Computation of subgame-perfect equilibria of repeated games with perfect monitoring and public randomization*

Bruno Sultanum
Pennsylvania State University
brunosalcedo.com
bruno@psu.edu

Bruno Sultanum
Pennsylvania State University
sultanum.com
sultanum@psu.edu

March 2012

Abstract We provide an efficient way to implement the Abreu-Pearce-Stacchetti algorithm to approximate the set of SPNE payoffs for repeated games with perfect monitoring and public randomization. The algorithm generates a decreasing sequence of sets that converge to the set of equilibrium payoffs. Each iteration is obtained by applying a set operator to the previous iteration. We use the fact that the operator maps polyotopes to polytopes to provide an efficient way to compute it that drastically reduces the computation time required. We also show that the set of equilibrium payoffs is a polytope, this suggests that the number of vertices of the sequence of iterations will remain bounded.

Keywords Generalized dynamic programming · APS · Repeated games · Computational game theory

JEL classification C61 · C73 · D70

In repeated interactions, agents can implement large sets of outcomes by coordinating their choices on public information on the history of play. Economic theory has thus found a great interest in repeated games. However, the lack of closed form solution for the set of equilibria makes numerical approximations necessary. Current algorithms are based on Abreu et al. (1990), hereon APS, which introduces a set operator B describing the set of payoffs that *could* be obtained in equilibrium, given a set of equilibrium

*We wish to gratefully acknowledge the comments, advise and supervision of Ed Green, Vijay Krishna, Paul Grieco and Nail Kashaev. This is a preliminary and incomplete version that was being prepared to be submitted for publication in the format of a short note or letter. It will no longer be submitted in view of the work of Abreu and Sannikov (2013) which anticipates our algorithm and results.

continuation values. APS shows that iteratively applying B to any compact set which is large enough, generates a decreasing sequence of approximations that converge to the set of equilibrium payoffs. Equilibrium strategies can be derived form this set.

Implementing the APS algorithm numerically is not straightforward, partly because it is difficult to represent generic infinite sets in a computer. Introducing public randomization helps because all the relevant sets become convex (Cronshaw and Luenberger, 1994), but the representing or approximating convex sets can be non-trivial. Different papers use different kind of approximations, for instance, **?** uses trigonometric approximations, while Cronshaw (1997) and Judd et al. (2003) (hereon JYC) use convex polytope approximations. In a related setting, Sannikov (2007) considers games in continuous time with particular monitoring structures, and solves a differential equation to compute the boundary of the equilibrium set.

We show that B maps polytopes to polytopes, and thus can be computed with a fast direct algorithm. This drastically reduces the computation time needed in comparison with other implementations. While some alternative methods can approximate generic convex sets efficiently, they are inefficient for computing polytopes with finitely many extreme points. Our characterization of B is similar to the one in Stahl (1991), who finds a closed form solution for the infinitely repeated prisoner's dilemma. We show that this characterization can be used to compute E efficiently for any finite game.

The algorithm could become impractical if the number of vertices increased without a bound. However, we show that the equilibrium set is actually a polytope, whose number of vertices can be bounded by a polynomial function of the size of the game. We provide a simple proof of this result for the two player case in §5. We conjecture that the result holds for n-player games, but extending the proof is not trivial.

§6 reports some numerical results. JYC report computation times of around 45 minutes for a discretized Cournot duopoly game with 15 actions per firm. We can compute the set of equilibrium payoffs for the same game with 20 actions per firm in a few seconds. The computation might be accelerated even more given that the algorithm can be easily parallelized.[1] This dramatic improvement in speed may enable the use of simulation methods to estimate games in applied work.[2]

1. Environment

A finite set of players interacts for a countable number of periods indexed by $t \in \mathbb{N}$. Let I denote the set of players, i, j denote typical players, and, for each player i, let $-i = I \backslash \{i\}$ denote the set of i's opponents. At the beginning of each period there is a publicly observed signal, drawn from a known i.i.d. distribution with rich support.[3]

[1] The computation of $B_a(W)$ for each a are independent and thus can be run in parallel.

[2] For a motivation of the importance of this claim see for instance Aguirregabiria and Mira (2007) or Bajari et al. (2007). While these papers consider dynamic games and we have not yet done this, we believe that the relevant operator for dynamic games *with finitely many states* also maps polytopes to polytopes and thus a similar implementation is possible.

[3] A large number of distributions work, we only need that it be rich enough so that players can condition upon it to generate any distribution on the set of action profiles. For the precise meaning of

After observing the signal, players play a simultaneous move game and the chosen actions are publicly observed. Since players can condition their choices on past actions and public signals, our environment is a repeated game with perfect monitoring and public randomization. Throughout this document we assume that the reader is familiar with the theory of repeated games, for a more detailed and rigorous analysis see §2 in Mailath and Samuelson Mailath and Samuelson (2006).

The stage game played at each period is characterized by a tuple $\Big(A, u\Big)$. $A = \times_i A_i$ is the set of action profiles, where A_i is a *finite* set of actions available to i. As usual, $A_{-i} = \times_{j \neq i} A_j$ denotes the set of action profiles for i's opponents. $u = (u_i) : A \to \mathbb{R}^I$ is such that u_i represents i's preferences over action profiles.

Players discount future payoffs with a common discount factor $\delta \in (0, 1)$. In the repeated game, each player tries to maximize his average discounted expected payoff given by:

$$\mathbb{E}\left[(1-\delta) \sum_{t=0}^{\infty} \delta^t u_i(a_t) \right], \tag{1}$$

where $\{a_t\}$ is the sequence of action profiles played.

We are interested in the set $E \subseteq \mathbb{R}^I$ of payoff vectors corresponding to subgame perfect Nash equilibria (SPNE) of the repeated game. Loosely speaking, SPNE are strategy profiles with the property that, conditional on every possible history of the game, no player could benefit from unilaterally deviating.

2. Generalized dynamic programming

It is well known that E is compact and convex and not empty, however there is no known closed form characterization for general games. Our analysis starts from the APS characterization, adapted to games with perfect monitoring and public randomization by Cronshaw and Luenberger Cronshaw and Luenberger (1994). This section develops some elements of this theory, in order to justify our algorithm.

Start from an arbitrary compact set of vector payoffs $W \subseteq \mathbb{R}^I$ used as an approximation of E. For each action profile $a \in A$ define $F_a(W), C_a(W), B_a(W) \subseteq \mathbb{R}^I$ to be the sets given by:[4]

$$F_a(W) = (1-\delta)\Big\{u(a)\Big\} + \delta W, \tag{2}$$

$$C_a(W) = \Big\{(1-\delta)\bar{u}(a) + \delta \underline{w}_i(W)\Big\} + \mathbb{R}^I_+, \tag{3}$$

$$B_a(W) = F_a(W) \cap C_a(W), \tag{4}$$

where $\underline{w}, \bar{u}(a) \in \mathbb{R}^I$ are the vectors given by:

$$\underline{w}_i(W) = \min_{w \in W} w_i \quad \text{and} \quad \bar{u}_i(a) = \max_{a'_i \in A_i} u(a'_i, a_{-i}). \tag{5}$$

"richness" see Aumann (1974).

[4]For sets $V, W \subseteq \mathbb{R}^I$ and numbers $\lambda \in \mathbb{R}$, we use the notation $\lambda V = \{\lambda v \mid v \in V\}$ and $V + W = \{v + w \mid v \in V, w \in W\}$.

If players play some pure action profile $a \in A$ in the first period, then their total payoff can be decomposed as $v = (1-\delta)u(a) + \delta v'$ for some continuation value $v' \in \mathbb{R}^I$. The set $F_a(W)$ is the set of payoffs vectors that can be decomposed in such a way, with continuation values belonging to the set W.

$F_a(W)$ describes feasible payoffs without any incentive considerations. Now consider a strategy profile according players choose a in the first period. If some player i deviated to a different action in such period, he could get at most $\bar{u}(a)$ and the other players could punish him by switching to strategies that will give him the worst available continuation value $\underline{w}_i$. Hence, C_a is exactly the set of payoffs that will make all players willing to play a on the first period, conditional on their continuation values belonging to W.

$B_a(W)$ is then the set of payoffs that can be obtained playing a, having a continuation value in W and such that no player would want to deviate on the first period. Since we allow for public randomization, we can mix actions on the first periods to get any payoffs in the *convex* set $B(W)$ given by:[5]

$$B(W) = \text{co}\left(\bigcup_{a \in A} B_a(W)\right) \tag{6}$$

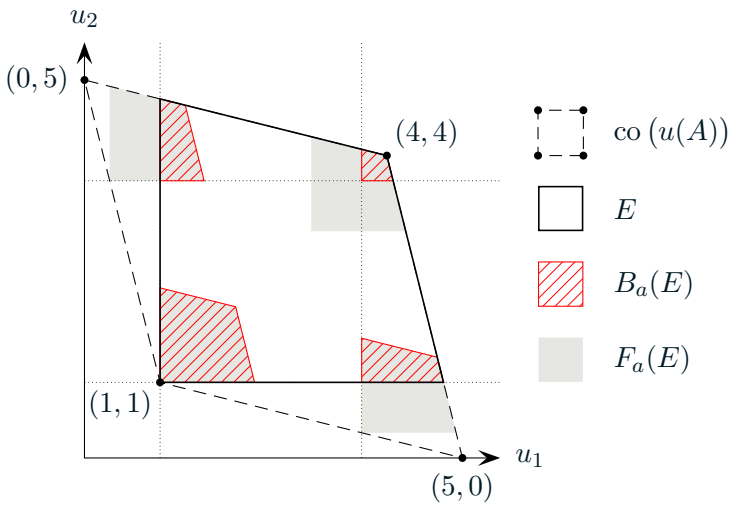

Figure 1 Computing $E = B(E)$ for a prisoner's dilemma.

Figure (1) illustrates the construction of $B(W)$ with $W = E$ for a prisoner's dilemma example. The set of feasible payoffs is represented by the dashed polygon, and the set of equilibrium payoffs $E = B(E)$ by the solid polygon. For each $a \in A$, there is a polygon shaded in gray representing the set $F_a(E)$ consisting of the set of linear combinations between the vector $u(a)$ and vectors in E with weights $(1-\delta)$ and δ. The hatched polygons then represent $B_a(E)$ which are obtained by adding incentive compatibility constraints.

The APS algorithm starts from any compact initial guess $W \subseteq \mathbb{R}^I$ that is known to contain E, and applies the B operator iteratively until a convergence criterion is met. It is justified by the well known fact that, for every compact set $W \in \mathbb{R}^I$, if $E \subseteq W$ then

[5]For every set $V \subseteq \mathbb{R}^I$ we use the notation $\text{co}(V)$ too denote its convex hull.

$\{B^n(W)\}$ is a $\subseteq$-decreasing sequence that converges to E with respect to the Hausdorff metric, see for instance Cronshaw and Luenberger (1994).

3. Efficient computation of the APS operator

Different papers suggest different ways of implementing the APS algorithm, which basically boil down to different methods for approximating the B operator. For example, JYC use convex polytopes to generate inner and outer approximations, and Cronshaw uses polynomial interpolations of points known to be in the boundary. Both approaches involve a time consuming optimization step.

A convex polytope, or simply polytope, is a set $W \subseteq \mathbb{R}^I$ that can be described by a finite set of affine inequalities, i.e., such that $W = \{w \in \mathbb{R}^I \mid Gw \leq d\}$ for some $G \in \mathbb{R}^{I \times N}$ and some $d \in \mathbb{R}^N$. Simple algebra shows that for every polytope $W = \{w \in \mathbb{R}^I \mid Gw \leq d\}$, and every $a \in A$, we can write $B_a(W)$ as:

$$B_a(W) = \left\{ v \in \mathbb{R}^I \;\middle|\; \begin{pmatrix} G_W \\ -I \end{pmatrix} v \leq \begin{pmatrix} \delta d_W + (1-\delta) G u(a) \\ -\delta \bar{u}(a) - (1-\delta)\underline{w}(W) \end{pmatrix} \right\}. \tag{7}$$

This implies that $B(W)$ is a polytope. Since we can always pick the polytope $F = \text{co}(u(A))$ as an initial guess, we can guarantee that all the iterates $B^n(F)$ will be polytopes. Then, computing $B_a(W)$ requires almost no computations. Hence, the only computations needed to compute $B(W)$ consist of finding the convex hull of $\cup_a B_a(W)$. For that purpose we suggest a different approach that looks at vertices instead of facets.[6]

Going back to figure (1), the vertices of $F_a(W)$ are given by $(1-\delta)u(a) + \delta y_m$, where $\{y_m\}$ are the vertices of W. Finding the vertices of each $B_a(W)$ boils down to identifying which vertices of $F_a(W)$ satisfy the incentive compatibility constraints, and finding the intersections of $B_a(W)$ with the incentive compatibility hyperplanes. For two player games this can be done efficiently by considering a large number of cases.[7] For games with more players we propose a brute force approach. Let V_a denote the vertices of B_a. Start with the set of vertices of F_a as an initial approximation of V_a, and do the following for each player i:

step 1: Find the set Y_a composed by the points of V_a that satisfy i's incentive constraint, and let $X_a = V_a \backslash Y_a$.

step 2: For each $y \in Y_a$ and each $x \in X_a$, compute the intersection of the line $\overline{yx}$ and i's incentive compatibility constraint.

step 3: Find the vertices of the convex hull of the set of such intersections, let Z_a denote such set.

step 4: Update V_a to be $Y_a \cup Z_a$.

Once again, the only time consuming computation involved is related to finding convex hulls. There are a different number of methods to do so. For two player games

[6] Recall that the set of vertices of polygon is the $\subseteq$-smallest set whose convex hull equals the polygon.

[7] This makes for a very long code but a very short running time.

the most efficient one is Graham's scan given the distribution of the points. However, this method is difficult to generalize to games with many players. For such cases there are still algorithms that solve the problem but they are time consuming. See the table to see whether this is a problem or not (I still don't know because we haven't finished computing the table).

One might be concerned about the number of vertices of the sequence $B^n(W)$ increasing without a bound, making the computation inefficient or impossible. We have not yet been able to establish a bound for the number of vertices of the sequence, however, it turns out that E is always a polytope. This suggests that the number of vertices of $B^n(W)$ will always remain finite (and moderately small). We only prove the result for two player games. Although we are yet to come up with a formal argument, we believe that the result is true for games with an arbitrary (finite) number of players.

Proposition 1 *For two player games, E is a polygon with at most $5\#A$ vertices.*

4. Some numerical experiments

Figure (2) show the sequence of iterations generated by the algorithm for a prisoner's dilemma game. The left panel corresponds to a sequence starting from $\text{co}(u(A))$ as an initial approximation, while the right panel corresponds to a sequence starting from a strictly larger polygon. It is noteworthy that the number of vertices of the approximations increases after the first application of B, but then decreases monotonically. Also, the sequence of vertices appear to follow lines, most of which are parallel to payoff differences $u(a) - u(a')$. We observe similar patterns for different games.

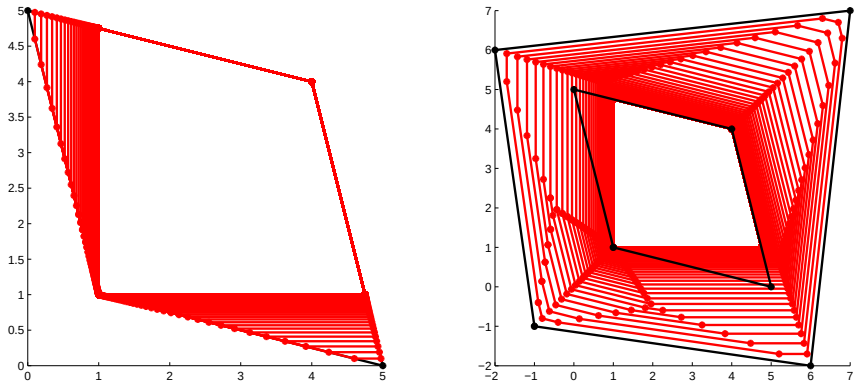

Figure 2 Sequence of approximations for a prisoner's dilemma.

To be able to compare the performance of our program versus other existent algorithms, we considered a Bertrand duopoly example with differentiated products from

JYC. We only write the profits (payoffs) as a function of prices (actions):

$$\pi_i(p) = \Big(p_i - c\Big) \cdot \max\Big\{0\,,\, 1 + ap_{-i} + bp_i\Big\}, \tag{8}$$

where $c, a, b \in \mathbb{R}_{++}$. We used the parameter values $a = 0.5$, $b = 0.25$, $c = 0.1$. Table (1) reports our average computation times in seconds (over 100 runs) for different values of n and δ.

n	$\delta = 0.3$	$\delta = 0.5$	$\delta = 0.7$	$\delta = 0.9$
5	0.24	0.43	0.49	0.48
10	0.41	1.02	1.05	1.07
20	1.00	3.35	3.45	3.47
50	5.55	21.72	22.43	22.58

Table 1 Computing times in seconds for a Bertrand duopoly example.

Figure (3) shows the resulting approximation of the equilibrium set with $\delta = 0.6$ and $n = 250$. The fact that it appears to have curvature suggests that E need not be a polytope for games with infinite action spaces. Hence, our method can be applied only to finite games or finite approximations of infinite games.

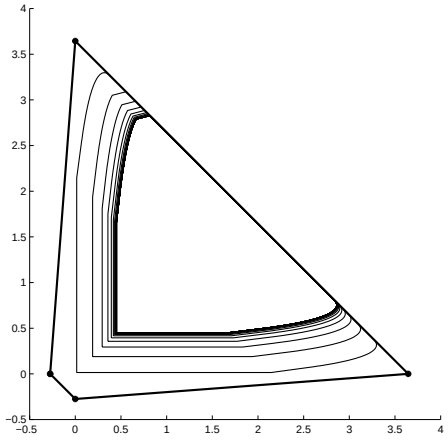

Figure 3 Equilibrium for Bertrand duopoly game with $n = 100$ and $\delta = 0.6$.

References

Abreu, D., Pearce, D., and Stacchetti, E. (1990). Toward a theory of discounted repeated games with imperfect monitoring. *Econometrica*, 58(5):1041–1063.

Abreu, D. and Sannikov, Y. (2013). An algorithm for two player repeated games with perfect monitoring. *Theoretical Economics, forthcomming*.

Aguirregabiria, V. and Mira, P. (2007). Sequential estimation of dynamic discrete games. *Econometrica*, 75(1):1–53.

Aumann, R. J. (1974). Subjectivity and correllation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96.

Bajari, P., Benkard, L., and Levin, J. (2007). Estimating dynamic models of imperfect competition. *Econometrica*, 75(5):1331–1370.

Cronshaw, M. B. (1997). Algorithms for finding repeated game equilibria. *Computational Economics*, 10(2):139–168.

Cronshaw, M. B. and Luenberger, D. G. (1994). Strongly symmetric subgame perfect equilibria in infinitely repeated games with perfect monitoring and discounting. *Games and Economic Behavior*, 6(2):220–237.

Judd, K. L., Yeltekin, S., and Conklin, J. (2003). Computing supergame equilibria. *Econometrica*, 71(4):1239 – 1254.

Mailath, G. J. and Samuelson, L. (2006). *Repeated Games and Reputations*. Oxford University Press.

Sannikov, Y. (2007). Games with imperfectly observable actions in continuous time. *Econometrica*, 75(5):1285–1329.

Stahl, D. O. (1991). The graph of prisoner's dilemma supergame payoffs as a function of the discount factor. *Games and Economic Behavior*, 3:368–384.

A. Proof of the main theorem

The set of extreme points of a convex set $W \subseteq \mathbb{R}^I$ is the set of points that cannot be written as a linear combination of other points in W, that is $\text{Ext}(W) = \{w \in W \mid w \notin \text{co}(W \backslash \{w\})\}$. The KreinÂ–Milman theorem states that $W = \text{co}(\text{Ext}(W))$. Furthermore, a convex set $W \subseteq \mathbb{R}^I$ is a polytope if and only if $\text{ext}(W)$ is finite and, in that case, the notion of extreme point corresponds to the notion of vertex. The proof that we provide counts the extremum points of E and shows that there are at most $5\#A$ of them. For the proof we use the notation: $X = \text{Ext}(E)$, $F_a = F_a(E)$, $C_a = C_a(E)$ and $E_a = B_a(E)$.

Lemma 2 *For every SPNE payoff $v \in E_a$, if no player's incentive compatibility constraints are binding, then there is another equilibrium payoff in the line segment connecting v with $u(a)$, i.e., $(\forall a \in A)(\forall v \in E_a \backslash C_a^0)(\exists \lambda \in (0,1))(\lambda u(a) + (1-\lambda)v \in E_a)$*

Proof. Fix some $a \in A$ and some $v \in E_a \backslash C_a^0$. Since $E_a \subseteq F_a$ we know that $w = \frac{1}{\delta} v + \frac{\delta - 1}{\delta} u(a) \in E$. By convexity of E we know that $v^\mu = \mu w + (1-\mu)v \in E$ for every $\mu \in (0,1)$. Since C_a^0 is an open set, we know that for λ close to 1 we have $(1-\delta)u(a) + \delta v^\mu \in C_a^0 \cap F_a \subset E_a$. Finally notice that $(1-\delta)u(a) + \delta v^\mu = \lambda u(a) + (1-\lambda)v$ for $\lambda = (1-\delta)(1-\mu) \in (0,1)$. ■

Lemma 3 *Every extreme SGPNE payoff can be obtained playing a pure strategy on the first period, i.e., $X = \bigcup_{a \in A} E_a$.*

Proof. Since $E = \text{co}(\cup_a E_a)$ then we know that any payoff $v \in E \backslash \cup_a E_a$ can be written as a linear combination of payoffs in $\cup_a E_a \subseteq E$ and thus cannot be an extreme point. ■

Lemma 4 *Every extreme SGPNE payoff corresponds either to a pure action payoff or has some player's incentive constraint saturated, i.e.,* $(\forall a \in A)(X \cap E_a \subseteq \{u(a)\} \cup \partial C_a)$.

Proof. Now consider any $v \in E_a \cap C_a^0 \backslash \{u(a)\}$. By lemma (2) there exists some $\lambda \in (0,1)$ such that $v^\lambda = \lambda u(a) + (1-\lambda)v \in E_a$ and since $v \in E_a$ we know that $w = \frac{1}{\delta}v + \frac{\delta-1}{\delta}u(a) \in E$. Now notice that we have $v = \mu v^\lambda + (1-\mu)w$ for $\mu = (1-\delta)/(1-\delta+\delta\lambda) \in (0,1)$. Hence v cannot be an extreme point. ■

Fix some $a \in A$ and choose any three distinct vertices $v^1, v^2, v^3 \in V \cap E_a$ saturating the same player's incentive constraint, i.e. such that $v_i^1 = v_i^1 = v_i^3 = (1-\delta)\bar{u}_i(a) + \delta\underline{w}_i$ for some $i \in I$. Without loss of generality we can assume that that $v_{-i}^1 > v_{-i}^2 > v_{-i}^3$. However, this implies that v^2 is a linear combination of v^1 and v^2 and thus it cannot be an extreme point of E. Hence there are *at most* two vertices in $X \cap E_a$ that saturate each player's incentive compatibility constraint. By lemma (4) and the fact that there are only two players, this implies that $\#(X \cap E_a) \leq 5$. Hence, by lemma (3) we have that $\#X = \sum_{a\in A} \#(X \cap E_a) \leq \sum_{a\in A} 5 = 5\#A$. Finally, by the Krein-Milman theorem we know that $E = \text{co}(X)$ and hence E is the convex hull of at most $5\#A$ vertices.

2ED