Algorithms for Rationalizability and CURB Sets

Michael Benisch, George Davis and Tuomas Sandholm
School of Computer Science
Carnegie Mellon University
{mbenisch, gbd, sandholm}@cs.cmu.edu

Abstract

Significant work has been done on computational aspects of solving games under various solution concepts, such as Nash equilibrium, subgame perfect Nash equilibrium, correlated equilibrium, and (iterated) dominance. However, the fundamental concepts of rationalizability and CURB (Closed Under Rational Behavior sets have not, to our knowledge, been studied from a computational perspective. First, for rationalizability we describe an LP-based polynomial algorithm that finds all strategies that are rationalizable against a mixture over a given set of opponent strategies. Then, we describe a series of increasingly sophisticated polynomial algorithms for finding all minimal CURB sets, one minimal CURB set, and the smallest minimal CURB set. Finally, we give theoretical results regarding the relationships between CURB sets and Nash equilibria, showing that finding a Nash equilibrium can be exponential only in the size of the smallest CURB set. We show that this can lead to an arbitrarily large reduction in the complexity of finding a Nash equilibrium. On the downside, we also show that the smallest CURB set can be arbitrarily larger than the supports of the enclosed Nash equilibrium.

Introduction

For multi-agent systems, game-theoretic solution concepts help agents choose strategies, help modelers predict system states, and help mechanism designers guarantee properties of the games they create. Significant attention has been given to algorithms for finding equilibria according to the solution concepts of subgame perfect Nash equilibrium (e.g., minimax search and α-β-pruning), Nash equilibrium (Lemke & Howson 1964; Porter, Nudelman, & Shoham 2004; Sandholm, Gilpin, & Conitzer 2005), correlated equilibrium (Gilboa & Zemel 1989), and more recently, iterative dominance (Knuth, Papadimitriou, & Tsitsiklis 1988; Conitzer & Sandholm 2005a) and related concepts (Conitzer & Sandholm 2005b).

Nash equilibrium (under which no agent has incentive to deviate from its mixed strategy given that the other agents do not deviate from theirs) remains the most important point-valued solution concept. However, it has been recently shown that finding a Nash equilibrium, even in two player normal-form games, is PPAD-complete (Chen & Deng 2005), suggesting that no polynomial-time algorithms exist for the problem.

However, there are other fundamental solution concepts that have certain known advantages over Nash equilibrium, and—as we will show—solutions according to those concepts can be found in polynomial time even in the worst case. Specifically, we will study the concept of *rationalizability* and the concept of the *minimal Closed Under Rational Behavior (CURB) set* (Basu & Weibull 1991) for two-person normal-form games. A game can have multiple Nash equilibria, but each of those is a point, i.e., a strategy profile. In contrast, in these concepts a solution is a *set* of strategies for each player.

The notion of rationalizability was introduced in (Pearce 1984) and (Bernheim 1984). It has by now been used as a robust solution concept in game theory, and in applications such as auctions (e.g., (Battigalli & Siniscalchi 2003) (Cho 2005) (Dekel & Wolinsky 2001)). Its main insight is that rationality restricts players from ever playing strategies that are not best responses given the beliefs they hold about their opponents; strategies that are not best responses to any belief about opposing strategies are not *rationalizable*. Formally, a strategy is rationalizable if and only if there exists some mixture over opponent strategies for which it is a best response.

Rationalizability is defined from one agent's perspective. CURB sets extend the same idea to multiagent settings. In short, a CURB set consists of a set of pure strategies for each agent so that all of the agents rationalizable strategies are included, given that the other agents only mix over their rationalizable pure strategies. In other words, a CURB set, S, is defined as a set of strategies that contains the best responses to any mixture over itself: if S is CURB and players believe that no strategy outside of S will be played with positive probability by their opponents, then such strategies will indeed not be played by rational players. Formally a CURB set is defined by the following operators.

- $\boldsymbol{\beta_i(m)}$: a function that returns player $\boldsymbol{i}$'s best responses to the mixed strategy $\boldsymbol{m}$.
- $\boldsymbol{M(S)}$: a function that returns all possible mixtures with supports in $\boldsymbol{S}$.
- $\boldsymbol{\beta_i(S)}$: a function that returns player $\boldsymbol{i}$'s best responses to any mixture with supports in $\boldsymbol{S}$,

$$\beta_i(S) = \bigcup_{m \in M(S)} \beta_i(m)$$

- $\boldsymbol{\beta(S)}$: the Cartesian product of the sets $\boldsymbol{\beta_i(S)}$ over all players, $\boldsymbol{i}$.

$$\beta(S) = \prod_i \beta_i(S)$$

Under this notation, a set, S, is CURB if $\beta(S) \subset S$. The entire game is trivially CURB by this definition. (Basu & Weibull 1991) distinguish a *minimal* CURB set as a CURB set that does not contain any CURB subsets.

The minimal CURB set solution concept has been motivated from several perspectives, including the following:

- As is well known, nonstrict mixed Nash equilibria can be highly unstable because a player is indifferent between (some of) his pure strategies. Strict Nash equilibria would be stable but many games lack such equilibria. Minimal CURB sets are the "nearest set-valued generalization of strict Nash equilibria" (it is the smallest set of strategies that includes all ways of choosing among the indifferences) and a solution according to this concept is guaranteed to exist (Basu & Weibull 1991) [1].
- Any CURB set can be viewed as a subspace of strategies within which any best-response dynamic (even a best-response dynamic of mixed strategies) will stay within. CURB sets have thus been examined as a solution concept that describes the strategy subspace where iteratively adapting agents will eventually settle (e.g., (Hurkens 1995)).
- More recently, Voorneveld *et.al.* have enumerated properties of minimal CURB sets that illustrate the advantages of set-based solution concepts over point-valued concepts such as Nash equilibria (Voorneveld, Kets, & Norde 2005).

A fast technique for finding CURB sets in extensive form games by utilizing their additional structure has been presented (Pruzhansky 2003). However, the literature to date suggests that finding minimal CURB sets has been prohibitively complex in normal form games (e.g., (Pruzhansky 2003; Voorneveld, Kets, & Norde 2005)). We present, to our knowledge, the first computational treatment of CURB sets in normal form games. We show that the complexity of finding CURB sets is polynomial even in the worst case.

The rest of the paper is organized as follows. We present and analyze a family of algorithms which compute, for a two player normal form game, in time polynomial in the total number of pure strategies, the set of all rationalizable strategies, all minimal CURB sets, a single minimal CURB set, and the smallest minimal CURB set. Finally, we discuss additional applications of our results, including the potential of finding minimal CURB sets to bound the computation involved in finding Nash equilibria.

Finding Rationalizable Strategies

Finding strategies which are rationalizable under specific beliefs (i.e. against a mixture over a given set of opponent strategies) is a problem of interest in its own right, and plays a central role in our computation of CURB sets. In this section, we present a polynomial time procedure, `all_rationalizable`, which accomplishes this task.

In 2-player games, iterative removal of strategies strictly dominated by (potentially mixed) strategies finds all rationalizable strategies. However, it cannot distinguish between strategies that are only globally rationalizable and strategies rationalizable under specific beliefs. CURB sets are defined with regard to internally consistent beliefs. Therefore, iterated dominance does not support algorithms for CURB sets. The technique presented in this section does (our `all_rationalizable` function conditions on given beliefs).

The algorithm below is for the row player. The column player's algorithm is symmetric. The parameters to the procedure are a set of row player strategies to consider, S_r, a set of column player strategies they may be played against, S_c, and the row-player's utility function, u_r. For each row strategy, $s_r \in S_r$, a linear feasibility problem (LFP) (i.e., a linear program with no objective) is constructed to find a mixture, p_{s_c}, over column player strategies such that s_r is the row player's best response. The constraints of the LFP ensure that the mixture is valid (sums to 1) and that the row player's utility by playing s_r against p_{s_c} is greater than or equal to that of any other strategy in S_r. If and only if the LFP has a feasible solution, s_r is added to the set of rationalizable strategies.

procedure `all_rationalizable`(S_r, S_c, u_r)
  $S_r^* \leftarrow \emptyset$
  for each row strategy, $s_r \in S_r$ **do**
    let `rationalizable` $\leftarrow \top$ iff there exists a feasible solution to the following linear feasibility problem:
    find p_{s_c} **such that**

$$\sum_{s_c} p_{s_c} = 1 \quad (1)$$

$$(\forall s'_r \in S_r/\{s_r\}) \quad \sum_{s_c} p_{s_c} u_r(s_r, s_c) \geq \sum_{s_c} p_{s_c} u_r(s'_r, s_c) \quad (2)$$

    if `rationalizable` **then** $S_r^* \leftarrow S_r^* \cup s_r$
  return S_r^*

The computational complexity of the procedures described in this paper depend on the total number of strategies in the game, which we will denote by n, and the complexity of solving a linear feasibility problem where the number of variables and the number of constraints are bounded by n, which we will denote as $\texttt{LFP}(n)$. Linear feasibility problems can be solved in low-order polynomial time even in the worst case. They are no slower to solve than linear programs (the fastest known algorithms for LFPs are faster, in the worst case, than the fastest known linear programming algorithms (Ye 2006)) because linear program solving involves solving a linear feasibility program as the first phase to find a feasible solution, after which the linear program solver needs to still improve that solution to reach an optimum. (In the experiments, we solve the LFP using the simplex algorithm, which has worst-case exponential com-

[1] A similar set-valued concept is the Set-Nash of Lavi and Nisan (Lavi & Nisan 2005), which has been examined for mechanism design but not from an algorithmic perspective. That concept differs from CURB sets in that only strategies which are rationalizable against *pure* opponent strategies within the set are included.

plexity but is known to outperform polynomial linear programming algorithms in practice.)

Prop. 1. `all_rationalizable` *returns all rationalizable strategies, and nothing else. It is* $O(n) \times \texttt{LFP}(n)$[2].

Finding CURB Sets

We now turn our attention to the problem of finding CURB sets. The procedure below finds a minimal set of strategies that both 1) contains a given seed strategy, s_r, and 2) is CURB. (Note that the returned set is not necessarily a minimal CURB set.) It alternates between the players, calling the `all_rationalizable` procedure to identify strategies that have become rationalizable via the addition of opponent strategies. If an iteration passes without strategies being added, the algorithm has converged.

procedure `min_containing_CURB`$(s_r, \langle S_r, S_c, u\rangle)$
  $S_r^* \leftarrow \{s_r\}$, $S_c^* \leftarrow \emptyset$
  `converged` $\leftarrow \bot$
  while $\neg$`converged` **do**
    `converged` $\leftarrow \top$
    for $(p, o) \in [(c, r), (r, c)]$ **do**
      $S'_p \leftarrow$ `all_rationalizable`$(S_p \setminus S_p^*, S_o^*, u_p)$
      if $S'_p \neq \emptyset$ **then**
        `converged` $\leftarrow \bot$
        $S_p^* \leftarrow S_p^* \cup S'_p$
  return sub-game, $G' = \langle S_r^*, S_c^*, u\rangle$

Prop. 2. `min_containing_CURB` *is* $O(n^2) \times \texttt{LFP}(n)$.

Thm. 1. *The* `min_containing_CURB` *algorithm is* correct*, that is, the returned set, S^*, is a minimal set of strategies that both 1) contains the given seed strategy, s_r, and 2) is CURB.*

Proof. We show that S^* is CURB by considering its state after `min_containing_CURB` converges. The convergence of the algorithm implies that no strategies outside of S^* are rationalizable with respect to S^*. Therefore, $\beta(S^*) \subset S^*$, and S^* is CURB.

To prove that S^* is the minimal containing CURB set of s_r (it contains no CURB subsets that include s_r), we will use induction on the strategies added to S^*.

- **Base Case:** Initially S^* contains only s_r and $\beta_c(s_r)$. At this point, S^* is trivially the minimal containing CURB set of s_r because we cannot remove s_r and removing the column player's best response to s_r breaks the CURB property.
- **Inductive Step:** Each time a new strategy s^* is added to S^* it is necessarily a best response to some mixture, $m \in M(S^*)$, over the strategies already contained in S^*. Since strategies are never removed from S^* during the execution of MCC, m will remain a valid mixture. Therefore, no new strategy s^* can be removed from S^* without breaking the CURB property. □

[2]We omit some proofs due to limited space.

We will now present three algorithms which use the above procedure to detect minimal CURB sets in a game. To facilitate understanding of these algorithms, we first present the following results regarding CURB set structure.

Thm. 2. *If each of two intersecting strategy sets is CURB, then their intersection is also CURB.*

Proof. Consider two CURB sets S_A and S_B with the non-empty intersection S_I. For any mixture over strategies in S_I belonging to (without loss of generality) the row player, there exists a pure strategy which is column player's best response, s_c^*. Because S_A is CURB and also contains the row mixture, $s_c^* \in S_A$; likewise $s_c^* \in S_B$. Therefore s_c^* is within their intersection, S_I. □

Since the intersection of two CURB sets must be CURB and contained in both sets, we have the following.

Cor. 1. *Minimal CURB sets cannot overlap (i.e., share any rows or columns).*

Cor. 2. *Each row strategy belongs to at most one minimal CURB set.*

Finding All Minimal CURB Sets. The broadest query one can make regarding the minimal CURB set structure of a game is to find all minimal CURB sets. For one, this is useful in the adaptive agent context, to identify regions of strategy space into which learning agents may settle (e.g., (Hurkens 1995)). The `all_MC` procedure described below answers this query.

To determine all of the minimal CURB sets, the `min_containing_CURB` procedure can be executed with each row strategy, in turn, as a seed. For the first seed, this call is made with the parameter $\langle S_r, S_c, u\rangle$ indicating the entire game. However, the result above regarding CURB intersections shows that each of the CURB sets returned by `min_containing_CURB` must contain the minimal containing CURB sets of each of its members. Therefore, we can accelerate future calls by maintaining a map between each strategy and the smallest CURB set in which it has been discovered so far. We use the subgame restricted to that strategy set as the parameter $\langle S_r, S_c, u\rangle$ when that strategy is used as the seed. Whenever such a call results in a smaller CURB set, we eliminate the previous CURB set from consideration, as it cannot be minimal. Once each strategy has been used as a seed, `all_MC` terminates and returns the CURB sets that have not been eliminated.

From the corollary above, the fact that `all_MC` executes `min_containing_CURB` on each row strategy, and the fact that no superset CURB sets remain, we have the following.

Prop. 3. `all_MC` *finds all minimal CURB sets, and nothing else. It is* $O(n^3) \times \texttt{LFP}(n)$.

Finding One Minimal CURB Set. Rather than finding all minimal CURB sets in a game, it may be desirable to quickly find any single minimal CURB set. To complete this query, we can use the `min_containing_CURB` procedure with a random strategy as the seed. Since the discovered CURB set might not be minimal, we recur within it by choosing as a seed a contained strategy that has not yet been used as a seed.

We repeat this until all strategies in the current set have been used as seeds, at which point we terminate and return the remaining set. This constitutes the one_minimal_CURB algorithm.

If the game has more than one CURB set, one_MC will be faster than all_MC because it will never leave the smallest CURB set within which it ever chooses a seed. The exact speed of one_MC depends on the first seed chosen. If it happens to be in a small CURB set, one_MC runs faster. In the worst case where the entire game is the only CURB set, one_MC executes all of the same steps as all_MC. Due to this fact, and the fact that the returned set has been confirmed to be the minimal containing CURB set for all strategies within it, we have the following.

Prop. 4. one_MC *returns a minimal CURB set, and is* $O(n^3) \times LFP(n)$.

Finding the Smallest Minimal CURB Set. As a different type of query, one may be interested in finding a *smallest* minimal CURB set. This is important, for example, if the CURB set is used for future computations (e.g., for Nash equilibrium finding as we will discuss later in the paper) and the complexity of those future computations increases with the size of the CURB set.

We find the smallest minimal CURB set using a pseudo-parallelization of all_MC, wherein we expand a candidate set only when it is one of the smallest currently available. First, we construct a candidate set for each row strategy containing only that strategy. We insert the sets into a priority queue where sets containing the fewest strategies receive highest priority. We repeatedly pop the smallest candidate set from the queue and add all the rationalizable strategies to that set using all_rationalizable. If new strategies were added, the resulting set is inserted back into the queue, and prioritized based on its new size. The algorithm terminates when a candidate set is removed from the queue that fails to admit any new rationalizable strategies. That set is returned, and it is a smallest minimal CURB set. We call this algorithm small_MC.

The smallest CURB set in any game must be minimal, and all other candidates in the queue are of size greater or equal to the returned set. The complexity of small_MC is bounded by the total number of strategies in the smallest CURB set, which we denote by n_{SC}. Since each call to all_rationalizable must add at least one strategy until small_MC terminates, we have the following.

Prop. 5. small_MC *returns a minimal CURB set that is a smallest minimal CURB set in the game, and it is* $O(n_{SC}\ n^2) \times LFP(n)$.

Experimental Results

We examined the runtime performance of our algorithms on two game distributions provided by the *GAMUT* instance generators (Nudelman *et al.* 2004): *random games*, and *covariant games*. Figure 1 shows how each minimum CURB finding algorithm scales with game size on a dataset of over 1000 random, square normal form games with total number of strategies n between 20 and 80. small_MC is faster than all_MC. This is consistent with their O-complexities, considering that many random games have small CURB sets. While the O-complexity of one_MC and all_MC is the same, experimentally one_MC is faster because it only needs to find one minimal CURB set. (On any game with more than one minimal CURB set, one_MC is faster than all_MC.)

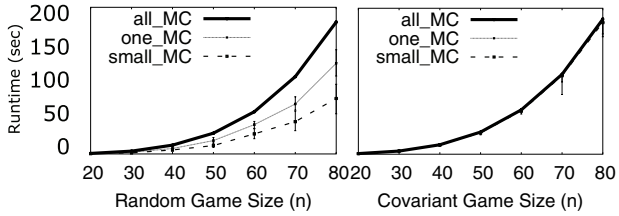

Figure 1: Scalability of our algorithms in game size (the curves on the right overlap).

We observed that the performance illustrated on random games in Figure 1 was typical of that of many other instance distributions provided by the *GAMUT* instance generators as well. However, to show potentially differing performance, we also experimented with the covariant game class, in which payoffs for both players are drawn from the same distribution with a specified covariance. In our experiments we used a covariance parameter of -0.5. That class and that setting have been shown to be particularly challenging for Nash equilibrium finding (with the Lemke-Howson algorithm and the Porter-Nudelman-Shoham algorithm) (Porter, Nudelman, & Shoham 2004). Figure 1 shows that the all_MC algorithm scales similarly on random and covariant games, while the other two algorithms lose their relative speed advantages when applied to the covariant class.

Most random games have small smallest CURB sets (in fact, often sets of size 2, i.e., pure-strategy equilibria), and those that do not, tend to have very large smallest CURB sets (Figure 2). On the other hand, covariant games tend to have almost no small smallest CURB sets and often have large smallest CURB sets. (This is consistent with the observed hardness of these games for support enumeration-based Nash equilibrium finding algorithms that try to find equilibria with small supports first (Porter, Nudelman, & Shoham 2004).) The disparity explains the lowered performance on covariant games for both the two minimal CURB finding algorithms, which are affected by the size of the smallest minimal CURB set.

To better understand how the minimal CURB finding algorithms scale with the size of the smallest CURB set, we bucketed the $n = 20$ games from above according to the size of the smallest CURB set. (For $n = 40$ games the buckets for medium sized small CURB sets were nearly empty, making it impossible for us to estimate mean runtimes with sufficient accuracy.) Figure 3 plots the average runtime for each bucket. On games with very small CURB sets, small_MC is fastest, but it is outperformed by both one_MC and all_MC as the smallest CURB set grows. The surprising average-case efficiency of the latter two algorithms is due to their leveraging of information across calls

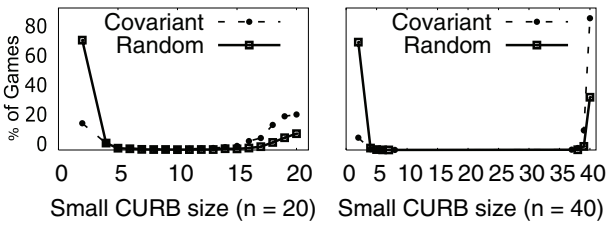

Figure 2: Distribution of Smallest CURB set size in random and covariant ($r = -0.5$) games, where $n = 20$ and $n = 40$ (3,000 games for each distribution and value of n).

to `min_containing_CURB` with different seeds. Because `small_MC` performs all the searches in parallel, this information is unavailable.

The results shown in Figure 1 are unintuitive because most random and covariant games have tiny CURB sets, which are the special case under which `small_MC` outperforms `one_MC`. If smallness of CURB sets were known in advance, one could use `small_MC` to accomplish the task of `one_MC` more efficiently, but generally this is slower. To get the best of both, one can timeslice between the two algorithms. The runtime of this hybrid is at most twice the runtime of the faster of the two (plus the length of a slice and the slicing overhead).

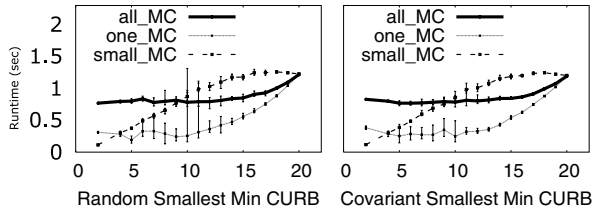

Figure 3: Average runtime on games were $n = 20$, with varying smallest CURB set sizes.

CURB Sets and Nash Equilibria

Both minimal CURB sets and Nash equilibria model strategy subspaces which are mutually reinforced given the rationality of agents and their common knowledge. A CURB set with only one row and one column strategy is *necessarily* a pure strategy Nash equilibrium. Furthermore, any minimal CURB set contains the supports for at least one Nash equilibrium in mixed strategies (Basu & Weibull 1991). We observe that this result suggests a secondary use for finding minimal CURB sets: our algorithms can be used to preprocess a game so that a Nash equilibrium finding algorithm needs to only operate on a minimal CURB set rather than the entire game. This can yield an arbitrarily large reduction of the search space, as Theorem 3(a) will show.

The most common prior preprocessing technique for Nash equilibrium finding, iterated removal of dominated strategies, attempts to eliminate strategies that cannot be played with any probability in any Nash equilibrium (Knuth, Papadimitriou, & Tsitsiklis 1988), (Gilboa, Kalai, & Zemel 1993). The same is true of a recent preprocessing technique, the generalized eliminability method (Conitzer & Sandholm 2005b). One comparative advantage of minimal CURB set-based elimination is that it can eliminate strategies that are played in some equilibria, while guaranteeing that the resulting set still contains the supports of at least one equilibrium.

Our CURB set-based preprocessor can reduce search space size by an arbitrary amount (i.e. to a CURB set of almost any dimension) even on games where no prior preprocessing technique can eliminate anything.

Thm. 3. *For any $r \geq 2$, $c \geq 2$ and r', c' such that $1 < r' \leq r$ and $1 < c' \leq c$ (or $r' = c' = 1$), there exists at least one $r \times c$ normal-form game, where*

a) the game has an $r' \times c'$ smallest CURB set.

b) iterated elimination of dominated strategies (even domination by mixed strategies) cannot eliminate any strategies,

*c) the recent recursive preprocessing technique (that **can** eliminate strategies that belong to some equilibrium as long as some other equilibrium remains) (Conitzer & Sandholm 2006) cannot eliminate any strategies, and*

d) if $r' + c' \leq \frac{r+c}{2}$ (i.e., the smallest CURB set is not huge), the general eliminability method (Conitzer & Sandholm 2005b) cannot eliminate any strategies. (Even if the smallest CURB set is huge, the general eliminability technique cannot eliminate more strategies than our CURB set-based technique.)

Proof. We first present the following family, Γ, of games. Let $\Gamma_{r'c'}$ denote such a game of size $r' \times c'$. The following generator produces such a game where $r', c' \geq 2$. Assign the payoffs $u(s_{r_1}, s_{c_1}) = u(s_{r_2}, s_{c_2}) = (0, 1)$ and $u(s_{r_1}, s_{c_2}) = u(s_{r_2}, s_{c_1}) = (1, 0)$. Then, for $i \in [2, r' - 1]$, set $u(s_{r_{i+1}}, s_{c_1}) = (\frac{r'-i}{r'}, 1)$ and $u(s_{r_{i+1}}, s_{c_2}) = (\frac{i}{r'}, 0)$. Next, for $j \in [2, c' - 1]$, set $u(s_{r_1}, s_{c_{j+1}}) = (0, \frac{c'-j}{c'})$ and $u(s_{r_2}, s_{c_{j+1}}) = (1, \frac{j}{c'})$. For all payoffs still unassigned, set $u(s_{r_i}, s_{c_j}) = (-i, -j)$.

For example, the game $\Gamma_{3,4}$ is as follows.

$\mathbf{\Gamma_{3,4}}$	s_{c_1}	s_{c_2}	s_{c_3}	s_{c_4}
s_{r_1}	0,1	1,0	0, $\frac{1}{2}$	0, $\frac{1}{4}$
s_{r_2}	1,0	0,1	1, $\frac{1}{2}$	1, $\frac{3}{4}$
s_{r_3}	$\frac{1}{3}$, 1	$\frac{2}{3}$,0	-3,-3	-3,-4

Any game generated in this way has a Nash equilibrium where the row player mixes between his first two strategies and the column player mixes among all his strategies. It also has an equilibrium where the column player mixes between his first two strategies and the row player mixes among all his. Thus, every strategy in $\Gamma_{r'c'}$ is part of some equilibrium. Additionally each column strategy is a best response to a mixture over the first two row strategies (and, to any column strategy, one of those two is a best response), and vice versa. Thus, $\Gamma_{r'c'}$ has a single minimal CURB set and it includes the entire game.

We now construct an $r \times c$ game with a minimally CURB $r' \times c'$ subset by putting the game $\Gamma_{r'c'}$ in the top left and the game $\Gamma_{(r-r')(c-c')}$ in the bottom right. All other payoffs are set to negative random noise. The resulting game is irreducible by (iterated) dominance and by general eliminability because every strategy participates in some Nash equilibrium. The game is irreducible by the recursive preprocessor because the row player's payoffs are distinct within each column and the column player's payoffs are distinct within each row. (If $\Gamma_{r'c'}$ is the *smallest* minimal CURB set and $r' + c' > \frac{r+c}{2}$, we instead construct a game by starting with $\Gamma_{r'c'}$ and including additional row and column strategies that are neither dominated nor part of any minimal CURB set. It is easy to generate such strategies, but they might not be part of any Nash equilibrium. Thus the general eliminability method may be able to eliminate those additional strategies.) □

However, three factors curb the promise of minimal CURB set algorithms as powerful preprocessors for Nash equilibrium finding. First, the fastest Nash equilibrium finding algorithms, while requiring exponential time in the worst case, tend to run faster than (at least the current implementations of) the CURB set finding algorithms on many instance distributions. Second, by Theorem 3(a), the smallest CURB set can be arbitrarily large (up to the size of the entire game, in which case the preprocessor does not eliminate any strategies from consideration). Third, even after the smallest minimal CURB set has been identified, the remaining search space (CURB set size) can be arbitrarily larger than the size of the supports of a contained Nash equilibrium:

Thm. 4. *A Nash equilibrium with supports consisting of two strategies for each player can be the only Nash equilibrium in an arbitrarily large minimal CURB set.*

Proof. Consider the following family of games that contain large minimal CURB sets and small-support equilibria. For any integer $k > 0$, we define the game Ω_k as follows. As in the previous proof, assign the payoffs $u(s_{r_1}, s_{c_1}) = u(s_{r_2}, s_{c_2}) = (0, 1)$ and $u(s_{r_1}, s_{c_2}) = u(s_{r_2}, s_{c_1}) = (1, 0)$. Then, for $i \in [3, 2 + k]$,

- $u(s_{r_i}, s_{c_1}) = (-\infty, \epsilon)$, $u(s_{r_1}, s_{c_i}) = (\epsilon, -\infty)$,
- $u(s_{r_i}, s_{c_i}) = (0, 0)$,
- $u(s_{r_i}, s_{c_{i-1}}) = (1 + \epsilon, 0)$, $u(s_{r_{i-1}}, s_{c_i}) = (0, 1 + \epsilon)$, and
- for all $j > i + 1$ and $j \leq 2 + n$, $u(s_{r_i}, s_{c_j}) = (0, -\infty)$, and $u(s_{r_j}, s_{c_i}) = (-\infty, 0)$

For example, the game Ω_2 is as follows.

Ω_2	s_{c_1}	s_{c_2}	s_{c_3}	s_{c_4}
s_{r_1}	0,1	1,0	ϵ,$-\infty$	ϵ,$-\infty$
s_{r_2}	1,0	0,1	0,1+ϵ	0,$-\infty$
s_{r_3}	$-\infty$,ϵ	1+ϵ,0	0,0	0,1 + ϵ
s_{r_4}	$-\infty$,ϵ	$-\infty$, 0	1+ϵ,0	0,0

Prop. 6. *Ω_k has a single minimal CURB set and it includes the entire game.*

Prop. 7. *In Ω_k, the only Nash equilibrium is the the strategy profile where s_{r_1}, s_{r_2}, s_{c_1} and s_{c_2} are each played with probability $\frac{1}{2}$.*

This completes the proof of the theorem. □

The above results mean that minimal CURB set algorithms do not always help in Nash equilibrium finding because the smallest CURB set can be arbitrarily large (Theorem 3(a)) and it can be arbitrarily loose around an enclosed Nash equilibrium (Theorem 4). However, on any game instance that has a small CURB set *or* a relatively tight minimal CURB set[3], and on which standard equilibrium-finding algorithms run very slowly, our technique should yield a drastic speed improvement for Nash equilibrium finding (the worst-case complexity of our preprocessor is polynomial, while any equilibrium-finding algorithm has super-polynomial run-time (unless PPAD=P)).

Furthermore, the existence of the polynomial-time algorithm for detecting a game's smallest CURB set (`small_MC`) allows us to offer the following theoretical result of potential general interest.

Thm. 5. *The complexity of finding a single Nash equilibrium for a two-player normal form game is super-polynomial only in the size of the game's smallest CURB set. (If PPAD=P, then it is not super-polynomial in anything.)*

Conclusions and Future Research

We presented the first algorithms for rationalizability and CURB sets, two important set-valued solution concepts for normal-form games. Our algorithms find all rationalizable strategies for a given support of opponent's strategies, all minimal CURB sets (`all_MC`), one minimal CURB set (`one_MC`), and a smallest minimal CURB set (`small_MC`), all in polynomial time. The CURB algorithms use techniques such as dovetailing with a priority queue, and exploiting information across overlapping CURB sets, to further improve speed.

Experiments on random games showed that `small_MC` is the fastest, `one_MC` is second, and `all_MC` is the slowest of the three. On covariant games, the speed advantage of the former two disappears. The runtime of the algorithms is affected by the size of the smallest CURB set. On games with small CURB sets, `small_MC` performed significantly better than the others, but the others (especially `one_MC`) performed better on games containing only large CURB sets.

We also examined the potential for using our algorithms as preprocessors for Nash equilibrium finding algorithms. We proved that our technique can eliminate an arbitrarily large portion of the game from consideration while guaranteeing that the remaining strategies contain a Nash equilibrium. This is the case even on games where prior preprocessing techniques are powerless.

On the downside, we showed that the smallest CURB set can be arbitrarily large and/or arbitrarily loose. Furthermore,

[3] If the game has a relatively tight CURB set, a Nash equilibrium can be found quickly by enumerating strategies of the CURB to be left out from the supports.

on many distributions, current Nash equilibrium finding algorithms run faster on average than the CURB set algorithms.

However, we showed that finding a Nash equilibrium is super-polynomial *only* in the size of the smallest CURB set of the game. Taken together with our CURB set algorithms that are polynomial even in the worst case, and the fact that Nash equilibrium finding algorithms are super-polynomial in the worst case (unless PPAD=P), this indicates that our preprocessing techniques will yield a drastic speed improvement on hard games that have a small *or* a relatively tight minimal CURB set. As we showed, games with these two properties do exist; future research involves determining whether such games occur naturally.

The CURB set definition is for any number of players. In this paper we considered the two-player case. For a larger number of players, the obstacle is finding rationalizable strategies quickly as a subroutine. The mathematical program we use is of degree d where the number of players is $d+1$; for two players it is an LFP but for three players it is already quadratic. Our CURB set algorithms can be trivially generalized to any number of players. Even the generalized versions only make a polynomial number of calls to the `all_rationalizable` subroutine. Therefore, if future research finds polynomial algorithms for finding rationalizable strategies for more than two players in polynomial time, then our generalized CURB set algorithms are also polynomial time.

Acknowledgments

This material is based upon work supported by National Science Foundation ITR grant 0205435, IGERT grant 9972762, and IIS grants 0121678 and 0427858, as well as Office of Naval Research grant N00014-02-1-0973, and a Sloan Fellowship. Additional support at Carnegie Mellon University was provided by the Agent-Mediated Electronic Marketplaces Laboratory, the Center for Computational Analysis of Social and Organizational Systems, and the e-Supply Chain Management Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the National Science Foundation, the Office of Naval Research, or the U.S. government. We thank Vincent Conitzer and Andrew Gilpin for their helpful input and guidance.

References

Basu, K., and Weibull, J. W. 1991. Strategy subsets closed under rational behavior. *Economics Letters* 36(2):141–146.

Battigalli, P., and Siniscalchi, M. 2003. Rationalizable bidding in first-price auctions. *Games and Economic Behavior* 45(1):38–72.

Bernheim, B. D. 1984. Rationalizable strategic behavior. *Econometrica* 52(4):1007–28.

Chen, X., and Deng, X. 2005. Settling the complexity of 2-player Nash-equilibrium. Technical Report TR05-140, Electronic Colloquium on Computational Complexity.

Cho, In-Koo. 2005. Monotonicity and Rationalizability in a Large First Price Auction. *Review of Economic Studies* 72(4):1031–55.

Conitzer, V., and Sandholm, T. 2005a. Complexity of (iterated) dominance. In *Proceedings of EC'05*, 88–97.

Conitzer, V., and Sandholm, T. 2005b. A generalized strategy eliminability criterion and computational methods for applying it. In *Proceedings of AAAI'05*, 483–488.

Conitzer, V., and Sandholm, T. 2006. A technique for reducing normal form games to compute a Nash equilibrium. In *AAMAS*.

Dekel E., and Wolinsky A. 2001. Rationalizable outcomes of large independent private-value first-price discrete auctions. Northwestern University, Center for Mathematical Studies in Economics and Management Science, Discussion Paper 1308

Gilboa, I., and Zemel, E. 1989. Nash and correlated equilibria: Some complexity considerations. *Games & Economic Behavior* 1:80–93.

Gilboa, I.; Kalai, E.; and Zemel, E. 1993. The complexity of eliminating dominated strategies. *Mathematics of Operations Research* 18(3):553–565.

Hurkens, S. 1995. Learning by forgetful players. *Games and Economic Behavior* 11(1):304–329.

Knuth, D.; Papadimitriou, C.; and Tsitsiklis, J. 1988. A note on strategy elimination in bimatrix games. *Operations Research Letters* 7(3):103–107.

Lavi, R., and Nisan, N. 2005. Online ascending auctions for gradually expiring items. In *Proceedings of SODA'05*.

Lemke, C., and Howson, J. 1964. Equilibrium points of bimatrix games. *Journal of the Society of Industrial and Applied Mathematics* 12:413–423.

Nudelman, E.; Wortman, J.; Shoham, Y.; and Leyton-Brown, K. 2004. Run the GAMUT: A comprehensive approach to evaluating game-theoretic algorithms. In *Proc. of AAMAS*, 880–887.

Pearce, D. G. 1984. Rationalizable strategic behavior and the problem of perfection. *Econometrica* 52(4):1029–50.

Porter, R.; Nudelman, E.; and Shoham, Y. 2004. Simple search methods for finding a Nash equilibrium. In *AAAI*, 664–669.

Pruzhansky, V. 2003. On finding CURB sets in extensive games. *International Journal of Game Theory* 32(2):205–210.

Sandholm, T.; Gilpin, A.; and Conitzer, V. 2005. Mixed-integer programming methods for finding Nash equilibria. In *AAAI*.

Voorneveld, M.; Kets, W.; and Norde, H. 2005. An axiomatization of minimal CURB sets. *International Journal of Game Theory* 33(4):479–490.

Ye, Y. 2006. Improved complexity results on solving real-number linear feasibility problems. *Mathematical Programming* 106(2):339–363.