Theoretical Economics 9 (2014), 313–338 1555-7561/20140313

An algorithm for two-player repeated games with perfect monitoring

Dilip Abreu
Department of Economics, Princeton University

Yuliy Sannikov
Department of Economics, Princeton University

Consider repeated two-player games with perfect monitoring and discounting. We provide an algorithm that computes the set V^* of payoff pairs of all pure-strategy subgame-perfect equilibria with public randomization. The algorithm provides significant efficiency gains over the existing implementations of the algorithm from Abreu et al. (1990). These efficiency gains arise from a better understanding of the manner in which *extreme points* of the equilibrium payoff set are generated. An important theoretical implication of our algorithm is that the set of extreme points E of V^* is *finite*. Indeed, $|E| \leq 3|A|$, where A is the set of action profiles of the stage game.

Keywords. Repeated games, perfect monitoring, computation.

JEL classification. C63, C72, C73.

1. Introduction

The paper develops a new algorithm for computing the set of subgame-perfect equilibrium payoff vectors in repeated games with finitely many actions, perfect monitoring, discounting, and public randomization. This is a very classical setting, which serves as a basis for many applications. Nevertheless, our results suggest that significant improvements in existing computational procedures can be obtained from a better understanding of the structure of equilibria, especially the generation of extreme equilibrium payoff vectors. Besides describing a faster computational algorithm, we also provide a publicly available implementation of our algorithm, which should be useful both to researchers trying to understand the impact of changes in underlying parameters on equilibrium possibilities and to students seeking to develop an understanding of dynamic games and how they "work."

Dilip Abreu: dabreu@princeton.edu
Yuliy Sannikov: sannikov@princeton.edu

We are very grateful to Benjamin Brooks, Rohit Lamba, and Richard Katzwer for excellent research assistance. Brooks and Lamba wrote the initial programs that implement our algorithm. Subsequently Brooks has been responsible for further development of the Matlab programs and the user interface with input from the authors, and Katzwer wrote the stand-alone Java implementation. We would also like to thank a co-editor and anonymous referees. Abreu acknowledges financial support from the National Science Foundation. Sannikov acknowledges financial support from the Alfred P. Sloan Foundation.

DOI: 10.3982/TE1302

Prior work on this topic has as its starting point the algorithm suggested by Abreu et al. (1990), hereafter APS. This is true of the approach presented here also. The APS algorithm works iteratively, starting with the set of feasible payoffs of the stage game W^0. The set of subgame-perfect equilibrium payoffs V^* is found by applying a set operator B to W^0 iteratively until the resulting sequence of sets $W^0, W^1, \ldots, W^{n+1} = B(W^n)$ converges. For a payoff set W, the operator $B(W)$ gives the set of payoffs that can be generated through some action profile a in the current period and using continuation values from W in the next period, while respecting all incentive constraints.

A classic paper by Judd et al. (2003) (to which we refer frequently below as JYC) provides a numerical implementation of the APS algorithm based on linear programming problems. Each set W^n is approximated by its supporting hyperplanes. Such an approximation can be defined by a set of linear inequalities, for a fixed number of directions, that the points of W^n must satisfy. The JYC implementation solves a set of linear programming problems, one for each action profile, to find the supporting hyperplane of the set W^{n+1} for each direction.

Rather than implementing the APS algorithm directly, we simplify it using a better understanding of how the extreme points of the equilibrium payoff set V^* are generated. It turns out that any extreme point of V^* can be generated either by (i) an infinite repetition of some action profile a on the equilibrium path or (ii) some action profile a in the current period and a vector of continuation payoffs for the next period such that the incentive constraint of at least one of the players is *binding*. In the former case, profile a generates a single extreme point of V^*. In the latter case, profile a generates at most four extreme points, all with a constraint of at least one player binding.

While the structure of the set V^* is fairly simple, the generation of the set $W^{n+1} = B(W^n)$ by the APS algorithm can be fairly complicated. Each action profile a can potentially generate very many extreme points of the set W^{n+1}. We modify the APS algorithm by introducing a stronger set operator that keeps at most four points that correspond to each action profile a by focusing on binding incentive constraints. Keeping at most four points per action profile speeds up the algorithm and allows us to represent payoff sets via their extreme points, rather than supporting hyperplanes. Clearly, because our algorithm is based on a stronger operator than B, it cannot converge to a set larger than V^*. We are also able to show that the sequence of sets we generate contains V^* and, therefore, it must converge to V^*.

We compare the running time of our algorithm to that of JYC both theoretically, and experimentally. Theoretically, the JYC algorithm takes considerably more time to perform one iteration than our algorithm. In practice, our algorithm also runs significantly faster than that of JYC. See Section 5 for details.

The most that one could hope for in this endeavor is to, in fact, be able to solve for the equilibrium value set analytically. A novel feature of our work is that we are indeed able to obtain analytical solutions by exploiting what the algorithm reveals about the limit equilibrium *structure*. This allows us to write down a set of simultaneous equations (in the coordinates of the extreme points of V^*) that may be solved to obtain the extreme points analytically in terms of the underlying payoffs of the stage game. This is discussed in Section 6.

This paper is organized as follows. Section 2 reviews background information on repeated games and the APS algorithm. Section 3 presents our algorithm together with a proof of convergence. Section 4 presents several computed examples and compares our algorithm with that of JYC. Section 5 discusses methods to evaluate numerical errors, including *inner approximations* that generate a lower bound on V^* and (as discussed above) methods to solve for the vertices of V^* analytically. A stand-alone implementation of our algorithm with a convenient graphical user interface is available online at www.princeton.edu/econtheorycenter/research-links/dynamic-games-algorithm/.[1]

2. The setting and background

Consider a two-player repeated game with simultaneous-move stage game $G = \{N, (A_i)_{i\in N}, (g_i)_{i\in N}\}$, in which players $N = \{1, 2\}$ have a common discount factor $\delta < 1$. The players observe each other's actions after each repetition of the stage game. Our objective is to construct an algorithm to compute the set V^* of payoff pairs achievable in pure-strategy subgame-perfect equilibria of the repeated game with public randomization. The set V^* is the largest compact *self-generating* set (see Abreu et al. 1986, 1990).[2] To formalize this concept, for any action $a_j \in A_j$, denote by

$$\overline{g}_i(a_j) = \max_{a_i} g_i(a_i, a_j)$$

the maximal one-period payoff that player i can get in response to a_j. Let $h_i(a) = \overline{g}_i(a_j) - g_i(a)$. For any compact set $X \subset \mathcal{R}^2$, denote the worst punishment for each player i by

$$P_i(X) = \min\{x_i \mid (x_1, x_2) \in X \text{ for some } x_j\}.$$

Definition 1. A point v is generated by the set X if there is an action pair $a \in A$ in the current period and a pair of continuation values $w \in X$ such that

$$\begin{aligned} &v = (1-\delta)g(a) + \delta w \quad \text{(adding up)} \\ &\delta(w - P(X)) \geq (1-\delta)h(a) \end{aligned} \tag{IC}$$

A set X is called *self-generating* if each extreme point of X can be generated by X.

The incentive constraints (IC) guarantee that each player prefers to follow her equilibrium action rather than to deviate and receive her worst equilibrium punishment. In what follows, we denote the worst punishments from the set V^* by $\underline{v} = P(V^*)$. APS propose an algorithm to compute the set V^*, which is based on the operator B defined as

$$B(W) = \text{co}\big\{v \mid \exists w \in W, a \in A \text{ s.t. } v = (1-\delta)g(a) + \delta w \text{ and } \delta(w - P(W)) \geq (1-\delta)h(a)\big\}.$$

[1] The portion of the implementation devoted to analytical solutions uses Mathematica in addition. A zip file of the program is also available in a supplementary file on the journal website, http://econtheory.org/supp/1302/supplement.zip.

[2] The straightforward adaptation of the original APS framework to the simpler setting of perfect monitoring environments appears in Cronshaw and Luenberger (1994), and in teaching notes and problem sets of the original authors. See Mailath and Samuelson (2006) for an excellent general exposition of discrete-time repeated games.

APS also show that the operator B is monotonic, that is, for any two sets $W \subseteq W'$,

$$B(W) \subseteq B(W').$$

The APS algorithm starts with a (bounded) convex set W^0 that clearly contains V^*, and applies operator B infinitely many times. Defining recursively $W^n = B(W^{n-1})$, APS show that

$$W^n \to V^* \quad \text{as } n \to \infty.$$

If $B(W^0) \subseteq W^0$, then the monotonicity of operator B implies that $V^* \subseteq \cdots \subseteq W^n \subseteq W^{n-1} \subseteq \cdots \subseteq W^0$. One such possible starting point for the iterative algorithm is the convex hull of all feasible payoffs,

$$W^0 = \text{co}\{g(a) \mid a \in A\}.$$

3. A theoretical result

We present here a basic result on how extreme points are generated. An immediate implication is that the number of extreme points of V^* is finite, in fact, at most $4|A|$, where $|A|$ is the number of stage game action profiles. This result is of theoretical interest. It also motivates the algorithm that we develop below.

To understand our approach, one needs a detailed mechanical understanding of the operator B. It is useful to "break down" the operator B as follows: for an action profile a and a threat point $u \in \mathbb{R}^2$, denote by $Q(a, W, u)$ the intersection of the quadrant

$$\{w \in R^2 \mid \delta(w - u) \geq (1 - \delta)h(a)\}$$

with the set W. To compute $B(W)$, one takes the linear combination

$$B_a(W) = (1 - \delta)g(a) + \delta Q(a, W, P(W))$$

and finds the convex hull of the union of $B_a(W)$ over all action profiles a. This procedure, which yields $B(W)$, is illustrated in Figure 1.

Since each set $B_a(W)$ could have as many vertices as the set W itself (or more), the number of vertices that the sets of the sequence $\{W^n\}$ have could potentially grow without bound. Hence, the APS algorithm does not yield a bound on the number of extreme points of V^*. However, the following theorem does lead to such a bound, by specifying a limited number of ways in which extreme points of V^* can be generated.

Theorem 1. *Any action profile a such that $g(a) \geq \underline{v} + (1 - \delta)/\delta h(a)$ generates at most one extreme point of V^*, $v = g(a)$. Any action profile for which*

$$g_1(a) < \underline{v}_1 + \frac{1 - \delta}{\delta} h_1(a) \quad \text{or} \quad g_2(a) < \underline{v}_2 + \frac{1 - \delta}{\delta} h_2(a)$$

generates at most four extreme points of V^*, using continuation payoff vectors w that are extreme points of $Q(a, V^*, \underline{v})$ such that

$$\delta(w_1 - \underline{v}_1) = (1 - \delta)h_1(a) \quad \text{or} \quad \delta(w_2 - \underline{v}_2) = (1 - \delta)h_2(a).$$

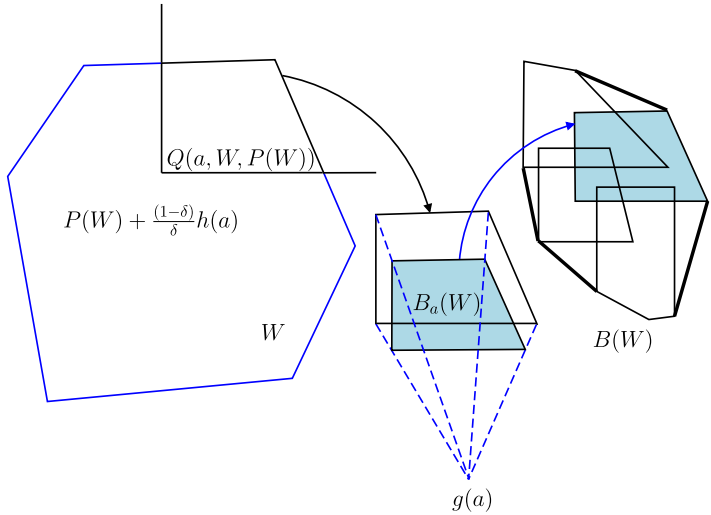

FIGURE 1. The construction of the operator $B(W)$.

The proof follows directly from a stronger Theorem 2 below.

Abreu (1986) has results with this flavor (see especially Lemma 44) in the special setting of continuous action repeated Cournot games.

In the finite setting of the current paper Theorem 1 has a significant theoretical implication of independent interest: the number of extreme points of V^* is at most $4|A|$, where $|A|$ is the number of stage game action profiles. In this setting, it was not even known (as far as we are aware) that the number of extreme points is finite. For instance, as noted above, the APS algorithm does not yield this conclusion.

In fact, this bound can be tightened to $3|A|$ extreme points; see Theorem 4 in the Appendix.

The reader may wonder whether a similar bound holds for games with three or more players. It is still true that each action profile a generates either a single payoff profile $g(a)$ or possibly several payoff profiles, each with a payoff of $\underline{v}_i + (1-\delta)/\delta h_i(a)$ for at least one player i who has a binding incentive constraint. However, for three or more players, this fact alone does not produce a bound on the number of extreme points of V^*.

4. Our algorithm

We develop an algorithm for computing the set V^* that is related to the APS algorithm, but happens to be more efficient and simpler to program. As noted earlier, the number of vertices of the sets of the APS sequence $\{B^n(W^0)\}$ could potentially grow without bound, making computation difficult. We propose a simpler computational procedure, which is faster and more parsimonious.

Motivated by the theoretical results above, we focus on points produced by the APS algorithm, which are generated with one of the constraints binding or attained by repeated play of a particular action profile. Specifically, our algorithm makes use of the operator $R(W, u)$ defined below.

DEFINITION 2. For an action profile a, a convex set W, and a punishment vector u, define $C(a,W,u)=\{g(a)\}$ if

$$\delta(g(a)-u)\geq(1-\delta)h(a),$$

and otherwise let $C(a,W,u)$ be the set of extreme points of $Q(a,W,u)$ such that

$$\delta(w_1-u_1)=(1-\delta)h_1(a)\quad\text{or}\quad\delta(w_2-u_2)=(1-\delta)h_2(a).$$

Let

$$R(W,u)=\operatorname{co}\bigcup_{a\in A}(1-\delta)g(a)+\delta C(a,W,u).$$

Figure 2 illustrates the relationship between the sets $Q(a,W,u)$ and $C(a,W,u)$. Note that $C(a,W,u)$ picks up at most *four* points of the set $Q(a,W,u)$.

Note that by Theorem 1, $V^*=R(V^*,\underline{v})$. Our algorithm is based on successive applications of the operator R. The decisive advantage of this operator is that for any $a\in A$, whereas B takes account of all extreme points of $Q(a,W,u)$, our operator R considers at most four extreme points. At each iteration, there are fewer computations and the set generated is smaller than under the operator B. This advantage is, of course, cumulative. Whereas the nth application of the APS operator B might yield a set with $|A|^{n+1}$ extreme points, our algorithm yields a set with at most $4|A|$ extreme points at every round.

Why does this approach work? There are many potential problems. The most obvious and primary one is that the operator R may discard too much. Of course, Theorem 1 offers some hope that this might not be the case.[3] Furthermore, what guarantees convergence? This is indeed potentially problematic because R does not necessarily generate a monotone sequence. The precise specification of our algorithm (in particular, the inductive definition of u^{n+1}) finesses this issue, although it does not guarantee monotonicity.

We suggest the following algorithm:

Step 1. Start with a convex set W^0 that contains the set V^* and a vector $u^0\in\mathbb{R}^2$ such that $u^0\leq P(V^*)$.

Step 2. Inductively compute $W^n=R(W^{n-1},u^{n-1})$ and denote

$$u^n=\max(u^{n-1},P(W^n)).$$

Then W^n contains some, but usually not all, points generated by W^{n-1}, that is, $W^n\subseteq B(W^{n-1})$. Moreover, the threats u^n defined inductively are potentially weaker than $P(W^n)$. Inductively, it follows that $W^n\subseteq B^n(W^0)$ (see Lemma 5). Since the sequence $\{B^n(W^0)\}$ converges to V^* as $n\to\infty$, it follows immediately that W^n also converges to V^* if we can show that $V^*\subseteq W^n$ for all n. In that case, the sequence $\{W^n\}$ would be *squeezed* between $\{B^n(W_0)\}$ and V^*, and the algorithm works.

[3] It is only suggestive because it is a property of V^* and not, in the form stated in Theorem 1, related to any sets that we might encounter in the process of *computing* V^*.

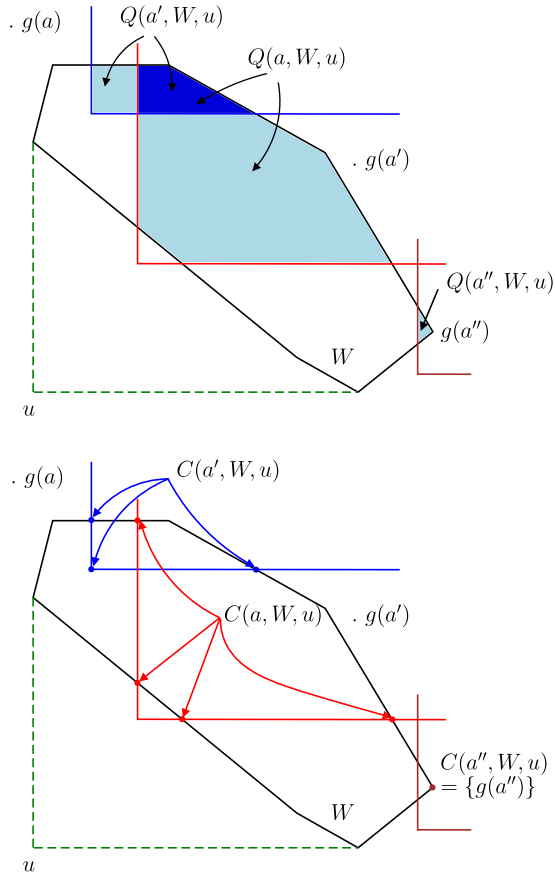

FIGURE 2. The relationship between the sets $Q(a, W, u)$ and $C(a, W, u)$.

Of course, given our analysis so far, there is no guarantee yet that W^n contains V^* for all n. All we have so far is crude intuition, motivated by Theorem 1, that the operator $R(W^{n-1}, u^{n-1})$ keeps only essential points and discards points that would be eventually eliminated by the APS algorithm. How do we establish that each element W^n of the sequence produced by the algorithm contains V^*?

It turns out that while the operator $R(W, u)$ is monotonic in W, it is not monotonic in u.[4] Therefore, inductively $V^* \subseteq W^{n-1}$ and $\underline{v} \geq u^{n-1}$ do *not* imply that $V^* = R(V^*, \underline{v}) \subseteq R(W^{n-1}, u^{n-1})$. To make the inductive argument work, as an intermediate step, we define sets $V(u)$ that contain V^* whenever $u \leq \underline{v}$, and prove by induction on n that

$$V(u^n) \subseteq W^n \quad \text{and} \quad u^n \geq \underline{v}$$

for all n. We proceed now to the details.

[4]The operator $R(W, u)$ does not have to be monotonic in u because co $C(a, W, u)$ is not monotonic in u. For severe threats u, $C(a, W, u)$ consists of a single point $\{g(a)\}$. As the threat weakens, $C(a, W, u)$ consists of multiple points. Even after that, co $C(a, W, u)$ is nonmonotonic because it does not contain the entire set $Q(a, W, u)$. Any point of $Q(a, W, u)$ becomes an element of $C(a, W, u')$ for some $u' \geq u$.

DEFINITION 3. Denote by $B(W,u)$ the set of points generated by a set $W \subseteq \mathbb{R}^2$ using the threats given by $u \in \mathbb{R}^2$, that is,

$$B(W,u) = \operatorname{co} \bigcup_{a \in A} (1-\delta)g(a) + \delta Q(a,W,u).$$

Let $V(u)$ denote the largest bounded self-generating set under $B(\cdot,u)$ (i.e., such that $V(u) \subseteq B(V(u),u)$).

Since $B(\cdot,u)$ is monotonic in its first argument, it follows directly from definitions that the union of all self-generating sets is self-generating and that $V(u) = B(V(u),u)$. Since stronger threats relax the incentive constraints, we have the following lemma, which implies that $V^* \subseteq V(u)$ whenever $u \leq \underline{v}$.

LEMMA 1. *If $u \leq u'$, then $V(u') \subseteq V(u)$.*

The proof follows directly from definitions.

Moreover, it turns out that any extreme point of the set $V(u)$ is generated by some action profile a with continuation payoff vector from $C(a,W,u)$, as shown in Theorem 2 below.

THEOREM 2. *We have $V(u) = R(V(u),u)$.*

PROOF. We need to show that if $v \in V(u)$ is an extreme point generated with the action profile $a \in A$ and continuation value $w \in V(u)$, then $w \in C(a,V(u),u)$. First, if $g(a) \geq u + (1-\delta)/\delta h(a)$, then $g(a) \in V(u)$. Since $v = (1-\delta)g(a) + \delta w$, it follows that $v = w = g(a)$ or else v cannot be an extreme point of $V(u)$ (since w is also in $V(u)$).

Otherwise, $g_1(a) < u_1 + (1-\delta)/\delta h_1(a)$ or $g_2(a) < u_2 + (1-\delta)/\delta h_2(a)$ and so $v \neq w$ (see Figure 3). Since v is extreme, w must be an extreme point of $Q(a,V(u),u)$. Note that also $[w,v] \equiv \{w' \mid w' = \lambda w + (1-\lambda)v, \lambda \in [0,1]\} \subseteq V(u)$. Now, if the incentive constraints are slack, that is, $w > u + (1-\delta)/\delta h(a)$, then $(w,v] \cap Q(a,V(u),u) \neq \varnothing$. Consider $w' \in (w,v] \cap Q(a,V(u),u)$ and let $z = (1-\delta)g(a) + \delta w' \in V(u)$. Then $v = \lambda w + (1-\lambda)z$ for some $\lambda \in (0,1)$ and so v is not extreme, a contradiction. It follows that w is an extreme point of $Q(a,V(u),u)$ such that $w_i = u_i + (1-\delta)/\delta h_i(a)$ for $i = 1$ or 2, that is, one of the two incentive constraints is binding. Therefore, $w \in C(a,V(u),u)$ in this case as well. □

It is also useful to note that R is increasing in its first argument.

LEMMA 2. *The function $R(\cdot,u)$ is increasing in its first argument.*

PROOF. Note that for all a, $\operatorname{co} C(a,W,u)$ is increasing in W. Therefore,

$$\operatorname{co}\{(1-\delta)g(a) + \delta C(a,W,u)\}$$

is increasing in W and $R(W,u)$ is increasing in W by the definition of $R(W,u)$. □

The following lemma presents the main inductive argument.

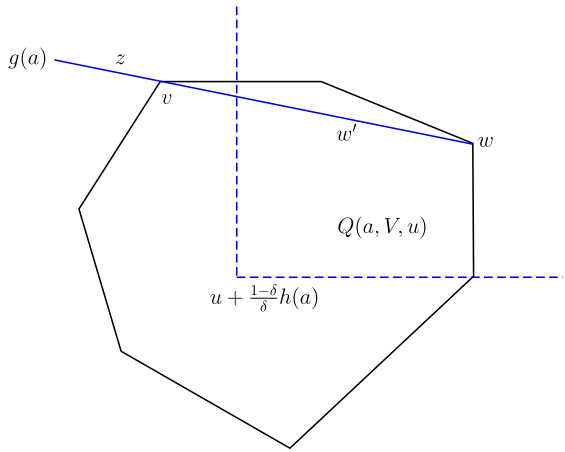

FIGURE 3. The proof of Theorem 2: the case of $g_1(a) < u_1 + ((1-\delta)/\delta)h_1(a)$.

LEMMA 3. *If $V^* \subseteq V(u^{n-1}) \subseteq W^{n-1}$ and $u^{n-1} \leq \underline{v}$, then $u^n \leq \underline{v}$ and $V^* \subseteq V(u^n) \subseteq W^n$.*

PROOF. By definition, $u^n \geq u^{n-1}$. It follows from Lemma 1 that $V(u^n) \subseteq V(u^{n-1})$.

Since $V(u^{n-1}) \subseteq W^{n-1}$, it follows from Lemma 2 that

$$R(V(u^{n-1}), u^{n-1}) \subseteq R(W^{n-1}, u^{n-1}).$$

The left hand side is $V(u^{n-1})$ (contains $V(u^n)$) by Theorem 2. The right hand side is W^n by definition. Therefore, $V(u^n) \subseteq W^n$.

Since $V^* \subseteq V(u^{n-1}) \subseteq W^n$ and $u^{n-1} \leq \underline{v}$, we have $u^n = \max(u^{n-1}, P(W^n)) \leq \underline{v}$. Hence, Lemma 1 implies that $V^* \subseteq V(u^n)$ (which is contained in W^n by the previous paragraph). □

Letting u^0 be the vector of pure-strategy minmax payoffs of players 1 and 2, we have $u^0 \leq \underline{v}$ and $V^* \subseteq V(u^0) \subseteq W^0$, and so Lemma 3 implies that $V^* \subseteq W^n$ for all n, by induction on n.

Hence, the sequence W^n is squeezed between $V(u^n)$, which contains V^*, and $B^n(W^0)$, which converges to V^* by APS. It follows that W^n must converge to V^* also, a conclusion that we summarize in Theorem 3.

THEOREM 3. *We have $W^n \to V^*$ as $n \to \infty$.*

PROOF. Lemma 3 implies that $V^* \subseteq V(u^n) \subseteq W^n$. Also, by induction on n, $W^n \subseteq B(W^{n-1}, u^{n-1}) \subseteq B(W^{n-1}) \subseteq B^n(W^0)$ (see Lemma 5 in the Appendix). This leads to the desired conclusion. □

That is, our algorithm does indeed yield convergence to V^* (despite possible lack of monotonicity and the fact that it drops points that the APS algorithm includes).[5]

[5]In particular, if the set V^* is empty, then $W^n = R(W^{n-1}, u^{n-1})$ becomes empty at some *finite* stage n; a decreasing sequence of closed nonempty sets cannot converge to an empty set. Note that a necessary condition for V^* to be empty is that the stage game have no pure-strategy Nash equilibria.

5. Implementation, examples, and evaluation

The set V^* is naturally described by its extreme points: any extreme point of V^* is generated by some action profile a today and some incentive compatible continuation value $w \in Q^*(a)$ tomorrow, where $Q^*(a)$ denotes the set of incentive compatible continuation values. According to Theorem 1, if $g(a) \in Q^*(a)$, then $w = g(a)$. If not, w must be an extreme point of $Q^*(a)$ at which one or both players' incentive constraints are binding. There are at most four such points. Crucially, we were able to show that an algorithm that mimics these properties of the *equilibrium set* V^* does, in fact, work and yields convergence to V^*. This yields an important simplification that goes beyond the natural focus on extreme points of $Q^*(a)$ that is a noteworthy feature of the APS algorithm.

In this section, we start by presenting a simple illustrative example. We then turn to a comparison of our algorithm with that of Judd et al. (2003) (JYC). To perform a practical comparison of running times, we use a Cournot duopoly example from JYC and calculate the equilibrium payoff set with the two algorithms. To perform theoretical comparisons, we estimate the number of steps that each algorithm takes to run for a game with a given number of actions per player and for a given level of computational precision.

Implementation. Because our algorithm exploits our knowledge about the structure of equilibria, the algorithm also produces output in a form that makes clear not only the shape of V^*, but also of the equilibria themselves. As noted in the Introduction, a standalone implementation of our algorithm is available in the supplementary material. Here the user may input payoffs for *any* finite two-player game and *any* discount factor, and immediately see the solution, the coordinates of the extreme points, how they are generated, solve for exact solutions (see Section 5), observe, if desired, the iterative steps of the algorithm, and so on. The reader is invited to explore the algorithm here and solve their games of choice.

Simple example. We turn now to a simple example that illustrates the algorithm and its implementation "in action." The computation of V^* and the decomposition of extreme points and associated diagrams are all taken directly from the implementation discussed above. Consider a Cournot duopoly game with the payoff matrix

	L	M	H
L	$16, 9$	$3, 13$	$0, 3$
M	$21, 1$	$10, 4$	$-1, 0$
H	$9, 0$	$5, -4$	$-5, -15$

In this game, the Nash equilibrium is (M, M), with payoffs $(10, 4)$. At the same time, profiles (L, L), (L, M), and (M, L) all produce a higher sum of payoffs. The minmax payoffs are 1 for player 1 and 0 for player 2.

Figure 4a shows the set of feasible payoffs and the hexagonal set V^* for the discount factor $\delta = 0.3$. It also illustrates how the extreme point $v \in V^*$, which maximizes the payoff of player 2, is generated using the profile (L, M) with payoffs $(3, 13)$. Because player 1

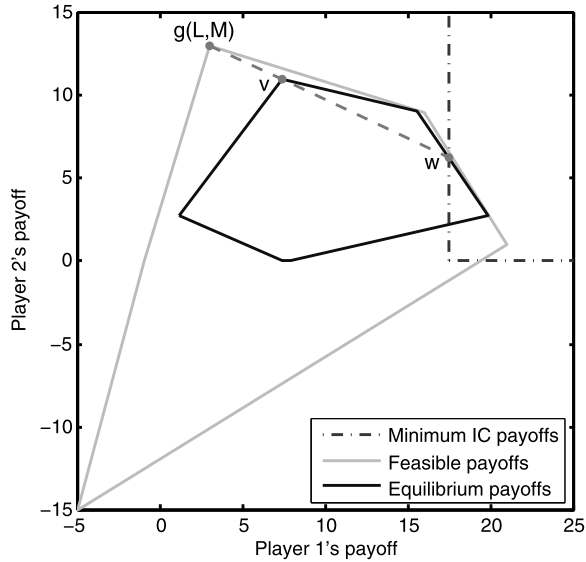

FIGURE 4a. The set V^* in the three-by-three Cournot example, $\delta = 0.3$.

has a large deviation payoff, this action profile is enforced using continuation value w, with a large payoff to player 1. The dashed quadrant in the figure tells us that player 1's constraint is binding.

The four small panels of Figure 4b illustrate the generation of *five* other extreme points, highlighting the current-period action profile, the continuation value in the next period, and the binding constraint(s). Interestingly, play on the Pareto frontier involves complex dynamics, in which all three action profiles (L, L), (L, M), and (M, L) necessarily have to be used on the equilibrium path. For example, the play of (L, L) requires a continuation value on the line segment, which is generated using public randomization between profiles (L, M) and (L, L) (see top right small panel). If the randomization points to (L, M), the continuation value in the following period requires public randomization between profiles (L, L) and (M, L) (see Figure 4a). A play of (M, L) in the following period would be followed by randomization between (M, L) and (L, L) (see top left small panel of Figure 4b). Thus, the information provided in the panels allows us to simulate the evolution of equilibrium play.

The punishment extreme points of V^* are used only after deviations off the equilibrium path. Note that both punishment points of player 2 are generated by the profile (H, M) with player 2's constraint binding. Thus, player 2 is indifferent between taking the equilibrium action or deviating to the best response L. If he does the latter, he receives his minmax payoff of 0 and receives punishment in the next period. From this, we know that the worst equilibrium punishment of player 2 is exactly 0.

In contrast, player 1's punishment point is generated by the profile (M, H) with *player 2's* constraint binding. Player 1 strictly prefers to participate in his own

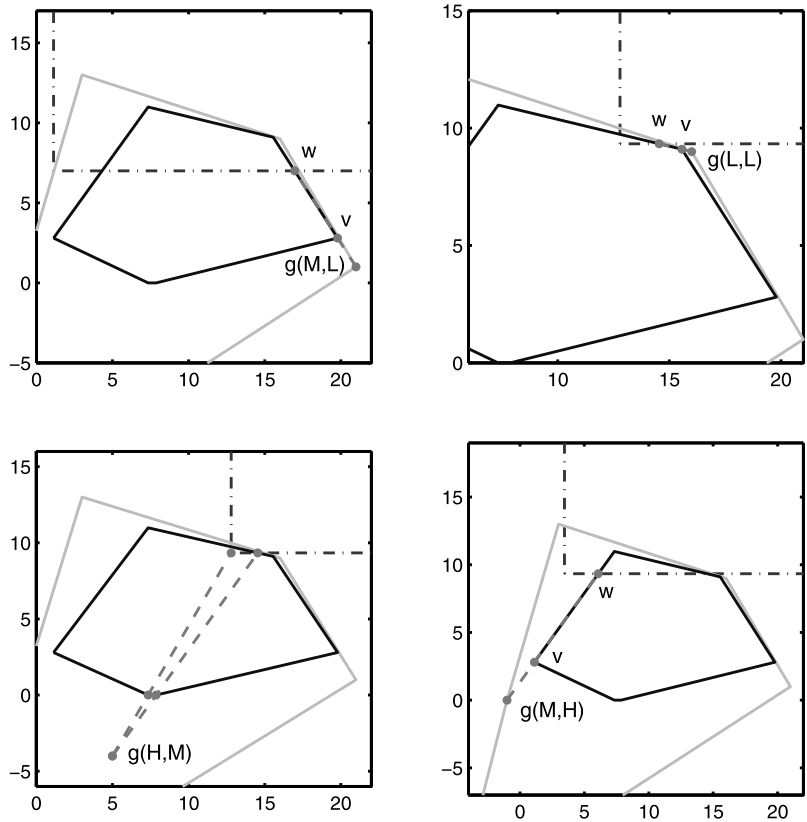

FIGURE 4b. The set V^* in the three-by-three Cournot example, $\delta = 0.3$.

punishment. Therefore, his worst punishment is strictly greater than 1, which is his best response payoff to action H of player 2, and also his minmax payoff.

In Section 6.1, we discuss how the knowledge of how each extreme point of V^* is generated translates into an exact system of equations for coordinates of these points. These equations can be often solved analytically.

This game takes 23 iterations (and 0.11 seconds) to solve.

THE JYC ALGORITHM AND THEORETICAL COMPARISON. The JYC algorithm, in principle, can be applied to games with more than two players. Below we describe the JYC algorithm in the context of two-player games.

The JYC algorithm is based on linear programming. Rather than representing sets via its extreme points, it represents sets by *outer approximations* via inequalities. A convex set W is approximated by

$$\{x \in \Re^2 \mid x \cdot h_l \leq c_l \text{ for all } l = 1, \ldots, L\},$$

where $h_l = (\cos(2\pi l/L), \sin(2\pi l/L))$, L is the number of approximating subgradients, and $c_l = \max_{w \in W} w \cdot h_l$. The JYC algorithm is based on linear programming

problems

$$\begin{aligned} c_l'(a) &= \max_w h_l \cdot [(1-\delta)g(a) + \delta w] \\ \text{s.t.} \quad &\text{(i) } w \cdot h_l \le c_l \quad \text{for all } l = 1, \ldots, L \\ &\text{(ii) } (1-\delta)g_i(a) + \delta w_i \ge (1-\delta)\overline{g}_i(a_j) + \delta P_i(W), \end{aligned} \tag{1}$$

where the number $P_i(W)$ itself can be found using linear programming. To find $B(W)$, JYC solve (1) for each subgradient, for each action profile $a \in A$. Then they let $c_l' = \max_{a \in A} c_l(a)$ and approximate $B(W)$ by

$$\{x \in \Re^2 \mid x \cdot h_l \le c_l' \text{ for all } l = 1, \ldots, L\}.$$

If a game has M action profiles, then the computational complexity of the JYC algorithm per iteration can be estimated as follows. Although the two-dimensional linear programming problem (1) would run quite fast, it takes at least $O(L)$ steps to solve, where L is the number of approximating subgradients. Since it has to be solved LM times during each iteration, the running time of the algorithm is at least $O(ML^2)$ per iteration.

In contrast, to evaluate the complexity of our algorithm, note that after each iteration, the set W^{n-1} has at most $4M$ extreme points. In our implementation, we order the points in such a way that the search for extreme points of each set $Q(a, W^{n-1}, u^{n-1})$ takes $O(\log(M))$ steps. Since we need to do this for every profile a, the running time of the algorithm is $O(M \log M)$. The JYC algorithm could potentially be faster on games with a huge number of actions when using relatively few approximating subgradients, that is, $M \gg L$. However, approximations via few subgradients could easily lead the JYC algorithm to converge to a much larger set than V^* (although JYC also propose an inner approximation method to deal with this issue).[6] If a large number of actions is chosen to approximate a continuous action space, it makes sense to choose L to achieve a similar precision of approximation for the shape of the boundary of V^*. At the minimum, L should be of $O(M^{1/2})$, that is, the number of actions per player. In this case, the running time of the JYC algorithm of $O(M^2)$ is slower than the running time of our algorithm of $O(M \log M)$. The difference can be a lot more significant when the number of subgradients is motivated by the precision of approximation of V^*, but the number of actions is finite. As we discuss at the end of this section, the error of approximating a set of radius R with L subgradients is about $R\pi^2/(2L^2)$. For many examples, this motivates values of L much larger than M.

In addition, unlike our algorithm, the JYC algorithm does not provide direct information about the paths of equilibrium play. For each subgradient direction, the JYC algorithm provides a point that maximizes payoffs in that direction, as well as continuation values used in the following period. However, after that, it is difficult to express the continuation values as linear combinations of points that can be generated, as the algorithm characterizes the set via subgradients and not extreme points. In contrast, once our algorithm has converged, each extreme point is generated as a linear combination

[6] See our discussion of *inner approximations* in Section 6.2.

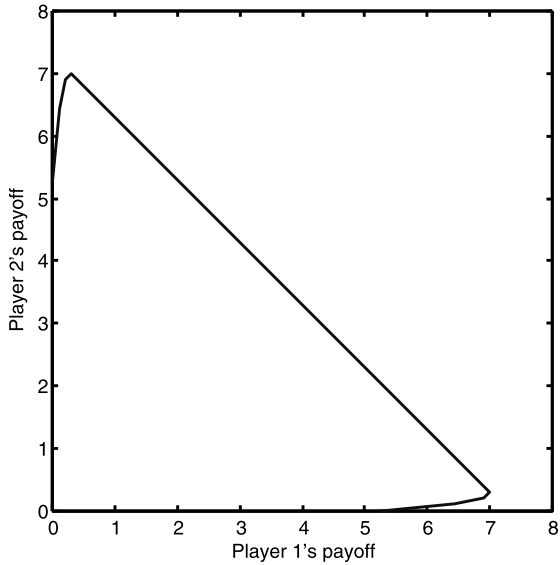

FIGURE 5. The set V^* in the Cournot duopoly game from JYC.

of at most two other extreme points. This provides direct information about the paths of play and serves as a basis for a procedure to get exact solutions that we describe in Section 6.1.

EXPERIMENTAL COMPARISON WITH THE JYC ALGORITHM. We borrow a game from JYC: a Cournot duopoly in which firms face the demand function $P = 6 - q_1 - q_2$, and receive payoffs $q_1(P - 0.6)$ and $q_2(P - 0.6)$ (that is, $c_1 = c_2 = 0.6$). The set of actions of each firm is the interval $[0, 6]$, discretized to include 15 evenly spaced points. That is, altogether the game has $15^2 = 225$ action profiles, which could result in a nontrivial computation time. The discount factor is set at $\delta = 0.8$.

Figure 5 presents the set V^*. The algorithm with an error threshold of 0.0001 (in Hausdorff metric) takes eight iterations and a total run time of 0.34 seconds.

We also performed the same computation using a Fortran implementation of the JYC algorithm, kindly provided to us by Sevin Yeltekin. These experimental results complement the theoretical analysis above. Certainly, comparisons of implementations in different programming languages are quite crude, because differences in running times could be driven by particularities of the language or the implementation, rather than fundamentals of the algorithm. So reported results should be taken with the understanding that there may be many reasons for the differences. In our experiment, the JYC algorithm with the same error threshold of 0.0001 converged after 48 iterations in 6 minutes 51 seconds.[7] The average time per iteration of the JYC algorithm was 8.56 seconds versus 0.043 seconds for our algorithm. The iterations of the JYC algorithm are not equivalent to those of our algorithm, since the JYC iteration produces somewhat larger

[7] Following the JYC implementation, we used 72 subgradients to represent sets in the JYC algorithm. See the description of the algorithm in this section.

sets than the APS algorithm, while an iteration of our algorithm produces somewhat smaller sets. That is why our algorithm converges in fewer steps than the JYC algorithm in this example. As it happens, our limit set is *strictly* contained in the JYC limit set (although this would not be obvious on visual inspection!). The following table records our observations of running times:

		JYC	AS
Error 10^{-4}	No. of iterations	48	8
	Run time	6 min 51 s	0.34 s
Error 2^{-52}	No. of iterations	260	26
	Run time	32 min 16 s	1.09 s

In their 2003 paper JYC report a run time of 44 minutes 53 seconds for this example with $L = 72$ approximating subgradients and faster running times of 4 minutes 46 seconds and 63 seconds for $L = 32$ and 16, respectively, using an error threshold of 10^{-5}. We can estimate the error of approximating a circle of radius R with L subgradients to be about $R\pi^2/(2L^2)$. This suggests that the number of approximating subgradients has to be tailored to the desired precision of calculation. To get an error margin of about 10^{-5} for the set of radius 3 (see Figure 5), it seems appropriate to use at least 1,000 approximating subgradients.[8]

6. Further topics

6.1 *From almost convergence to exact solutions*

This section combines computational methods with analytical ones in a novel way. This hybrid approach may well be more widely applicable and is an important component of the current paper.

Once the algorithm "stops," the set of extreme points remains fixed and we can express how each extreme point is generated as a linear combination of at most two other extreme points. These relationships provide direct information about the paths of equilibrium play. They also lead to a simultaneous equation system that, in principle, can be solved exactly. This is particularly simple when the system is linear, as it will frequently turn out to be.

Suppose the algorithm has almost converged and we, therefore, know (generically) how many extreme points there are and precisely how they are generated. Let the extreme points be ordered in some convenient way, say clockwise, $E^1, E^2, \ldots, E^M$. Let $m(i)$ index the extreme point that yields the lowest payoff to player i. There is an action profile a^m associated with every extreme point E^m. From Theorems 1 and 3, we know that there are four possibilities:

(i) Neither player's constraint binds and $E^m = g(a^m)$, where g denotes the stage game payoff function.

[8]JYC indicate that they ran their program on a 550 MHz Pentium PC. We ran their program on a 3.4 GHz machine.

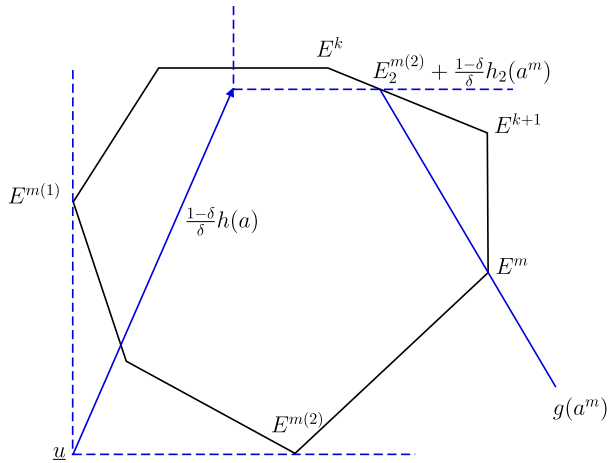

FIGURE 6. The case of player 2's binding constraint.

(ii) Player 2's constraint binds. This case is depicted in Figure 6.

Recall that $h_i(a^m)$ is the (maximal) *gain* to player i of deviating from a_i^m given that his opponent plays a_j^m in the stage game. If player 2's constraint binds, then player 2's continuation payoff is $w_2 = E_2^{m(2)} + (1-\delta)/\delta h_2(a^m)$. Hence,

$$E_2^m = (1-\delta)g_2(a^m) + \delta \underbrace{\left(E_2^{m(2)} + \frac{1-\delta}{\delta} h_2(a^m)\right)}_{w_2} = (1-\delta)\overline{g}_2(a^m) + \delta E_2^{m(2)}.$$

The continuation payoff w_2 is obtained by public randomization between the extreme points E_2^k and E_2^{k+1} with weights α^m and $(1-\alpha^m)$, respectively.

We may solve for α^m, where

$$\alpha^m E_2^k + (1-\alpha^m)E_2^{k+1} = E_2^{m(2)} + \frac{1-\delta}{\delta} h_2(a^m) \quad (= w_2).$$

Furthermore,

$$E_1^m = (1-\delta)g_1(a^m) + \delta[\alpha^m E_1^k + (1-\alpha^m)E_1^{k+1}].$$

(iii) Player 1's constraint binds. This case is analogous to (ii) above with the indices 1 and 2 reversed.

(iv) Both constraints bind. Then

$$\begin{aligned} E^m &= (1-\delta)g(a^m) + \delta\left[\underline{u} + \frac{1-\delta}{\delta} h(a^m)\right] \\ &= (1-\delta)\overline{g}(a^m) + \delta\underline{u}, \quad \text{where } \underline{u} \equiv (E_1^{m(1)}, E_2^{m(2)}). \end{aligned}$$

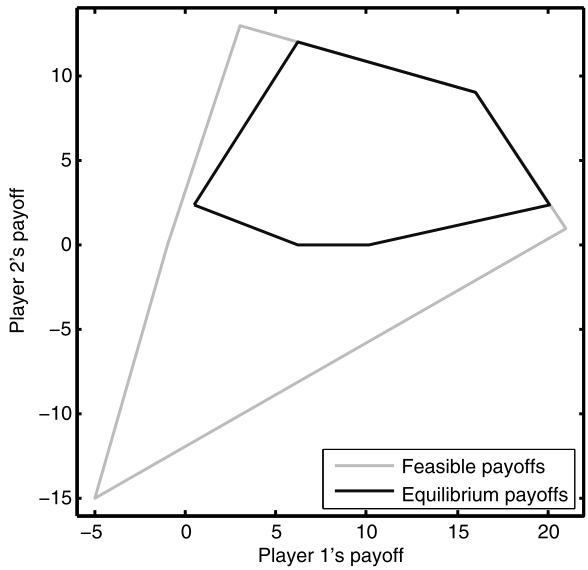

FIGURE 7. The set V^* in the three-by-three Cournot example of Section 5, for $\delta = 0.4$.

This yields a system of simultaneous equations in the unknown E_i^m's and (in the relevant cases) corresponding α^m's. The equations that correspond to cases (i) and (iv) are trivial. Some of the type (ii) and (iii) equations will also be simple (in particular, linear) when E_i^k and E_i^{k+1} correspond to cases (i) or (iv) above. So linearity might be contagious. In any case, this system of equations does not seem terribly complicated and in many instances should solve quite easily. This would then yield a kind of "holy grail" (at least locally): a closed form solution for the set of discounted equilibrium payoffs in a repeated game.

Such exact solutions are of great interest because they are expressed symbolically in closed form (in our current implementation, we simply use Mathematica) and may be used to perform comparative statics with respect to changes in payoffs (driven by changes in underlying parameters of cost or demand, for instance) in a neighborhood of the original payoffs. The ability to do this in applied problems is potentially quite exciting.

We demonstrate this approach with the example introduced earlier, which we now evaluate with a discount factor $\delta = 0.4$ (which also serves to illustrate the evolution of the set as the discount factor increases). The algorithm yields the limit set in Figure 7.

The coordinates of the respective extreme points are reported as

$$(0.43169727312472, 2.4), \quad (6.17267888662042, 0),$$
$$(10.15, 0), \quad (20.125, 2.4), \quad (16, 9),$$
$$(6.17267888662042, 12.0237911118091).$$

The equation system that corresponds to the above limit set is also reported as a final step in our algorithm and we solve this system using Mathematica. Mathematica

reports the exact solution for the extreme points,

$$\left(\frac{3{,}573-\sqrt{12{,}520{,}729}}{80},\frac{12}{5}\right),\qquad \left(\frac{4{,}773-\sqrt{12{,}520{,}729}}{200},0\right),$$
$$\left(\frac{203}{20},0\right),\qquad \left(\frac{161}{8},\frac{12}{5}\right),\qquad (16,9),$$
$$\left(\frac{4{,}773-\sqrt{12{,}520{,}729}}{200},\frac{4{,}277+\sqrt{12{,}520{,}729}}{650}\right),$$

which agrees with the earlier algorithmic solution up to the 12th decimal place.

Now suppose we replace the payoff pair that corresponds to, say, (M,H) by $(-1+x,0+y)$. The exact solutions obtained by Mathematica are

$$\left(\frac{3{,}613+A(x,y)}{80},\frac{12}{5}\right),\qquad \left(\frac{4{,}773+A(x,y)}{200},0\right),$$
$$\left(\frac{203}{20},0\right),\qquad \left(\frac{161}{8},\frac{12}{5}\right),\qquad (16,9),$$
$$\left(\frac{4{,}773+A(x,y)}{200},\frac{4{,}277-A(x,y)}{650}\right),$$

where

$$A(x,y)=40x-195y-\sqrt{12{,}520{,}729-217{,}360x+1{,}600x^2-1{,}081{,}470y-15{,}600xy+38{,}025y^2},$$

and we have explicit expressions for how the extreme points vary with x and y.

6.2 *Inner approximation*

Very low error thresholds give one substantial confidence that the true equilibrium value set V^* has been basically attained and it is not obvious to us, as a practical matter, that one needs further confirmation that V^* has been, for all practical purposes, determined. Nevertheless there is the theoretical possibility of a discontinuity. In response to this concern JYC suggest performing "inner approximation" to complement their "outer approximation" of the set.[9]

In this section, we (i) describe JYC's inner approximation procedure, (ii) discuss potential problems with the procedure, and (iii) to address the problems, propose a modification of the JYC procedure based on a new theoretical result (Lemma 4 below). Of course, the inner approximation procedure is unnecessary in cases where we obtain *exact* solutions along the lines outlined in the preceding subsection; in those cases, we are decisively done.

The procedure that JYC suggest works as follows. After running their implementation of the APS algorithm and almost converging, they "shrink" their final set W^n by

[9]Technically, the set produced by our main algorithm is an outer approximation of V^*, that is, it contains V^*.

a small amount (such as 2–3%) to obtain a new set $\underline{W}_0$. The first step in their procedure entails checking if $\underline{W}_0 \subseteq B(\underline{W}_0)$. If it is, they proceed to iteratively compute $\underline{W}_n$. However, unlike in the original iteration, which errs on the side of a larger set by approximating via supporting hyperplanes, here they use an alternative implementation B^I that errs on the side of a smaller set. In particular, given a fixed set of search subgradients $\{h_l, l = 1, \ldots, L\}$, $B^I(W)$ is the convex hull of extreme points of $B(W)$ that solve $\max_{x \in R(W)}\{x \cdot h_l \mid x \in B(W)\}$.

The operator B^I is also used initially to test whether $\underline{W}_0$ is self-generating. If this test is passed, it follows directly that $\underline{W}_n = B^I(\underline{W}_{n-1})$ is an *increasing* sequence. Convergence is registered when the distance between $\underline{W}_n$ and $\underline{W}_{n-1}$ is small, and $\underline{W}_n$ is taken to be a *lower bound* on V^*.

If $\underline{W}_0 \subseteq B^I(\underline{W}_0)$, then, in a sense, JYC are already done; $\underline{W}_0$ is contained in V^* and is, moreover, within (say) 2% of it. The additional steps are less essential and serve to whittle away at that initial slight shrinkage.

However, a potential difficulty with the starting point of the JYC procedure is that the initial set $\underline{W}_0$ may *not* be self-generating. Consider, for example, a game with the payoff matrix

	1	2	3
1	400, 530	0, −400	1, 1
2	1,100, −1,200	0, 0	−400, 0
3	1, 1	−1,200, 1,100	530, 400

For the discount factor $\delta = 0.6$, the set V^* for this game is presented in Figure 8. The extreme points of V^* are

$$(0, 0), \qquad (490, 440), \quad \text{and} \quad (440, 490).$$

Unless the search directions chosen in the JYC algorithm accidentally coincide with the faces of V^*, then the outer approximation W^n obtained by JYC will have the form presented in Figure 9.

Depending on the grid of possible search directions and their relation to the true V^*, a "slightly" shrunken version[10] of W^n may or may not be contained in V^*. If not, such a $\underline{W}_0$ will obviously not self-generate.[11]

[10] We "shrink" a convex set C with a finite number of extreme points as follows. Let $\overline{e}$ denote the arithmetic average of the set of extreme points. Then for every extreme point e of C, we define a new extreme point $e' = \alpha\overline{e} + (1 - \alpha)e$, where α is the shrinkage factor. Our shrunken set C' is the convex hull of the e's. In the example, we use $\alpha = 0.02$.

[11] In their footnote 11, JYC (p. 1246) suggest that in such a situation, one might define $\underline{W}_0$ to be the set of payoffs obtainable using simple subgame-perfect strategies such as those that prescribe constant action profiles and Nash reversion in the event of deviation. In the example above, the only such equilibrium strategy is playing the Nash equilibrium always and the inner approximation this yields will be $\underline{W}_0 = \{(0, 0)\}$ itself.

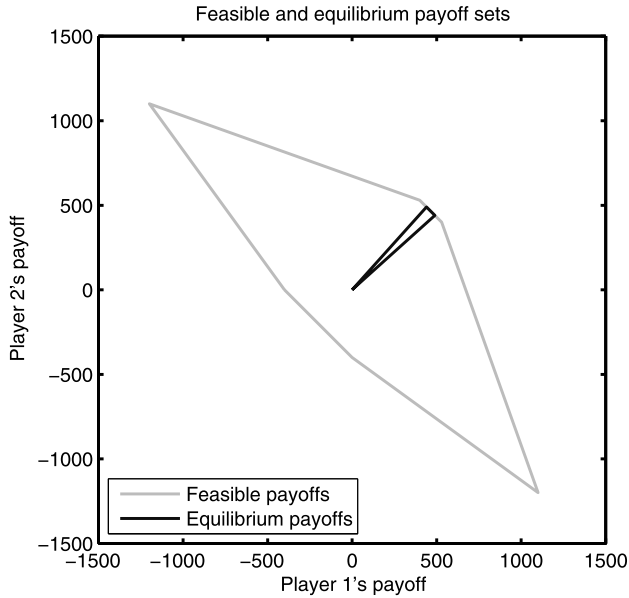

FIGURE 8. A *narrow* set V^* for $\delta = 0.6$.

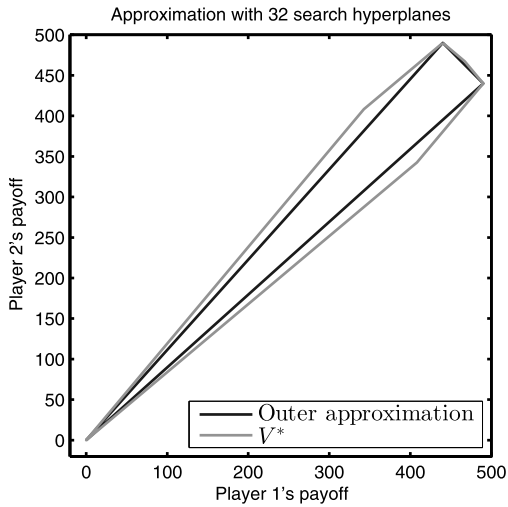

FIGURE 9. Outer approximation with 32 search hyperplanes.

REMARK. The difficulty here is that there are spurious extreme points that do not get shrunken away. An easy fix to this problem is to specify $\underline{W}_0$ by shrinking $B^I(W^{n-1})$ rather than the larger set $B^O(W^{n-1}) \equiv W^n$.

However, even when we modify the JYC procedure as above, it is not the case that "generically" $\underline{W}_0 \nsubseteq B(\underline{W}_0)$. There is a more subtle difficulty also illustrated by the same example.

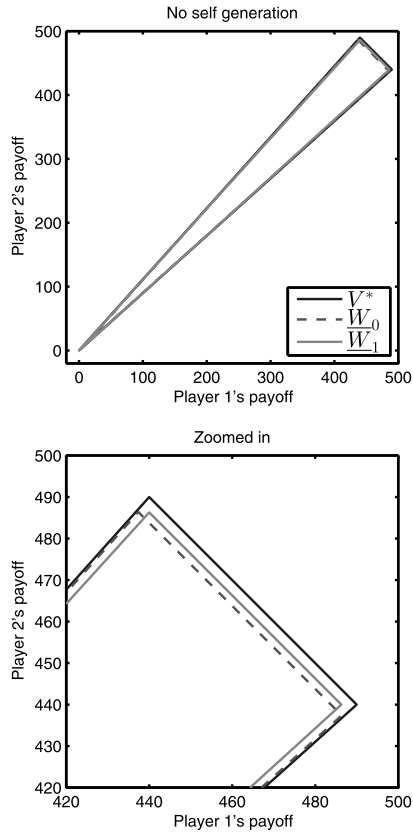

FIGURE 10. An example in which the set $\underline{W}_0$ does not supergenerate.

Let us shrink V^* by 2% to obtain $\underline{W}_0$. Then as shown in Figure 10, $\underline{W}_0 \nsubseteq B(\underline{W}_0)$. Hence, $\underline{W}_0 \nsubseteq B^I(\underline{W}_0)$. Note that there is nothing nongeneric about this example. Slight perturbations of payoffs or the discount factor do not affect the conclusion.

The example demonstrates that there is no presumption that a slightly smaller set than V^* will necessarily expand in *all* directions in one round of application of the B (or related) operator(s). Intuitively, the difficulty is that there are many interdependent extreme points and the initial movement in some may be large relative to others. One cannot be assured of *initial* containment followed by a monotonic process of expansion. Unfortunately, JYC inner approximation only "launches" if $W_0 \subseteq B(W_0)$.[12] We suggest an alternative procedure that is also based on slight initial shrinkage but bypasses the (possibly overly stringent) JYC test. Our starting point is the reasonable expectation that successive applications of the operator within a "basin of attraction" of V^* will get close to V^* and, therefore, *eventually* contain the initial slightly shrunken set. But how does

[12]As noted earlier, if JYC launches from a slightly shrunken set, the procedure is somewhat redundant; "slight" shrinkage is reduced to "slighter" shrinkage. On the other hand, if W_0 is much smaller than V^*, then there is the danger that inner approximation converges to a fixed point of B that is distinct from (and smaller than) V^*. Footnote 11 provides an extreme example.

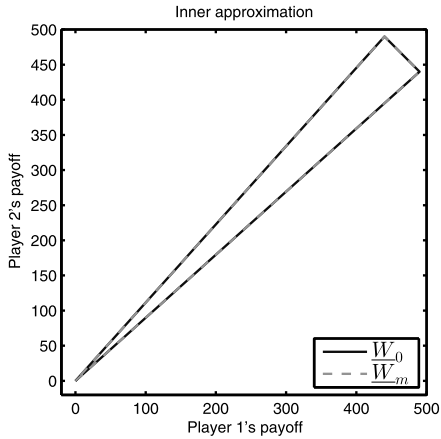

FIGURE 11. Sets $\underline{W}_0$ and $\underline{W}_m$ in our inner approximation procedure.

one convert this eventual containment into a sufficient condition for a legitimate inner approximation?

The following new lemma provides an answer; it generalizes the self-generation result of APS.

LEMMA 4. *Suppose $X \subseteq \mathbb{R}^2$ is compact and let $Y \equiv \bigcup_{k=1}^{m} B^k(X)$. If $X \subseteq Y$, then*

$$Y \subseteq B(Y) \subseteq V^*.$$

PROOF. We first argue that $Y \subseteq B(Y)$. That $B(Y) \subseteq V^*$ then follows directly from self-generation. Suppose $y \in B^k(X)$, $k = 2, \ldots, m$. Then (by definition) $y \in B(B^{k-1}(X)) \subseteq B(Y)$ (since B is a monotone operator). Finally, if $y \in B^1(X)$, then $y \in B(Y)$ since $X \subseteq Y$ by assumption. □

Motivated by Lemma 4 and the preceding discussion, we propose the following inner approximation procedure (where the sets W^n now are taken to denote the sets obtained iteratively by *our* algorithm).

Step 1. Shrink W^n slightly to obtain $\underline{W}_0$.

Step 2. Apply our operator R to $\underline{W}_0$ to obtain $\underline{W}_1$ and proceed to define $\underline{W}_2, \underline{W}_3, \ldots$ inductively until convergence at $\underline{W}_m$.

Step 3. Check if $\underline{W}_0 \subseteq \underline{W}_m$. If so, $\bigcup_{k=1}^{m} \underline{W}_k \subseteq V^*$. In particular, $\underline{W}_m$ is an inner approximation of V^*.

Applying this procedure to our example, we find that we obtain convergence to within 10^{-6} in 0.0847 seconds and in 31 iterations.

Furthermore, $\underline{W}_0 \subseteq \underline{W}_m$ as required. As can be seen in Figure 11, $\underline{W}_m$ is visually indistinguishable from $\underline{W}_0$.

In terms of the above schema, one could replace the operator R in Step 3 with the JYC inner approximation operator even when $\underline{W}_0 \nsubseteq B(\underline{W}_0)$. (As noted above, we would define $\underline{W}_0$ by shrinking $B^I(W^{n-1})$ rather than $B^O(W^{n-1})$ as JYC do.)

We have argued above and demonstrated by example that an inner approximation procedure based on an initial condition $\underline{W}_0 \subseteq B(\underline{W}_0)$ might "fail to launch." Fortunately, the new Lemma 4, suggests a way out of this difficulty and provides the needed theoretical foundation for the approach we suggest.

7. Conclusion

In the context of two-player finite action repeated games with perfect monitoring, we developed a new algorithm for computing the set of pure-strategy subgame-perfect equilibrium payoffs. Compared to the well known algorithm of JYC, ours appears to be remarkably quick; indeed over 1,000 times quicker (6 minutes 51 seconds versus 0.34 seconds) in the Cournot example featured in JYC. Our algorithm is inspired by a theoretical result of independent interest: any action profile a can generate at most four extreme points. The most one could hope for from a computational perspective are closed form expressions ("exact" solutions) for the set of extreme points. We demonstrate how, in many cases, this is attainable. We complement our "exact" approach with a new inner approximation procedure that is also quick and finesses some potential difficulties with the approach to inner approximation pioneered by JYC. Finally, we have made available on the web a user friendly implementation of our algorithm, which is available in the supplementary material. This provides a very accessible way for students and researchers to investigate the equilibrium value sets of two-player repeated games, to see their structural properties in terms of generation via actions today and continuation values, to solve for exact solutions, perform inner approximations, and so on. We note that our algorithm exploits the particular geometry of two-player games. The JYC method, on the other hand, is quite general and can be directly applied to games with three or more players. It is unclear at this point whether this leads to computation in reasonable time frames for games with three or more players, because of the "curse of dimensionality" that arises under the JYC approach.

The key to our approach was to develop a finer understanding of how extreme points are generated and, in particular, to determine which incentive compatible continuation values could possibly generate extreme points. In the environment considered here, the JYC approach based on solving families of linear programming problems seems unnecessarily cumbersome and a relatively simple approach based directly on computing all necessary extreme points at every iteration turns out to be very effective. In fact, a straightforward implementation of APS could run into problems if the number of extreme points increased, possibly without bound, along the sequence of iterations. The JYC approach evades this difficulty by limiting the number of extreme points considered to the number of exogenously given "search subgradients." Of course, a central feature of our algorithm is that at every iteration, the number of extreme points is bounded above by $4|A|$, where $|A|$ is the number of action profiles. Some readers have expressed a preference for solutions to games with continuous action spaces. We are a bit mystified by strong tastes in this regard. On the one hand, our view is that the choice between continuous and discrete modeling is purely a matter of analytical convenience. On the

other hand, we are not aware of any application in which in reality the units of measurement are continuous. If anything, the concern should be whether continuous models provide a good approximation to a discrete underlying economic reality.

Turning to other dynamic game settings, our work suggests that it might be productive to look for special structure in how extreme points are generated and to incorporate the relevant structure in the design of the algorithm as we have done here. For example, while it is not the case that our results extend directly to the case of two-player *stochastic* games, many basic ideas have useful analogues.[13] This is a subject of ongoing research by the authors together with Ben Brooks.

Even absent the availability of very sharp structural results, it is quite possible that the number of extreme points does not grow without bound in actual applications. In such cases, it is quite possible that a naive implementation of APS is more speedy than an optimization based approach that involves repeated solution of families of linear programs.

The detailed simplifications obtained here depend on the geometry of two-player games. However, other kinds of structural features might apply to different settings. Three or more player games with perfect monitoring, and, of course, the entire universe of perfect and imperfect public monitoring dynamic games, could, in principle, be usefully investigated from this perspective.

Appendix

Lemma 5. *We have* $W^n \subseteq B(W^{n-1}, u^{n-1}) \subseteq B(W^{n-1}) \subseteq B^n(W^0)$.

Proof. First, note that $B(W,u)$ is increasing in W and decreasing in u. Clearly $W^n \equiv R(W^{n-1}, u^{n-1}) \subseteq B(W^{n-1}, u^{n-1})$. Since $u^{n-1} \geq P(W^{n-1})$, $B(W^{n-1}, u^{n-1}) \subseteq B(W^{n-1}, P(W^{n-1})) \equiv B(W^{n-1})$. Furthermore, $W^{n-1} \subseteq B^{n-1}(W^0)$ implies $B(W^{n-1}) \subseteq B^n(W^0)$. Since $W^1 \subseteq B(W^0)$, the conclusion follows. □

Theorem 4. *The number of extreme points of* V^* *is at most* $3|A|$.

Proof. The proof of Theorem 2 in Section 4 implies that each action profile a such that $g(a) \geq \underline{v} + (1-\delta)/\delta h(a)$ generates at most one extreme point of V^*, $v = g(a)$. Any other action profile a generates at most four extreme points, using continuation values w that are extreme points of $Q(a, V^*, \underline{v})$ such that

$$w_1 = \underline{v}_1 + \frac{1-\delta}{\delta} h_1(a) \quad \text{or} \quad w_2 = \underline{v}_2 + \frac{1-\delta}{\delta} h_2(a). \tag{2}$$

To prove Theorem 4, it is sufficient to narrow down this set of possibilities to *three*. We need to consider two cases.

[13] Promising examples include the games from Thomas and Worrall (1990) and Phelan and Stacchetti (2001). Indeed, these are particularly close in that they embody both perfect monitoring and a two-player structure.

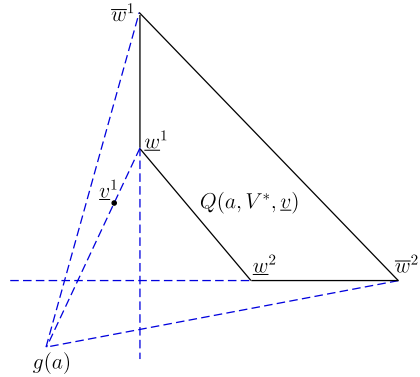

FIGURE 12. Case 2 in the proof of Theorem 4.

Case 1 (The result of this case is due to Ben Brooks): Suppose $g_i(a) \geq \underline{v}_i + (1-\delta)/\delta h_i(a)$ for some $i = 1, 2$, but $g_j(a) < \underline{v}_j + (1-\delta)/\delta h_j(a)$ for $j \neq i$. Then a can only generate extreme points v with continuation values w that, in addition to being extreme points of $Q(a, V^*, \underline{v})$, satisfy $w_j = \underline{v}_j + (1-\delta)/\delta h_j(a)$. Clearly there are at most two such points. For an extreme point w of $Q(a, V^*, \underline{v})$ to generate an extreme point v of V^*, as in the proof of Theorem 3, it is necessary that $(w, v] \cap Q(a, V^*, \underline{v}) = \varnothing$. Note that $(w, v] \subset V^*$. If $w_j > \underline{v}_j + (1-\delta)/\delta h_j(a)$, then by continuity for small $\lambda \in (0, 1)$, $(1-\lambda)w + \lambda v \geq \underline{v} + (1-\delta)/\delta h(a)$ (because also $v_i = (1-\delta)g_i(a) + \delta w_i \geq \underline{v}_i + (1-\delta)/\delta h_i(a)$). Hence $(w, v] \cap Q(a, V^*, \underline{v}) \neq \varnothing$, a contradiction.

Case 2: $g_i(a) < \underline{v}_i + (1-\delta)/\delta h_i(a)$ for both $i = 1, 2$. Then we claim that $Q(a, V^*, \underline{v})$ has at most three extreme points such that (2) holds. Suppose that there are four such points $\overline{w}^i, \underline{w}^i$, $i = 1, 2$, as shown in Figure 12. Then all of these points have to be on the boundary of V^*. This cannot be the case because $\underline{w}^1$ is strictly inside the triangle with vertices $\overline{w}^1, \overline{w}^2, \underline{v}^1 = (1-\delta)g(a) + \underline{w}^1 \in V^*$, a contradiction. □

REMARK. We can modify the R operator along the lines of Case 1 of the proof of Theorem 4; however, there appears to be no natural modification that covers Case 2 in a manner that preserves the monotonicity of R in W. So unlike Theorem 1, Theorem 4 does not translate directly into an algorithmic procedure.

REFERENCES

Abreu, Dilip (1986), "Extremal equilibria in oligopolistic supergames." *Journal of Economic Theory*, 39, 191–225. [317]

Abreu, Dilip, David Pearce, and Ennio Stacchetti (1986), "Optimal cartel equilibria with imperfect monitoring." *Journal of Economic Theory*, 39, 251–269. [315]

Abreu, Dilip, David Pearce, and Ennio Stacchetti (1990), "Toward a theory of discounted repeated games with imperfect monitoring." *Econometrica*, 58, 1041–1063. [313, 314, 315]

Cronshaw, Mark B. and David G. Luenberger (1994), "Strongly symmetric subgame perfect equilibria in infinitely repeated games with perfect monitoring and discounting." *Games and Economic Behavior*, 6, 220–237. [315]

Judd, Kenneth, Sevin Yeltekin, and James Conklin (2003), "Computing supergame equilibria." *Econometrica*, 71, 1239–1254. [314, 322]

Mailath, George J. and Larry Samuelson (2006), *Repeated Games and Reputations*. Oxford University Press, New York. [315]

Phelan, Christopher and Ennio Stacchetti (2001), "Sequential equilibria in a Ramsey tax model." *Econometrica*, 69, 1491–1518. [336]

Thomas, Jonathan and Timothy Worrall (1990), "Income fluctuations and asymmetric information: An example of a repeated principal–agent problem." *Journal of Economic Theory*, 51, 367–390. [336]

Submitted 2012-8-6. Final version accepted 2013-1-15. Available online 2013-1-15.