Computing Equilibria by Incorporating Qualitative Models*

Sam Ganzfried
Department of Computer Science
Carnegie Mellon University
sganzfri@cs.cmu.edu

Tuomas Sandholm
Department of Computer Science
Carnegie Mellon University
sandholm@cs.cmu.edu

ABSTRACT

We present a new procedure for solving large games of imperfect information. Our approach involves—somewhat counterintuitively—solving an infinite approximation of the original game, then mapping the equilibrium to a strategy profile in the original game. Our main algorithm exploits some qualitative model of equilibrium structure as an additional input to find an equilibrium in continuous games. We prove that our approach is correct even if given a set of qualitative models (satisfying a technical property) of which only some are accurate. We compute equilibria in several classes of games for which no prior algorithms have been developed. In the course of our analysis, we also develop the first mixed-integer programming formulations for computing an epsilon-equilibrium in general multiplayer normal and extensive-form games based on the extension of our initial algorithm to the multiplayer setting, which may be of independent interest. Experiments suggest that our approach can outperform the prior state of the art, abstraction-based approaches. In addition, we demonstrate the effectiveness of our main algorithm on a subgame of limit Texas hold'em—the most studied imperfect-information game in computer science.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Artificial Intelligence

General Terms

Algorithms, Economics

Keywords

Game theory, equilibrium finding, continuous games

1. INTRODUCTION

Finding Nash equilibria in games of imperfect information is an important problem. Significant progress has been made in the last few years. In particular, several algorithms have been developed for solving large (finite) two-player zero-sum imperfect-information games (e.g., [10, 21]). On the other hand, relatively little work has been done on developing algorithms for computing equilibria in imperfect-information games that are non-zero sum (with some notable exceptions, e.g., [11]), have more than two players (e.g., [9]), and/or are continuous (i.e., infinite) (e.g., [17]). Such games are significantly harder to solve in the complexity-theoretic sense: two-player zero-sum games can be solved in polynomial time, while two-player general-sum games are PPAD-hard [6], and games with three or more players are FIXP-hard [7].

To make matters worse, many interesting real-world games are so large that computing an equilibrium directly seems hopeless even in the two-player zero-sum case. The standard approach to deal with this is to run an abstraction algorithm on the full game to construct a smaller game that is strategically similar to the original game (e.g., [3]). Then the abstracted game is solved, and its solution is mapped to a strategy profile in the original game. While this approach seems quite natural and promising so far, recent research has shown that significant and surprising pathologies can arise in which a finer-grained abstraction results in strategies that are actually much more exploitable in the full game than the equilibrium strategies of a coarser abstraction [20]. Thus, this abstraction-based approach is not theoretically sound.

We develop a new approach for solving large games. Rather than construct a smaller game via an abstraction algorithm, we propose solving an infinite approximation of the original game, then mapping the equilibrium of the infinite game to a strategy profile in the original game. Perhaps counterintuitively, it is often the case that the infinite approximation can be solved much more easily than the finite game. We show that sometimes very fine abstractions would be needed to match the solution quality of our approach.

Our main algorithmic innovation takes advantage of the fact that in many multiagent settings it is significantly easier to infer qualitative models of the structure of equilibrium strategies than it is to actually compute an equilibrium. For example, in (sequences of) take-it-or-leave-it offers, equilibria involve accepting offers above a certain threshold and rejecting offers below it [13]. Threshold strategies are also common in auctions (e.g., [5]) and in deciding when to make and break partnerships and contracts (e.g., [15]). In poker the hole cards are private signals, and in equilibrium, often the same action is taken in continuous regions of the signal space (e.g., [1]).

We develop an approach for exploiting such qualitative models in equilibrium finding. We study a broad class of imperfect-information games where players are given private signals at the start. We first consider the two-player (general-sum) case in which private signals are drawn independently from finite sets. For this case, we develop an algorithm based on a mixed integer linear feasibility program (MILFP) formulation that provably computes an equilibrium assuming we are given a "correct" qualitative model

*This material is based upon work supported by the National Science Foundation under grants IIS-0427858 and IIS-0905390. We also thank Intel Corporation and IBM for their machine gifts.

Cite as: Computing Equilibria by Incorporating Qualitative Models, Sam Ganzfried and Tuomas Sandholm, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

as input. The size of the program is polynomial in the parameters of the problem and the constraints are very sparse, suggesting that it can be solved quickly in practice. Our experiments confirm that the algorithm runs very fast on a simplified endgame of limit Texas hold'em, leading to a significant performance improvement.

Next, we generalize our algorithm to computing equilibria in the following important extensions: many players, continuous private signal distributions, and multiple candidate qualitative models that satisfy a certain technical property (some of which can be incorrect). In most of these cases, we present the first algorithm that provably solves the class of games. We also develop new mixed-integer programming based algorithms for computing equilibria in general multiplayer normal and extensive-form games based on the extension of our initial algorithm to the multiplayer setting, which may be of independent interest.

2. CONTINUOUS GAMES

Continuous games generalize finite strategic-form games to the case of (uncountably) infinite strategy spaces. Many natural games have an uncountable number of actions; for example, games in which strategies correspond to an amount of time, money, or space. One example of a game that has recently been modeled as a continuous game in the AI literature is computational billiards, in which the strategies are vectors of real numbers corresponding to the orientation, location, and velocity at which to hit the ball [2].

DEFINITION 1. *A* continuous game *is a tuple* $G = (N, S, U)$ *where*

- $N = \{1, 2, 3, \ldots, n\}$ *is the set of players,*
- $S = (S_1, \ldots, S_n)$ *where each* S_i *is a metric space corresponding to the set of strategies of player* i*, and*
- $U = (u_1, \ldots, u_n)$ *where* $u_i : S \rightarrow \mathbb{R}$ *is the utility function of player* i*.*

The main result regarding the existence of a Nash equilibrium in continuous games is the following [8]:

THEOREM 1. *Consider a strategic-form game whose strategy spaces* S_i *are nonempty compact subsets of a metric space. If the payoff functions* u_i *are continuous, there exists a (mixed strategy) Nash equilibrium.*

While this existence result has been around for a long time, there has been very little work on practical algorithms for computing equilibria in continuous games. One interesting class of continuous games for which algorithms have been developed is *separable games* [17]:

DEFINITION 2. *A* separable game *is a continuous game with utility functions* $u_i : S \rightarrow \mathbb{R}$ *taking the form*

$$u_i(s) = \sum_{j_1=1}^{m_1} \cdots \sum_{j_n=1}^{m_n} a_i^{j_1 \cdots j_n} f_1^{j_1}(s_1) \ldots f_n^{j_n}(s_n),$$

where $a_i^{j_1 \cdots j_n} \in \mathbb{R}$ *and the* $f_i^j : S_i \rightarrow \mathbb{R}$ *are continuous.*

As we will see, this is a significant restriction on the utility functions, and many interesting continuous games are not separable. Additionally, algorithms for computing approximate equilibria have been developed for several other classes of infinite games, including simulation-based games [19] and graphical tree-games [16].

For a broad class of games, we will show that the equilibrium existence theorem above does not hold directly, but we can nevertheless prove existence of equilibrium by incorporating a qualitative equilibrium model. However, we show that these games are not separable, so the prior algorithm does not apply. These are the topics of the next two sections. After that, we will develop new algorithms for solving these games.

3. MOTIVATING EXAMPLE

Consider the following simplified poker game [1]. Suppose two players are given private signals, x_1 and x_2, independently and uniformly at random from [0,1]. Suppose the pot initially has size ρ (one can think of this as both players having put in an ante of $\frac{\rho}{2}$, or that we are at the final betting round—aka final *street*—of a multi-street game). Player 1 is allowed to bet or check. If player 1 checks, the game is over and the player with the *lower* private signal wins the pot (following the convention of [1]). If player 1 bets, then player 2 can call or fold. If player 2 folds, then player 1 wins the pot. If player 2 calls, then whoever has the lower private signal wins $\rho + 1$, while the other player loses 1. This situation can be thought of as an abstraction of the final street of a hand of limit Texas hold'em where raising is not allowed and player 2 has already checked.

It seems natural to define the strategy space S_1 of player 1 as the set of functions from $[0, 1]$ to $\{\text{bet}, \text{call}\}$ (i.e., to $\{0, 1\}$), and to define S_2 for player 2 similarly. Let $p_i(a_1, a_2, x_1, x_2)$ denote the payoff of player i when the players play actions a_i and are given private signals x_i. Then the utility of player i under the strategy profile $s = (s_1, s_2)$ is defined as

$$u_i(s) = \int_{x_1=0}^{1} \int_{x_2=0}^{1} p_i(s_1(x_1), s_2(x_2), x_1, x_2) dx_2 dx_1.$$

We would like to represent each player's strategy set as a compact metric space, so that we can apply Theorem 1. Unfortunately, the naive representation does not yield compact metric spaces; so, we need to go through a number of transformations to achieve this goal. In particular, by iteratively eliminating dominated strategies, we arrive at a representation where each player's strategy space is isomorphic to a compact subset of a Euclidean space.

In order for $u_i(s)$ to be defined, we must restrict the strategy spaces S_i to consist of only the measurable functions. In addition, if we want to turn S_i into a metric space, we need to define a distance function. A natural distance function to use is the L_1 distance function: $d_i(s_i, s_i') = \int_{X_i} |s_i(x_i) - s_i'(x_i)| dx_i$. Note that (S_i, d_i) does not quite define a metric space because the condition $d_i(s, t) = 0$ iff $s = t$ is not satisfied. To turn it into a metric space, we can let $\sim$ be the equivalence relation defined by $s \sim_i s'$ iff $d_i(s, s') = 0$. If we then let $\overline{S_i}$ equal the set of equivalence classes with respect to $\sim_i$, then $(\overline{S_i}, d_i)$ is a metric space.

Unfortunately, the metric space $(\overline{S_i}, d_i)$ is not compact, and we cannot apply Theorem 1 to guarantee the existence of an equilibrium.

PROPOSITION 1. *The metric space* $(\overline{S_i}, d_i)$ *is not compact.*

For brevity we omit the proofs. A full version with the proofs is available on the authors' websites.

Similarly, the space fails to be compact if we use other common distance functions, such as the discrete metric, any L_p metric, or L_∞. So we can not simply use one of those distance functions instead of L_1 to get around Proposition 1.

However, the following observations allow us to restrict our attention to a much smaller set of strategies.

DEFINITION 3. *Let S_i be the set of pure strategies of player i. Then $s_i \in S_i$ is* weakly dominated *for player i if there exists a strategy $s_i^* \in S_i$ such that for all strategy profiles $s_{-i} \in S_{-i}$ for the other players, we have $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$.*

DEFINITION 4. *Let $\overline{s_i}$ be an equivalence class of player i's pure strategies with respect to $\sim_i$. Then $\overline{s_i}$ is* weakly dominated *if s_i is weakly dominated for all $s_i \in \overline{s_i}$.*

DEFINITION 5. *The equivalence class $\overline{s_i}$ is* undominated *if it is not weakly dominated.*

PROPOSITION 2. *The equivalence class of strategies $\overline{s_2} \in \overline{S_2}$ of player 2 is undominated only if it contains a unique strategy of the following form: call if $x_2 \leq x_2^*$ and fold otherwise, for some x_2^*.*

We can remove all of the strategies from $\overline{S_2}$ that are dominated according to Proposition 2 from our consideration, forming a much smaller strategy space. In addition, we can remove the strategies not satisfying the threshold property given in the proposition from each equivalence class $\overline{s_2} \in \overline{S_2}$, thus turning the equivalence classes into singletons. In the remainder of our discussion, we will let $\overline{S_2}$ denote this smaller set.

We can now iteratively remove many strategies from S_1 by the following observation.

PROPOSITION 3. *The equivalence class of strategies $\overline{s_1} \in \overline{S_1}$ of player 1 is a best response to some element of $\overline{S_2}$ only if it contains a unique strategy of the following form: bet if $x_1 \leq \underline{x_1^*}$, check if $\underline{x_1^*} \leq x_1 \leq \overline{x_1^*}$ and bet if $\overline{x_1^*} \leq x_1 \leq 1$, for some $\underline{x_1^*} \leq \overline{x_1^*}$[1].*

After removing the dominated strategies, the new strategy space for player 1 becomes isomorphic to a compact subset of $\mathbb{R}^2$, and the new strategy space for player 2 becomes isomorphic to a compact subset of $\mathbb{R}$. Let $\hat{S}_1$ and $\hat{S}_2$ now refer to these new strategy spaces.

It turns out that the functions u_i are continuous in s for both players using the new strategy spaces, if we define the distance between two strategy profiles $s = (s_1, s_2)$ and $s' = (s_1', s_2')$ as $d(s,t) = d_1(s_1, s_1') + d_2(s_2, s_2')$.

PROPOSITION 4. *For both players, the utility functions u_i are continuous in s.*

It follows from Theorem 1 that the game has a Nash equilibrium using the new strategy spaces $\hat{S}_1$ and $\hat{S}_2$.

Now that the existence of an equilibrium is guaranteed, we must figure out how to compute one. It turns out that the game is not separable, so one cannot apply the algorithm from prior work [17].

PROPOSITION 5. *This game is not separable.*

However, it turns out that we can still solve this game quite easily if we notice that every equilibrium will have $\underline{x_1^*} \leq x_2^* \leq \overline{x_1^*}$. Given this guess of the form of an equilibrium, it is easy to compute an equilibrium by noting that a player must be indifferent between two actions at each threshold. For example, at $\underline{x_1^*}$ player 1 must be indifferent between betting and checking. His expected payoff of betting is $(1 - x_2^*)\rho + (x_2^* - \underline{x_1^*})(\rho + 1) - \underline{x_1^*}$ and his expected payoff of checking is $\rho(1 - \underline{x_1^*})$. Setting these two expressions equal to each other yields $x_2^* = 2\underline{x_1^*}$. We can create a linear equation similarly for the other two thresholds. Thus we have a system of n linear equations with n unknowns (where n is the total number of thresholds), which can be solved efficiently by matrix inversion.

[1]One can think of $[0, \underline{x_1^*}]$ as player 1's "value betting" range—where he bets hoping to get called by worse hands—and $[\overline{x_1^*}, 1]$ as his "bluffing" range—where he bets hoping to get better hands to fold.

4. OUR SETTING

The main setting of the rest of the paper will be a generalization of the setting of the above example along several dimensions.

DEFINITION 6. *A* continuous Bayesian game *is a tuple $G = (N, X, C, U, F)$ where*

- *$N = \{1, 2, 3, \ldots, n\}$ is the set of players,*
- *$X = (X_1, \ldots, X_n)$ where each X_i is a compact subset of $\mathbb{R}$ corresponding to the set of private signals of player i,*
- *$C = (C_1, \ldots, C_n)$ where each C_i is a compact subset of $\mathbb{R}$ corresponding to the set of actions of player i,*
- *$U = (u_1, \ldots, u_n)$ where $u_i : C \times X \to \mathbb{R}$ is the measurable utility function of player i, and*
- *$F = (F_1, \ldots, F_n)$ where $F_i : X_i \to [0, 1]$ is the cumulative distribution function (CDF) of the private signal distribution of player i.*

The strategy space S_i of player i is the set of all measurable functions from X_i to C_i. We define Λ_i to be the set of Borel probability measures on S_i, giving us the mixed strategy space of player i. Let $\Lambda = \times_i \Lambda_i$. A *(mixed) strategy profile* is $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$, where $\sigma_i \in \Lambda_i$. Let $\sigma_{i|x_i}$ denote the induced probability density function of σ_i given signal x_i over C_i. Then define

$$\hat{u_i}(\boldsymbol{\sigma}, \mathbf{x}) = \int_C \prod_j \sigma_{j|x_j}(c_j) u_i(\boldsymbol{c}, \boldsymbol{x}) d\boldsymbol{c}.$$

Define $u_{i|x_i}(\boldsymbol{\sigma})$, where $\boldsymbol{\sigma} \in \Lambda$, $x_i \in X_i$, as follows:

$$u_{i|x_i}(\boldsymbol{\sigma}) = \int_{X_1} \ldots \int_{X_{i-1}} \int_{X_{i+1}} \ldots \int_{X_n} \hat{u_i}(\boldsymbol{\sigma}, \boldsymbol{x})$$

$$dF_n \ldots dF_{i+1} dF_{i-1} \ldots dF_1.$$

This denotes the expected utility of player i given that he has received private signal x_i when strategy profile $\boldsymbol{\sigma}$ is played.

For $i \in N$, $x_i \in X_i$, $\sigma_i \in \Lambda_i$ and $\sigma_{-i} \in \times_{j \neq i} \Lambda_j$, define $u_{i|x_i}(\sigma_i, \sigma_{-i})$ as the expected utility of player i given private signal x_i when he plays σ_i and the other players play σ_{-i}.

According to the definition of Nash equilibrium, player i can play arbitrarily in cases where he has a private signal that has zero measure in the signal distribution F_i. Such behavior can result in equilibria that violate our qualitative models (discussed later) even when "equivalent" equilibria exist that satisfy the models. Thus, we define a slightly stronger notion of equilibrium in order to rule out arbitrary behavior in these regions of measure zero. In other words, we require an agent to play rationally even if he gets a private signal that has zero probability. (This strengthening of the equilibrium concept is analogous to *perfect Bayesian equilibrium*, where each agent has to act rationally even in information sets that are reached with zero probability due to the strategies of the players. In our case the reaching with zero probability is due to nature's action, i.e., giving the agents types.)

DEFINITION 7. *Strategy profile $\sigma \in \Lambda$ is an* every-private-signal Bayesian (EPSB) equilibrium *of G if for all $i \in N$, for all $x_i \in X_i$, and for all $\tau_i \in \Lambda_i$, we have $u_{i|x_i}(\sigma_i, \sigma_{-i}) \geq u_{i|x_i}(\tau_i, \sigma_{-i})$.*

PROPOSITION 6. *Let G be a game of the form given in Definition 6. Then G has an EPSB equilibrium if and only if it has an equilibrium.*

We now strengthen the notions of best response and dominated strategies analogously. These will be useful when we analyze our algorithms.

DEFINITION 8. *Strategy σ_i is an EPSB best response for player $i \in N$ to profile σ_{-i} if for all $x_i \in X_i$ and for all $\tau_i \in \Lambda_i$, we have $u_{i|x_i}(\sigma_i, \sigma_{-i}) \geq u_{i|x_i}(\tau_i, \sigma_{-i})$.*

DEFINITION 9. *Strategy σ_i is an EPSB ϵ-best response for player $i \in N$ to profile σ_{-i} if for all $x_i \in X_i$ and for all $\tau_i \in \Lambda_i$, we have $u_{i|x_i}(\sigma_i, \sigma_{-i}) \geq u_{i|x_i}(\tau_i, \sigma_{-i}) - \epsilon$.*

DEFINITION 10. *Strategy σ_i is EPSB-undominated if for all $\tau_i \in \Sigma_i$ there exist $x_i \in X_i$, $\sigma_{-i} \in \Sigma_{-i}$ such that $u_{i|x_i}(\sigma_i, \sigma_{-i}) > u_{i|x_i}(\tau_i, \sigma_{-i})$.*

5. PARAMETRIC MODELS

In many multiagent settings, it is significantly easier to infer qualitative models of the structure of equilibrium strategies than it is to compute an equilibrium. The introduction gives several examples, including sequences of take-it-or-leave-it offers, certain auctions, and making or breaking partnerships and contracts. In general, we call the values that divide the different strategic regions *thresholds* (e.g., $\underline{x_1^*}, \overline{x_1^*}$, and x_2^* in the example above), and refer to the guess of the structure of an equilibrium defined by these thresholds a *parametric model*. Many additional examples of games that are solved by the procedure used in the above example appear in [1].

DEFINITION 11. *A* parametric model *of game $G = (N,X,C,U,F)$ is a tuple $P = (T, Q, \prec)$ where*

- *$T = (T_1, \ldots, T_n)$, where T_i denotes the number of regions for player i,*
- *$Q = (Q_1, \ldots, Q_n)$, where Q_i is a sequence $\{q_{ij} : 1 \leq j \leq T_i\}$, where $q_{ij} \in C_i$ denotes the action taken by player i in his j'th region of the model (at a boundary the lower region's action is taken), and*
- *$\prec$ is a partial ordering over the set of tuples $(y_{ij}, y_{i'j'})$, where $y_{ij} \prec y_{i'j'}$ if we require that the lower threshold of player i's j'th region is less than or equal to the lower threshold of player i''s j''th region.*

We saw in Section 3 that restricting the strategy spaces of a game by forcing all strategies to conform to a specified parametric model can allow us to both guarantee the existence of an equilibrium and to actually compute one when neither of these could be accomplished in the original game by previously known techniques.

6. OUR MAIN ALGORITHM

In this section we present our algorithm for computing an equilibrium given a parametric model. While parametric models associate a pure action for each interval of signals, this can be problematic when the probability of obtaining individual private signals is nonzero. In this case, our algorithm will actually output mixed strategies.

For now we assume the game is finite, has two players, and a single parametric model is specified. We will extend the algorithm to more general settings along each of these dimensions in Section 7.

6.1 Constructing a MILFP

Given a problem instance $G = (N, X, C, U, F)$ and a parametric model P, we first construct a mixed integer linear feasibility program (MILFP) that contains both integer and continuous variables. Since X_i is finite for all players, we assume without loss of generality that it is the set of integers from 1 to $\overline{n}$. Let $\{t_i\}$ denote the union of the sets of thresholds for both players under P. For each threshold t_i, we introduce two real-valued variables, x_i and y_i, where x_i corresponds to $F_1(t_i)$ and y_i corresponds to $F_2(t_i)$. For each threshold t_i and each integer $j \in [1, \overline{n}]$, we introduce an indicator (0–1 integer) variable, $z_{i,j}$, such that $z_{i,j} = 1$ implies $j - 1 \leq t_i \leq j$. So, overall we have $2|T| + |T||S|$ variables, where $|T|$ is the number of thresholds in P and $|S|$ is the number of private signals.

Indifference constraints.

As the example in Section 3 demonstrates, we want to obtain indifference between two actions at each threshold. Thus, we have $|T|$ equality constraints where each is of the form $f(x, y) = 0$ where f is linear in the x_i and y_i.

Threshold ordering constraints.

In order to guarantee that the solution conforms to the parametric model, we must add inequality constraints corresponding to the partial ordering $\prec$. If $t_i \prec t_j$, then we add the constraints $x_i \leq x_j$, $y_i \leq y_j$.

Consistency constraints.

Next, we require that for each i, x_i and y_i are consistent in the sense that there exists some value for t_i such that $F_1(t_i)$ corresponds to x_i and $F_2(t_i)$ corresponds to y_i. To accomplish this, we include indicator constraints of the following form for each $i, j : z_{i,j} = 1 \Rightarrow F_1(j-1) \leq x_i \leq F_1(j)$ and $z_{i,j} = 1 \Rightarrow F_2(j-1) \leq y_i \leq F_2(j)$ where we define $F_1(-1) = F_2(-1) = 0$. Stated explicitly, we add the following 4 linear inequalities for each $i \in [1, |T|], j \in [0, \overline{n}]$:

$$x_i - F_1(j-1)z_{i,j} \geq 0 \qquad x_i - z_{i,j}(F_1(j) - 1) \leq 1$$

$$y_i - F_2(j-1)z_{i,j} \geq 0 \qquad y_i - z_{i,j}(F_2(j) - 1) \leq 1$$

Finally, we must ensure that for each i, $z_{i,j} = 1$ for precisely one j (i.e., $t_i \in [j-1, j]$ for some j). We accomplish this by adding the equality constraint $\sum_{j=0}^{\overline{n}} z_{i,j} = 1$ for each i.

Thus, there are $O(|T||S|)$ consistency constraints, where $|S|$ is the number of private signals. There are more consistency constraints than constraints of any other type, and thus the MILFP has $O(|T||S|)$ total constraints.

6.2 Obtaining mixed strategies from the MILFP solution

Once we obtain the x_i and y_i by solving the MILFP, we must map them into mixed strategies of the game. Suppose player 1 is dealt private signal $z \in [1, \overline{n}]$ and consider the interval $I = [F_1(z-1), F_1(z)]$. Now define the intervals $J_i = [x_{i-1}, x_i]$ where we define $x_{-1} = 0$. Let O_i denote the overlap between sets I and J_i. Then player 1 will play the strategy defined by region i with probability $\frac{O_i}{\sum_i O_i}$. The strategy for player 2 is determined similarly, using the y_i and F_2.

6.3 Algorithm soundness and completeness

We are now ready to prove that our algorithm indeed yields an equilibrium.

THEOREM 2. *Suppose that for all $i \in N$ and for all $\sigma_{-i} \in \Lambda_{-i}$, all pure-strategy EPSB best responses of player i to σ_{-i} satisfy the given parametric model. Then our algorithm outputs an equilibrium.*

The theorem directly implies the following corollary. In some settings it may be easier to check the premise of the corollary than the premise of the theorem.

COROLLARY 1. *Suppose all EPSB-undominated strategies follow the given parametric model. Then our algorithm outputs an equilibrium.*

7. EXTENSIONS TO MORE GENERAL SETTINGS

In this section we describe several important extensions of our approach to more general settings.

7.1 Continuous private signal distributions

In this subsection we generalize the approach to continuous private signal distributions. If the CDFs F_i are continuous and piecewise linear, we only need to alter the consistency constraints. Suppose that $\{c_i\}$ denotes the union of the breakpoints of the CDFs of the F_i, and suppose $F_i(x) = a_{ij}x + b_{ij}$ for $c_{j-1} \leq x \leq c_j$ (where $c_{-1} = 0$). Then we add the following constraints for all j and all pairs of players (i, i') with $i' = i + 1$, where $x_i, x_{i'}$ correspond to the given threshold variables, $\gamma = a_{i'j}b_{ij} - a_{ij}b_{i'j}$, and we assume all the $x_i, x_{i'}$ lie in $[\underline{M}, \overline{M}]$:

$$a_{i'j}x_i - a_{ij}x_{i'} \geq z_{i,j}(\gamma - a_{i'j}\underline{M} + a_{ij}\overline{M}) + a_{i'j}\underline{M} - a_{ij}\overline{M}$$

$$a_{i'j}x_i - a_{ij}x_{i'} \leq z_{i,j}(\gamma - a_{i'j}\overline{M} + a_{ij}\underline{M}) + a_{i'j}\overline{M} - a_{ij}\underline{M}.$$

We also retain the constraints from the discrete case which ensure that $z_{i,j} = 1$ implies $c_{j-1} \leq x_i \leq c_j$.

Using the technique described in the next subsection, we can also approximately solve the problem for continuous CDFs that are not piecewise linear by approximating them with piecewise linear functions.

7.2 Many players

In games with more than two players, the indifference constraints are no longer linear functions of the variables. (All other constraints remain linear.) With n players the indifference constraints are degree $n - 1$ polynomials. Therefore, there is a need to represent products of continuous variables, $x_i x_j$, using linear constraints only since we wish to model the problem as a MILFP.

7.2.1 Approximating products of continuous variables using linear constraints

In this subsection we describe how a modeling technique [4] can be applied to approximate the nonlinear functions by piecewise linear functions. First we define two new variables $\beta_1 = \frac{1}{2}(x_i + x_j)$ and $\beta_2 = \frac{1}{2}(x_i - x_j)$, noting that $\beta_1^2 - \beta_2^2 = x_i x_j$. To approximate $w_1 = \beta_1^2$, we select k breakpoints from the interval [0,1]—in our experiments we will use $q_i = \frac{i-1}{k-1}$, where $k(\epsilon)$ is a function of an input parameter ϵ. Next, we add the constraint $\sum_{i=1}^{k} \lambda_{1i} q_i^2 = w_1$, where the λ_{1i} are continuous variables. Next we add the constraint $\sum_{i=1}^{k} \lambda_{1i} q_i = \beta_1$. We also add the constraint $\sum_{i=1}^{k} \lambda_{1i} = 1$, where we also require that at most two adjacent $\lambda'_{1i}s$ are greater than zero (we accomplish this in the standard way of adding a binary indicator variable per segment and the appropriate constraints, called SOS2 constraints). Then if we define the variable u_{ij} to represent the product $x_i x_j$, we just need to add the constraint $u_{ij} = w_1 - w_2$, where w_2 and its additional constraints are defined analogously to w_1.

Finally, we replace each indifference equation $f(x) = 0$ with the inequalities $f(x) \leq \frac{\epsilon}{2}$ and $f(x) \geq -\frac{\epsilon}{2}$ where ϵ is an approximation parameter given as input to the algorithm.

7.2.2 Tying the accuracy of the approximation to the accuracy of the equilibrium

Suppose we select $k + 1$ breakpoints per piecewise linear curve, with

$$k \geq \sqrt{\frac{(\overline{T} + 2)^{(n-1)} M(n-1)}{\epsilon}}, \tag{1}$$

where $\overline{T}$ is the maximal number of thresholds of one of the parametric models for a player, M is the difference between the maximum and minimum possible payoff of the game, n is the number of players, and ϵ is an approximation parameter given as input to the algorithm.

THEOREM 3. *Suppose that for all $i \in N$ and for all $\sigma_{-i} \in \Lambda_{-i}$, all pure-strategy EPSB ϵ-best responses of player i to σ_{-i} satisfy the given parametric model. Furthermore suppose that the number of breakpoints satisfies Equation 1. Then our algorithm outputs an ϵ-equilibrium.*

For particular games, the number of breakpoints needed to obtain a desired ϵ can actually be far smaller. For example, if each indifference equation consists of the sum of at most T^* expressions, for $T^* < (\overline{T} + 2)^{(n-1)}$, then we can replace $(\overline{T} + 2)^{(n-1)}$ with T^* to create a tighter upper bound. Additionally, even though the number of breakpoints in Equation 1 is exponential in the number of players, one can alternatively model the problem as a MILFP using a polynomial number (in n) of constraints and variables. This is accomplished by a recently published way of modeling piecewise linear functions in a MIP [18]. (It uses a binary rather than unary encoding to refer to the pieces via indicator variables.)

7.2.3 New MIP algorithms for computing equilibria in normal and extensive-form games

It is worth noting that the modeling approach of Section 7.2.1 can be used to develop new algorithms for computing an ϵ-equilibrium in general-sum games with two or more players in both normal and extensive form. In particular, the *MIP Nash* algorithm for computing an equilibrium in two-player general-sum normal-form games [14] can be directly extended to a MIP formulation of multiplayer normal-form games which contains some nonlinear constraints (corresponding to the expected utility constraints). If we apply our approach using sufficiently many breakpoints, we can obtain an ϵ-equilibrium for arbitrary ϵ by approximating the nonlinear constraints by piecewise linear constraints. Additionally, we can represent the equilibrium-computation problem in multiplayer extensive-form games as a MIP if we write out the expected utility constraints separately on a per-information-set basis. This leads to a new algorithm for computing equilibria in multiplayer extensive-form games, an important class of games for which no algorithms for computing a Nash equilibrium with solution guarantees were known.

7.3 Multiple parametric models

Quite often it is prohibitively difficult to come up with one parametric model, P, that is correct, but one can construct several parametric models, P_i, and know that at least one of them is correct.

This is the case for our experiments on Texas hold'em in Section 8.2. This scenario could arise for several reasons; for example, often we can immediately rule out many parametric models because all strategies that satisfy them are dominated. We now generalize our approach to this situation.

We define the notion of model refinement in a natural way:

DEFINITION 12. *$P = (T, Q, \prec)$ is a* refinement *of $P' = (T', Q', \prec')$ if for each $i \in N$, Q'_i is a (not necessarily contiguous) subsequence of Q_i.*

DEFINITION 13. *P is a* US-refinement *of P' if Q'_i corresponds to a unique subsequence of Q_i for each i.*

For example, if $N = \{1\}$ and $Q'_1 = \{1, 2\}$ while $Q_1 = \{1,2,3,2\}$, then P is a refinement of P', but is not a US-refinement.

We now generalize our algorithm to the setting where the P_i have a common US-refinement P'. We first define an indicator variable ζ_i corresponding to each model. Next we replace each indifference constraint $f(x) = 0$ corresponding to model P_i by the following two inequalities, where K is a sufficiently large constant: $f(x) - K\zeta_i \geq -K$ and $f(x) + K\zeta_i \leq K$.

Next we add certain inequalities corresponding to the models P_i that differ from P'. For simplicity, we will demonstrate these by example. Suppose that, under P', player 1 plays action a_1 in his first region, a_2 in his second region, and a_3 in his third region. Suppose that in P_1 he plays a_1 in his first region and a_3 in his second region (recall that P' is a refinement of P_1). Then we must add two constraints that ensure that at the first threshold of P_1, both a_1 and a_3 are (weakly) preferred to a_2. In general, whenever actions of P' are omitted by a P_i, we must add constraints to the neighboring actions at their intersection ensuring that they are preferred to the omitted actions.

We also replace each order constraint $x_j - x_{j'} \leq 0$ corresponding to model P_i by $x_j - x_{j'} + K\zeta_i \leq K$. Finally, we add the equality $\sum_i \zeta_i = 1$ to ensure that only the constraints corresponding to one of the candidate parametric models are used in the solution.

The following theorem states that our approach is correct even in this setting where there are multiple parametric models, assuming they have a common US-refinement.

THEOREM 4. *Let there be two players. Let $\{P_i\}$ be a set of parametric models with a common US-refinement. Suppose that for all $i \in N$ and for all $\sigma_{-i} \in \Lambda_{-i}$, all pure-strategy EPSB best responses of player i to σ_{-i} satisfy at least one of the P_i (not necessarily the same P_i). Then our algorithm outputs an equilibrium.*

We can also obtain a theorem with an ϵ guarantee similar to Theorem 3 for the case of more than two players.

It is worth noting that the number of variables and constraints in the new MILFP formulation is still $O(|S||T|)$ (assuming a constant number of parametric models). Alternatively, we could have solved several MILFP's—one for each parametric model. While each MILFP would be smaller than the one we are solving, each would still have $O(|S||T|)$ variables and $O(|S||T|)$ constraints, and thus have the same size asymptotically as our formulation. This alternative formulation is also potentially effective, assuming we have access to several processors to run the threads in parallel.

8. EXPERIMENTS

We now present results from several experiments that investigate the practical applicability of our algorithm and extensions, as well as the overall procedure of solving large finite games by approximating them by continuous games.

n	50	100	150	200	250
$v(G_n)$	-0.0576	-0.0566	-0.0563	-0.0561	-0.0560
$\pi(\sigma_n)$	-0.0624	-0.0612	-0.0579	-0.0583	-0.0560

Figure 1: Worst-case payoff of playing the projection of the equilibrium of G_∞ ($\pi(\sigma_n)$) versus the value of G_n ($v(G_n)$).

8.1 Approximating large finite games by continuous games

In Section 3 we saw an example of a game with infinite strategy spaces that could be solved by an extremely simple procedure (once we guessed a correct parametric model). If instead the set of private signals were the finite set $\{1, \ldots, n\}$, then it is clear that as n gets large, the running time of computing an exact equilibrium of the game will get arbitrarily large; on the other hand, solving the infinite approximation as n goes to infinity will still take essentially no time at all, and we would expect the solution to the infinite game to be very close to the solution of the finite game. In this section we will consider a similar game and show that very fine-grained abstractions would be needed to match the solution quality of our approach.

Kuhn poker is a simplified version of poker that was one of the first games studied by game theorists [12]. It works as follows. There are two players and a deck containing three cards: 1, 2, and 3. Each player is dealt one card at random, and both players ante \$1. Player 1 acts first and can either check or raise by \$1. If player 1 raises, player 2 can either call—in which case whoever has the higher card wins the \$4 pot—or fold—in which case player 1 wins the entire \$3 pot. If player 1 checks, player 2 can check—in which case whoever has the higher card wins the \$2 pot—or bet. If player 1 checks and player 2 bets, player 1 can call—in which case whoever has the higher card wins the \$4 pot—or fold, in which case player 2 wins the \$3 pot.

Generalized Kuhn poker, G_n, has the same rules as Kuhn poker except that the deck contains n cards instead of 3. Define G_∞ to be the same as G_n except the players are both dealt a real number drawn uniformly at random from the unit interval $[0, 1]$. Informally, G_∞ is like the limit as n approaches infinity of G_n. It turns out that G_∞ has a relatively simple pure strategy Nash equilibrium that is derived in [1]. It can be computed by solving a system of six linearly independent indifference equations with six unknowns.

Once the infinite game, G_∞, has been solved, its solution can be projected down to a corresponding strategy profile in G_n: call this profile σ_n. We ran experiments for several settings of n. We compared the performance of σ_n against its nemesis to the value of the game to player 1 (i.e., how an optimal strategy performs): the results are summarized in Figure 1. The payoffs agree to four decimal points when $n = 250$.

Next, we considered different abstractions of G_{250} obtained by grouping consecutive private signals together. For each abstraction, we computed an equilibrium in the corresponding abstracted game, then determined the payoff of player 1's component of that equilibrium against its nemesis in the full game G_{250}. As Figure 2 shows,

# buckets	2	5	10	25	50	125
payoff	-0.2305	-0.0667	-0.0593	-0.0569	-0.0562	-0.0560

Figure 2: Experiments for G_{250}.

125 buckets are needed to obtain agreement with the value of the

game to four decimal places—something that σ_{250} accomplishes as we showed above. As n gets even larger, we would expect to require even more buckets in our abstraction to obtain a strategy with exploitability as low as that of σ_n. Thus we can potentially obtain a given level of exploitability with a much lower runtime with our projection approach, since the computation required by abstraction-based approaches increases dramatically as n increases, while solving G_∞ and projecting its solution down to G_n requires very little computation.

To put these results in perspective, the game tree for two-player limit Texas hold'em has approximately 9.17×10^{17} states, while recent solution techniques can compute approximate equilibria for abstractions with up to 10^{10} game states (e.g., [10]). Thus the ratio of the number of states in the largest solvable abstraction to the number of states in the full game is approximately 10^{-8}. On the other hand, we saw in G_{250} that we require at least half of the number of states in the full game in our abstraction to compete with the solution generated by the infinite game (assuming we are restricting ourselves to uniform abstractions). Thus, it is conceivable that one can come up with an infinite approximation of Texas hold'em (or one of its subgames) that results in less exploitable strategies than the current strategies obtained by abstraction-based approaches.

In the next section we conduct an investigation of the feasibility of applying the algorithm and extensions developed in this paper to large real-world games, using Texas hold'em as a benchmark.

8.2 Limit Texas hold'em

We ran our algorithm on a game similar to the river endgame of a hand of limit Texas hold'em. In this game, there is an initial pot of ρ, and both players are dealt a private signal from [0,1] according to piecewise linear CDFs F_1 and F_2. Player 1 acts first and can either check or bet \$1. If player 1 checks, then player 2 can check or bet; if he checks the game is over, and if he bets then player 1 can call or fold. If player 1 bets, then player 2 can fold, call, or raise (by 1). If player 1 bets and player 2 raises, then player 1 can either call or fold. Thus, our game is similar to Game 10 in [1], except that we do not assume uniform private signal distributions.

To obtain a wide range of interesting prior distributions, we decided to use the actual prior distributions generated by a high caliber limit Texas hold'em player. Once the river card is dealt in Texas hold'em, there is no more information to be revealed and the game becomes a 1-street game like the game described above (except that in limit Texas hold'em the private signals are dependent[2], and up to three raises are allowed in each betting round). If we assume that the full strategies of both players are known in advance, then when the river card is dealt we can create a distribution over the possible 5-card hand rankings each player could have, given the betting history so far (e.g., the probability he has a royal flush, a full house with 9's over 7's, etc.).

In particular, we obtained the strategies from GS4—a bot that performed competitively in the 2008 AAAI computer poker competition. To test our algorithm, we created a new bot GS4-MIP that plays identically to GS4 on the first three streets, and on the river plays according to our algorithm. Specifically, we assume that both players' hand rankings on the river are distributed assuming they had been following the strategy of GS4 until that point; these determine the private signal distributions.

Given this game model, we developed three different parametric models that we expected equilibria to follow (depending on the private signal distributions at the given hand). This is noteworthy since [1] only considers a single parametric model for their game, and our experiments revealed that if we did not include all three models, our MILFP would sometimes have no solution, demonstrating that all three models are necessary. The models are given in the appendix. It is easy to see that the first model is a US-refinement of the other two. To solve the MILFP, we used CPLEX's MIP solver on a single machine.

Once we solved this simplified game, we used a very naive mapping to transform it to a strategy in the full 3-raise game (e.g., player 1 will put in a second raise with hands in the top half of player 2's re-raise range). Since this mapping was so simple, we suspect that most of the success of the strategy was due to the solution computed by our algorithm.

We ran GS4-MIP against the top five entrants of the 2008 AAAI computer poker competition, which includes GS4. For each pairing, we used 20,000 duplicate hands to reduce the variance. GS4-MIP performed better against 4 of the 5 competitors than GS4 did. In the match between GS4-MIP and GS4, GS4-MIP won at a rate of 0.018 small bets per hand. This is quite significant since most of the top bots in the competition were separated by just 0.02–0.03 small bets per hand overall, and the only difference between GS4 and GS4-MIP is on the river street, which is reached only on some hands. Additionally, GS4-MIP averaged only 0.49 seconds of computation time per hand on hands that went to the river (and 0.25 seconds of computation per hand overall) even though it was solving a reasonably large MIP at runtime[3] (1,000–2,000 rows and several hundred columns). The actual competition allows an average of 7 seconds per hand, so our algorithm was well within the time limit. Perhaps it is the sparsity of the constraints that enabled CPLEX to solve the problems so quickly, as the majority of the constraints are indicator constraints which only have a few non-zero entries.

It is worth noting that our algorithm is perhaps not the most effective algorithm for solving this particular problem; in the discrete case of actual Texas hold'em, the river subgame can be formulated as a linear program (which can probably be solved faster than our MILFP). On the other hand, continuous games, two player general-sum games, and multiplayer games cannot be modeled as linear programs while they can be solved with our approach. Furthermore, the results in the previous subsection show that large finite two-player zero-sum games can sometimes be solved more effectively (both according to runtimes and quality of solutions) by approximating them by a continuous game that is easier to solve, than by abstracting the game and solving a smaller finite game.

9. CONCLUSION

We developed an approach for computing approximate equilibria in large games of imperfect information by solving an infinite approximation of the original game. A key idea is that we include additional inputs in the form of qualitative models of equilibrium strategies (how the signal space should be qualitatively partitioned into action regions). The approach is guaranteed to work even if given a set of qualitative models of which only some are correct. Experiments suggest that this approach can outperform abstraction-based approaches on some games. We construct a game in which only a tiny amount of abstraction can be performed while obtaining

[2]We do not expect the dependence to have a large effect in practice for this game due to the large number of possible private signals. In addition, we have developed an efficient extension of our MILFP to deal with the case of dependent private signals, which can be used if we expect dependence to have a significant effect.

[3]The reason we need to solve the MIP at runtime is that we have to solve a different MIP for each betting sequence up until the river and each set of community cards (in the full game, not in the abstract game). Since there is such a large number of such subgames, it is much easier to just solve them quickly at runtime than to solve them all in advance.

strategies that are no more exploitable than the equilibrium strategies of our infinite approximation. We also showed how to extend our algorithm to the cases of more than two players and continuous private signal distributions. In most of these cases, we presented the first algorithm that provably solves the class of games. Our experiments show that the algorithm runs efficiently in practice and leads to a significant performance improvement in two-player limit Texas hold'em—the most studied imperfect-information game in computer science.

Our approach presents a viable alternative to abstraction-based approaches. This is particularly promising in light of the recently uncovered abstraction pathologies. While in this paper we inferred the infinite approximations of finite games and the parametric models manually, future work on our approach should develop methods for generating them systematically and automatically.

10. REFERENCES

[1] J. Ankenman and B. Chen. *The Mathematics of Poker*. 2006.

[2] C. Archibald and Y. Shoham. Modeling billiards games. *AAMAS*, 2009.

[3] D. Billings, et al. Approximating game-theoretic optimal strategies for full-scale poker. *IJCAI*, 2003.

[4] J. Bisschop. *AIMMS—Optimization Modeling*. 2006.

[5] L. Blumrosen, et al. Auctions with severely bounded communication. *JAIR*, 2007.

[6] X. Chen and X. Deng. Settling the complexity of 2-player Nash equilibrium. *FOCS*, 2006.

[7] K. Etessami and M. Yannakakis. On the complexity of Nash equilibria and other fixed points. *FOCS*, 2007.

[8] D. Fudenberg and J. Tirole. *Game Theory*. 1991.

[9] S. Ganzfried and T. Sandholm. Computing equilibria in multiplayer stochastic games of imperfect information. *IJCAI*, 2009.

[10] A. Gilpin, et al. Gradient-based algorithms for finding Nash equilibria in extensive form games. *WINE*, 2007.

[11] D. Koller, et al. Efficient computation of equilibria for extensive two-person games. *GEB*, 1996.

[12] H. Kuhn. Simplified two-person poker. *Contributions to the Theory of Games*, 1950.

[13] T. Sandholm and A. Gilpin. Sequences of take-it-or-leave-it offers: Near-optimal auctions without full valuation revelation. *AAMAS*, 2006.

[14] T. Sandholm, et al. Mixed-integer programming methods for finding Nash equilibria. *AAAI*, 2005.

[15] T. Sandholm and V. Lesser. Leveled commitment contracts and strategic breach. *GEB*, 2001.

[16] S. Singh, et al. Computing approximate Bayes-Nash equilibria in tree-games of incomplete information. *EC*, 2004.

[17] N. Stein, et al. Separable and low-rank continuous games. *International Journal of Game Theory*, 2008.

[18] J. Vielma and G. Nemhauser. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *IPCO*, 2008.

[19] Y. Vorobeychik and M. Wellman. Stochastic search methods for Nash equilibrium approximation in simulation-based games. *AAMAS*, 2008.

[20] K. Waugh, et al. Abstraction pathologies in extensive games. *AAMAS*, 2009.

[21] M. Zinkevich, et al. Regret minimization in games with incomplete information. *NIPS*, 2007.

APPENDIX

In this section we present the parametric models used for our experiments in Section 8.2 on limit Texas hold'em.

The first parametric model, shown in Figure 3, is identical to the model presented in [1] (with the thresholds renamed). For player 1, the action before the hyphen specifies the first action taken in the betting round, and the action after the hyphen specifies the next action taken if the betting gets back to him. A *bluff* denotes a bet with a bad hand (where the player betting is hoping the other player folds). For player 2, the first action listed denotes the action taken when player 1 bets, and the second action (after the slash) denotes the action taken when player 1 checks. The second parametric model, shown in Figure 4, is identical to the first model except that threshold i is shifted above threshold d. In the third parametric model (Figure 5), player 1 only checks when he is first to act (and never bets).

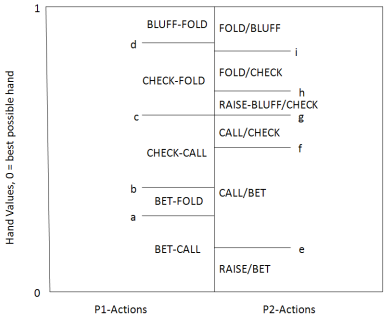

Figure 3: First parametric model.

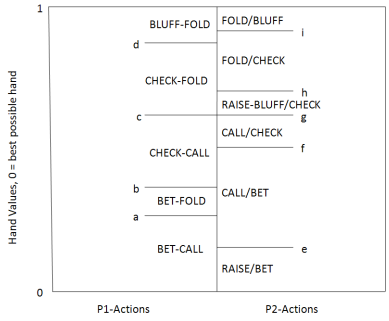

Figure 4: Second parametric model.

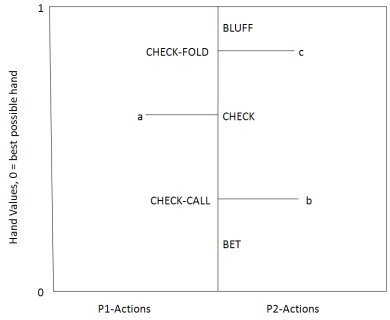

Figure 5: Third parametric model.