$. The objective function of the program is the social welfare function (the sum of the players' expected utilities). The constraints of the program are:

1. For each action profile $s \in S$, the probabilities associated with that action profile must sum to 1.
$$\forall_{s \in S} \sum_{o \in O} x_{s,o} = 1$$
2. For each player i, the expected utility for the truth-telling strategy must be greater than or equal to 0. This specifies that there is nonnegative benefit to the players for participating.
$$\forall_{s \in S} \sum_{o \in O} x_{s,o} u_i(s_i, o) \geq 0$$
Note that in the truth-telling strategy, player i's utility at a particular outcome is simply $u_i(s_i, o)$ since s_i is her true type.
3. For a given player, the expected utility for the truth-telling strategy (when all other players are playing the truth-telling strategy) must be greater than or equal to the expected utility for any other pure strategy. This guarantees that truth-telling is a Bayesian-Nash equilibrium.
$$\forall_{s \in S} \forall_{\sigma \in S_i} \sum_{o \in O} x_{s,o} u_i(s_i, o) \geq \sum_{o \in O} x_{s',o} u_i(\sigma, o)$$

(where $s'_j = s_j$ for $j \neq i$ and $s'_i = \sigma$)

Remark If we forbid randomized outcomes, this program becomes a mixed-integer programming problem with linear constraints. Mixed integer programs are still solvable (although not as efficiently), but in some subsequent constructions we will generate polynomial constraints. Mixed-integer programs with polynomial constraints are undecidable, and so we focus on randomized outcomes which lead to general nonlinear programming problems.

An example

We employ the following divorce settlement example, also discussed in (Conitzer & Sandholm 2003), to illustrate our techniques and methodology.

Divorce Settlement. A husband and a wife are getting divorced. An arbitrator must allocate a prized painting.

1. The husband and the wife are each either of type low (L) or high (H) with respect to their interest in the painting.
2. The possible outcomes are to give the painting to the husband (h), to give it to the wife (w), joint ownership (j), or to burn the painting (b).
3. The utilities, which are the same for each player, are as follows: $u_L(b) = -10$, $u_L(\text{other has}) = 0$, $u_L(\text{own}) = 2$, $u_(j) = 1$, $u_H(b) = -10$, $u_H(\text{other has}) = 0$, $u_H(\text{own}) = 100$, $u_H(j) = 50$.
4. Each person is type L with probability 0.8 and type H with probability 0.2.

In our linear program formulation , the variable x^a_{yz} represent the probability that the outcome is a given that husband and wife announce types y and z respectively. Solving the this linear program as described in yields the following mechanism:

♀/♂	L	H
L	$h = w = j = \frac{1}{3}$	$h = 1$
H	$w = 1$	$h = w = j = \frac{2}{11}$, $b = \frac{5}{11}$

Figure 1: Divorce Settlement Mechanism. The Husband's actions are across the columns, and the Wife's actions are on the rows. The expected utility for both players is $17\frac{302}{550}$.

Note that these numbers are different then the solution presented in (Conitzer & Sandholm 2003) — there are many optimal solutions. Interestingly, the solution in (Conitzer & Sandholm 2003) does not have symmetric utilities: the husband's utility is $18\frac{8}{11}$ and the wife's utility is $16\frac{102}{275}$.

The dynamical system induced by repeated play

We consider the simple repeated-play gradient-ascent dynamics discussed in (Singh, Kearns, & Mansour 2000). In this model, each player updates their strategy between simulations of the games by adjusting the probabilities of action selection along the gradient of their expected utility. A brief description of the framework follows.

The strategy profile for player i is represented as a collection of variables $\{\alpha_{i,j}\}$ indexed on all but one of the possible actions in S_i, where $\alpha_{i,j}$ is the probability of playing action j. The probability of the deleted action is implicitly $1-\sum_j \alpha_{i,j}$ in order to ensure that the probabilities sum to 1. In our case, the set of pure strategies for player i corresponds to the set of types $S_i = \theta_i = \{\theta_i^1, \ldots, \theta_i^k\}$, and so $\alpha_{i,j}$ is the probability that player i announces type θ_i^j. Define the gradient vector for player i as $\hat{\alpha}_i = (\dot{\alpha}_{i,1}, \ldots, \dot{\alpha}_{i,k})$ where

$$\dot{\alpha}_{i,j} = \frac{\partial E[u_i]}{\partial \alpha_{i,j}}$$

The gradient-ascent method begins at some initial strategy profile $s \in S$ and repeatedly updates the strategies by taking a small step along $\hat{\alpha}$. To extend the dynamics to the Bayesian-Nash case, we compute the expected utility assuming that both a player's own type and the types of the other players are drawn from given distributions — we assume the player does not use knowledge about their own type as they simulate. This is reasonable in some circumstances (a player simulating their auction behavior might well assume that their valuation of an item will fluctuate) and unreasonable in others. See also (Dekel, Fudenberg, & Levine 2001) for a discussion of related concerns.

In the same manner as (Singh, Kearns, & Mansour 2000), we consider the modified dynamics in which steps which go outside the boundaries of the unit cube are projected back into the cube. This has the effect that equilibrium points of the modified dynamics may not be equilibrium points of the unconstrained dynamics. As such, we have to use modified versions of the analytic tools for stability determination. See (Blumberg & shelat 2004) for an analysis of this situation.

In addition, it should be noted that the system of differential equations we describe is obtained as the limit for very small step size of a discrete update process. However, it can be shown that as the step size decreases the dynamics approach the "continuous dynamics" — we defer further discussion of this to the full paper.

In order to illustrate the gradient-descent repeated-play dynamics for our running example, we first compute the expected utility for the husband, which we denote V. Here α_1 is the probability that the husband plays low when his type is low, α_2 is the probability he plays low when his type is high, and the same applies to α_3 and α_4 for the wife. Also, u^x_{yz} is the husband's expected payoff when the his type is x and the actions y and z are played (that is, it is $\sum_a x^a_{yz} u_y(a)$). For notational convenience, let $\overline{x}$ represent $(1-x)$.

$$V =$$
$$(t_1 t_2)(\alpha_1\alpha_3 u^L_{LL} + \alpha_1\overline{\alpha_3} u^L_{LH} + \overline{\alpha_1}\alpha_3 u^L_{HL} + \overline{\alpha_1\alpha_3} u^L_{HH}) +$$
$$(\overline{t_1} t_2)(\alpha_2\alpha_3 u^H_{LL} + \alpha_2\overline{\alpha_3} u^H_{LH} + \overline{\alpha_2}\alpha_3 u^H_{HL} + \overline{\alpha_2\alpha_3} u^H_{HH}) +$$
$$(t_1 \overline{t_2})(\alpha_1\alpha_4 u^L_{LL} + \alpha_1\overline{\alpha_4} u^L_{LH} + \overline{\alpha_1}\alpha_4 u^L_{HL} + \overline{\alpha_1\alpha_4} u^L_{HH}) +$$
$$(\overline{t_1 t_2})(\alpha_2\alpha_4 u^H_{LL} + \alpha_2\overline{\alpha_4} u^H_{LH} + \overline{\alpha_2}\alpha_4 u^H_{HL} + \overline{\alpha_2\alpha_4} u^H_{HH})$$

where $t_1 = t_2 = 0.8$. The gradient is obtained by differentiating this expression :

$$\frac{\partial V}{\partial \alpha_1} = t_1 t_2(\alpha_3 u^L_{LL} + \overline{\alpha_3} u^L_{LH} - \alpha_3 u^L_{HL} - \overline{\alpha_3} u^L_{HH}) + t_1 \overline{t_2}(\alpha_4 u^L_{LL} + \overline{\alpha_4} u^L_{LH} - \alpha_4 u^L_{HL} - \overline{\alpha_4} u^L_{HH})$$

(the other derivatives are similar and omitted).

Searching for stability

There are many ways in which the dynamical system defined above might be unstable. For example, the system might never converge to any Nash equilibrium, no matter what the initial conditions are. Or the system might have many Nash equilibria in addition to the desired one, and each might have large basins of convergence.

Remark One difficulty with automated mechanism design is the inability to prevent the existence of extraneous equilibria in addition to the desired solution. This is in contrast to the mechanisms obtained from the classical general theorems, which do not have any additional equilibria (with respect to the solution concept) — although it should be noted that much of the unpleasantness of the results obtained there (e.g. "integer games") arises from the need to rule out unwanted equilibria. Since "generic" games have large numbers of Nash equilibria (McLennan 1999), this is an issue worthy of future work.

We are particularly concerned with two types of stability under perturbation, *structural stability* and *asymptotic stability*.

Structural stability

A system of differential equations is structurally stable if given a specific trajectory, small changes in the coefficients of the system produce a system with a "nearby" trajectory to the original (see (Hirsch & Smale 1974) for details). Equilibrium points near which the system is structurally stable are called *hyperbolic*. A necessary condition for structural stability is that all equilibrium points are hyperbolic. This can be checked by inspecting the Jacobian matrix of the system of differential equations — if there are no eigenvalues with zero real parts, then the point is structurally stable (Hirsch & Smale 1974).

Unfortunately, our running example never has structurally stable equilibria. In fact, we have the following general result.

Lemma 1 *The Bayesian-Nash gradient dynamics for games with two players and two types is never structurally stable. In particular, the divorce settlement mechanism is always structurally unstable.*

Proof Sketch Consider the Jacobian for the dynamics:

$$\begin{pmatrix} 0 & 0 & \frac{\partial \dot{\alpha_1}}{\partial \alpha_3} & \frac{\partial \dot{\alpha_1}}{\partial \alpha_4} \\ 0 & 0 & \frac{\partial \dot{\alpha_2}}{\partial \alpha_3} & \frac{\partial \dot{\alpha_2}}{\partial \alpha_3} \\ \frac{\partial \dot{\alpha_3}}{\partial \alpha_1} & \frac{\partial \dot{\alpha_3}}{\partial \alpha_2} & 0 & 0 \\ \frac{\partial \dot{\alpha_4}}{\partial \alpha_1} & \frac{\partial \dot{\alpha_4}}{\partial \alpha_2} & 0 & 0 \end{pmatrix}$$

Computing, we find that the the 2×2 submatrix in the top right corner is

$$\begin{aligned} \frac{\partial \dot{\alpha_1}}{\partial \alpha_3} &= t_1 t_2(u^L_{LL} - u^L_{LH} - u^L_{HL} + u^L_{HH}) \\ \frac{\partial \dot{\alpha_1}}{\partial \alpha_4} &= t_1 \overline{t_2}(u^L_{LL} - u^L_{LH} - u^L_{HL} + u^L_{HH}) \\ \frac{\partial \dot{\alpha_2}}{\partial \alpha_3} &= \overline{t_1} t_2(u^H_{LL} - u^H_{LH} - u^H_{HL} + u^H_{HH}) \\ \frac{\partial \dot{\alpha_2}}{\partial \alpha_4} &= \overline{t_1 t_2}(u^H_{LL} - u^H_{LH} - u^H_{HL} + u^H_{HH}) \end{aligned}$$

Hence, the two top rows are linearly dependent and so this submatrix is singular. Similarly, we obtain that the bottom two rows are linearly dependent, which implies that there are two nonzero eigenvectors with eigenvalue 0 (one of the form $(0\ 0\ x\ y)^T$ and the other of the form $(z\ w\ 0\ 0)^T$, where (x, y) is a nonzero solution to the singular 2×2 system in the upper righthand corner and (z, w) is a nonzero solution to the singular 2×2 system in the lower lefthand corner). □

This result is not surprising as a little bit of experimentation makes it clear that the dynamics are sensitive to small changes in the payoff matrix. Larger games are not necessarily structurally unstable, and in a forthcoming paper we determine more precisely the conditions for the Bayesian-Nash gradient dynamics to be structurally stable in general. Unfortunately, searching for structural stability is hard — writing constraints that prevent the Jacobian from having eigenvalues with nonzero real part seems to be infeasible. Moreover, as some of the equilibrium points of the projected dynamics are not necessarily equilibrium points of the unconstrained dynamics, this method is not sufficient to analyze behavior on the boundary of the unit cube.

Asymptotic stability

Asymptotic stability means that for small perturbations away from the equilibrium, over time the system converges back to that equilibrium point. In contrast to our negative result for structural stability, we are able to generate games for which the desired Bayesian-Nash equilibrium is asymptotically stable. We take two approaches to searching for asymptotically stable systems — a "direct" approach and an approach via an adaptation of Lyapunov's criterion (Merkin 1997).

Strict Bayesian-Nash equilibria

One of the fundamental insights from evolutionary game theory is that a major source of instability within various kinds of response dynamics is the presence of alternative optimal choices — if there are multiple actions available to a player which achieve a Bayesian-Nash equilibrium value (assuming all the other players are fixed and playing their Bayesian-Nash equilibrium strategies), best-response dynamics can slip between these actions (Samuelson 1997). Although the concept of an evolutionary Bayesian-Nash equilibrium does not seem to be easily expressed via constraints (as the definition involves an existential quantification), we can consider instead the (stronger) notion of

Bayesian-Nash equilibrium. As the name suggests, strict Bayesian-Nash equilibrium is defined identically to ordinary Bayesian-Nash equilibria except that the inequalities are made strict. In other words, there is a unique best response to the equilibrium action profile. When the number of actions is finite, strict Bayesian-Nash equilibria are asymptotically stable.

In our running example, truth-telling is a Bayesian-Nash equilibrium. However, any strategy of the form $(x, 0)$ (the husband is type L and lies about his type with probability $(1-x)$) is also a best-response to the truth-telling strategy for the wife. Therefore, the truth-telling strategy is not asymptotically stable — small perturbations lead to another best-response, at which point the gradient will not change the solution. For example, at the point $(0.99, 0, 1, 0)$ the gradient is $(0, -8.9, 0.03, -8.9)$ which implies that neither party can improve their situation since their gradients force them into the boundaries. Thus, even very small deviations from the truth-telling equilibrium $(1, 0, 1, 0)$ do not return. Fortunately, it is straightforward to modify the linear program described before in order to search for strict Bayesian-Nash equilibria.

To produce the strict linear program, we replace the third set of constraints with

$$\forall_{s\in S}\forall_{s'_i\in S_i} \sum_{o\in O} x_{s,o}u_i(s_i, o) \quad \geq \quad \sum_{o\in O} x_{s,o}u_i(s'_i, o) + \epsilon$$

where ϵ controls the precision needed to see the strictness. Setting $\epsilon = 0.1$, and solving the new linear program, we obtain the following solution:

♀/♂	L	H
L	$w = h = j = \frac{1}{3}$	$h = 1$
H	$w = 1$	$h = w = j = \frac{43}{264}, b = \frac{135}{264}$

Figure 2: Strict BNE mechanism. The payoff for each player at the equilibrium is slightly lower than it was before ($17\frac{227}{550}$ vs $17\frac{302}{550}$), but now truth-telling is a strict Bayesian-Nash equilibrium.

A problem with this method is that there exist games with asymptotically stable equilibria which are not strict equilibria, i.e. the condition is too strong. Moreover, picking ϵ is a tradeoff which requires consideration.

The method of Lyapunov functions

The definitive method for proving the asymptotic stability of a solution trajectory is to provide a Lyapunov function (Merkin 1997) for the dynamics.

Lyapunov Function. A Lyapunov function for a system of differential equations in variables $x_1, \ldots, x_n$ is a function $Y(x_1, x_2, \ldots, x_n)$ which is positive definite and such that $\frac{dY}{dt}$ is negative (semi)-definite.

The existence of such a function implies stability at the origin, and if the derivative is negative definite, the origin is asymptotically stable. To utilize this method, we produce additional constraints for the linear program which induce a simultaneous search for a mechanism and for a Lyapunov function which guarantees that the desired equilibrium point is stable or asymptotically stable (usually the latter). In general, the resulting program will have nonlinear polynomial constraints. In addition, it is a straightforward matter to transform coordinates so that the desired equilibrium point is at the origin.

However, because we are working with a system which is externally constrained to lie in the unit cube, we must employ a slight modification of the Lyapunov method. This technique is further developed in (Blumberg & shelat 2004).

Recursive Lyapunov Function. A recursive Lyapunov function for a system of differential equations constrained to lie on the unit cube in variables $x_1, \ldots, x_n$ is a function $Y(x_1, x_2, \ldots, x_n)$ which is positive definite on the unit cube and such that $\frac{dY}{dt}$ is negative definite on the unit cube. In addition, the restriction of $\frac{dY}{dt}$ obtained by setting a subset of the variables $\{x_j\}$ and the derivatives $\{\dot{x_j}\}$ to 0 is also negative definite.

We show elsewhere (Blumberg & shelat 2004) the following result, which we employ herein.

Theorem 2 *For a given constrained dynamics on the unit cube, if there exists a recursive constrained Lyapunov function, L, then the dynamics have an asymptotic equilibrium point at* 0.

In the remaining portion of this section, we will use our theorem in our running example. For the divorce mechanism, we consider a candidate Lyapunov function written as

$$Y = \sum_{i>j} a_{ij}\alpha_i\alpha_j$$

Evaluating the derivative, we find that

$$\frac{dY}{dt} = 2a_{11}\alpha_1\dot{\alpha}_1 + a_{12}(\dot{\alpha}_1\alpha_2 + x_1\dot{\alpha}_2) + a_{13}(\dot{\alpha}_1\alpha_3 + \alpha_1\dot{\alpha}_3) + \ldots + a_{44}\alpha_4\dot{\alpha}_4$$

We now augment our original program with the extra variables, $a_{11}, \ldots, a_{44}$ and now determine additional constraints in order to guarantee that Y is positive definite. To assure definiteness, we could employ the following constraints:

1. For every i, $a_{ii} > 0$. This ensures that when all the other variables are set to 0 the form is still positive.
2. For every i and j, $a_{ij} \geq 0$.

In order to ensure that dY/dt is negative definite over the unit cube, we must place additional constraints on the variables x^a_{yz} and a_{ij}. For the divorce example, calculation shows that dY/dt is a multivariate polynomial of degree 2. Consider a generic multivariate polynomial of degree 2,

$$\sum_k c_k x_k + \sum_{i,j} d_{ij}x_i x_j$$

For this to be negative definite on the unit cube, it suffices for the following conditions to hold:

1. For every i, $c_i \leq 0$ and $c_i + d_{ii} < 0$. This ensures that when all the other variables are set to 0 the form is still negative.
2. For every i and j, $d_{ij} \leq 0$.

Now, in the running example the $\{c_i\}$ and $\{d_{ij}\}$ are algebraic expressions involving the a_{ij}, the constants t_1, t_2 and u^*_{**}, and the other decision variables in the numerical program — $\{x_{s,o}\}$. Expanding this out, we obtain a program with quadratic constraints. For example, the constraint on c_1 requires the following expression be negative:

$$\begin{aligned}
&-8/5a_{13}x_{HH,h}+8a_{13}x_{HH,b}-4/5a_{13}x_{HH,j}-8a_{13}x_{HL,b}\\
&+8a_{11}x_{HH,b}+20a_{12}x_{LH,h}-20a_{12}x_{HH,h}+2a_{14}x_{HH,b}\\
&-20a_{14}x_{HH,h}+8/5a_{13}x_5+4/5a_{13}x_{HL,j}+8/5a_{11}x_{LH,h}\\
&+10a_{14}x_{HL,j}-2a_{12}x_{12}-4/5a_{11}x_{HH,j}-8a_{11}x_{LH,b}\\
&+10a_{12}x_{LH,j}+2a_{12}x_{HH,b}+4/5a_{11}x_{LH,j}-10a_{14}x_{HH,j}\\
&-2a_{14}x_{HL,b}-8/5a_{11}x_{HH,h}-10a_{12}x_{HH,j}+20a_{14}x_{HL,h}
\end{aligned}$$

Finally, we require the constraints which guarantee that Y is a recursive Lyapunov function — for every subset of the variables $\{x_i\}$, we must have that Y remains a negative definite form when all of the members of the subset and their derivatives are set to 0. These constraints are generated using the same criterion for negative definiteness specified above.

Remark One might be concerned by the quantity of additional constraints required for performing the simultaneous search for a game and a Lyapunov function. Indeed, the number of constraints required in the simultaneous search is exponential in the number of players and actions. Although this is unpleasant, it is no worse than the general mechanism design problem, which also requires constraints exponential in the number of players and actions.

When we solve this program, the game generated has a strict Bayesian-Nash equilibrium — but in addition we are provided with a proof of asymptotic stability in the form of the Lyapunov function encoded in the additional program variables.

Remark The conditions to ensure definiteness which we employ are sufficient, but considerably stronger than necessary — we use them for simplicity, but there are benefits in generality to be obtained by utilizing more precise conditions. Also, larger games will involve higher degree polynomials for which somewhat different conditions are required.

Conclusions

In this paper we have demonstrated extensions of the techniques of automated mechanism design which permit us to search for (and obtain) mechanisms which possess complicated additional properties beyond those implied by the basic problem statement. We believe that this general method of exploiting the flexibility of automated mechanism design to find mechanisms satisfying additional constraints will have broad applicability. For example, one could construct mechanisms which are insensitive to certain kinds of collusion.

We apply this methodology to the problem of finding mechanisms which are robust against player errors by presenting an algorithm which simultaneously searches for a game and a proof that the desired equilibrium point is asymptotically stable. We also discuss the difficulties of finding structurally stable mechanisms. Although we focus on the dynamics associated to a particular simple model of learning, any model of bounded rationality which leads to a dynamical system associated to repeated play is amenable to a similar analysis. In future work, we plan to extend our analysis to the system of stochastic differential equations that is associated to the classical economic notion of fictitious play.

References

Blumberg, A. J., and shelat, a. 2004. Lyapunov methods for constrained dynamics. Submitted.

Conitzer, V., and Sandholm, T. 2002. Complexity of mechanism design. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, 103–110.

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

Dekel, E.; Fudenberg, D.; and Levine, D. K. 2001. Learning to play bayesian games. In *Discussion Papers*, number 1322 in 1. Northwestern University, Center for Mathematical Studies in Economics and Management Science.

Fudenberg, D., and Tirole, J. 1991. *Game Theory*. The MIT Press.

Hirsch, M. W., and Smale, S. 1974. *Differential Equations, Dynamical Systems, and Linear Algebra*. San Diego, CA: Academic Press.

Larson, K., and Sandholm, T. 2001. Bargaining with limited computation: Deliberation equilibrium. *Artificial Intelligence* 132(2):183–217.

McLennan, A. 1999. On the expected number of nash equilibria of a normal form game. University of Minnesota working paper.

Merkin, D. R. 1997. *Introduction to the Theory of Stability*. Springer-Verlag.

Osborne, M. J., and Rubinstein, A. 1994. *A Course in Game Theory*. The MIT Press.

Samuelson, L. 1997. *Evolutionary Games and Equilibrium Selection*. The MIT Press.

Singh, S.; Kearns, M.; and Mansour, Y. 2000. Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 541–548. Morgan Kaufmann.