FINDING ALL EQUILIBRIA IN GAMES OF STRATEGIC COMPLEMENTS

FEDERICO ECHENIQUE

Abstract. I present a simple and fast algorithm that finds all the pure-strategy Nash equilibria in games with strategic complementarities. This is the first non-trivial algorithm for finding all pure-strategy Nash equilibria.

1. Introduction

I present an algorithm that finds all the pure-strategy equilibria in n-player games with strategic complementarities (GSC).

GSC were first formalized by Topkis (1979), and were introduced by Vives (1990) to economics.

GSC are important in many areas in economics. For example, both price- and quantity-competition oligopoly models can be modeled as GSC; arguably covering most static market models one may wish to consider. See Milgrom and Roberts (1990), Milgrom and Shannon (1992), Topkis (1998), and Vives (1999) for economic examples of GSC.

Many models in operations research have recently been analized as GSC. Examples are Lippman and McCardle (1997), Bernstein and Federgruen (2004a), Bernstein and Federgruen (2004b), Netessine and Rudi (2003), Netessine and Shumsky (2003), Cachon and Lariviere (1999), and Cachon (2001). See Cachon and Netessine (2003) for a survey.

I wish to emphasize three features of the algorithm:

(1) It finds all pure-strategy equilibria, but no mixed-strategy equilibria. The omission is justified because mixed-strategy equilibria are not good predictions in GSC (Echenique and Edlin, 2002). Further, pure-strategy equilibria always exist in GSC (Topkis, 1979; Vives, 1990).
(2) It is generally very fast. For example, it needs less than 10 seconds to find all equilibria in a game with 3.6×10^9 strategy profiles.

Date: March 15, 2004.

JEL Classification. C63, C72.

I thank Gerard Cachon, Eddie Dekel, Juan Dubra, Matt Jackson, Ivana Komunjer, Andy McLennan, John Rust, Bill Sandholm, Ilya Segal, Chris Shannon, and Bernhard von Stengel for comments and suggestions. I also thank seminar audiences in a number of institutions for their comments.

(3) It is simple. I use the algorithm "by hand" on some bimatrix games to show that the algorithm is very simple to apply.

There are many algorithms for finding *one* equilibrium, called a "sample" equilibrium (see the surveys by McKelvey and McLennan (1996) and von Stengel (2002)). But there is currently only one method for finding *all* pure equilibria: the "underlining" method one teaches undergraduates—fix one player, for each strategy-profile of the player's opponents, find her best-response, and then check if some opponent wants to deviate. The method is close to testing all the game's strategy-profiles to see if they are equilibria; I shall call this method the "trivial algorithm." Not surprisingly, the trivial algorithm is very slow, and computationally infeasible on large games.

Some algorithms find a sample equilibrium that survives an equilibrium refinement—typically perfection (a recent example is von Stengel, van den Elzen, and Talman (2002); see McKelvey and McLennan (1996) and von Stengel (2002) for other examples). This is some times adequate, but it is in general restrictive: there is normally no guarantee that only one equilibrium survives the refinement, and the refinements do not always have bite. (An exception is Judd, Yeltekin, and Conklin (2000); their algorithm finds all perfect-equilibrium payoffs in repeated games.)

The algorithm I present is based on Topkis's (1979) results that Robinson's (1951) method of "iterating best-responses" finds an equilibrium in GSC (see also Vives (1990)), so the algorithm uses different—and simpler—ideas than the more recent literature on finding equilibria.

I shall not apply the algorithm to economic, or operations research, examples. The paper presents a method, and it argues that the method works well. So the applications of the algorithm either illustrate how it works or show that the algorithm is fast. Nevertheless, there are many applications in operations and economics. I give two examples:

- Supply-chain analysis. Cachon (2001) studies inventory competition in a supply chain with retailers that face stochastic demands. The resulting game is a GSC. Cachon compares numerically the system-optimal solution to the Nash equilibria of the game. He uses exhaustive search (after identifying the extremal equilibria—see Section 3) to find all Nash equilibria. The algorithm I introduce can be used instead of exhaustive search; it will be more efficient. [1]
- Oligopoly models. Under mild conditions, Bertrand oligopoly with differentiated products is a GSC (Milgrom and Shannon, 1994). In

[1]Other applications to supply-chain analysis include discretized versions of Lippman and McCardle (1997).

turn, Bertrand oligopoly with differentiated products is a very common market structure. The algorithm has then natural applications in the empirical analysis of markets.

One important example is the evaluation of mergers by the US Department of Justice. The Department of Justice needs to predict the consequences of mergers between firms. They postulate a model of a market—they often use Bertrand models with differentiated products, see for example Werden, Froeb, and Tschantz (2001) or Crooke, Froeb, Tschantz, and Werden (1997)—-and compute a Nash equilibrium before and after the merger of some firms in the market. [2]

But their conclusions might of course change if they could find all equilibria before and after the merger. For example, the merger could have no effect on price if you look at some equilibria, but a large price increase if you compare most equilibria.

Besides Bertrand oligopoly, the algorithm can also be applied to Cournot oligopoly. By exploiting Amir and Lambson's (2000) ideas for using GSC-techniques in Cournot oligopoly, one can easily adapt the algorithm to find all symmetric equilibria in Cournot models as well. Arguably, the algorithm is applicable to most static models of a market one may wish to consider in applied work.

Finally, let me mention how the paper contributes to the theory of GSCs. The seminal papers on GSCs (Topkis, 1979; Vives, 1990; Milgrom and Roberts, 1990) stress how the smallest and largest equilibria can be obtained as limits of certain monotone sequences. The source of the sequences is interpreted both as out-of-equilibrium adjustments (learning), and as an algorithm. But no equilibrium-selection theory says that the smallest, or largest, equilibrium are good predictions. It is important then to use GSC methods to approach other equilibria, and the results I present here are the first to do that.

The paper is organized as follows. Section 2 presents some preliminary definitions and results. Section 3 shows informally how the algorithm works. Section 4 defines the algorithm and presents the main results of the paper. Section 5 develops two simple examples. Section 6 outlines the argument that the algorithm is fast, and explains the benchmark algorithm. Section 7 presents computational results for simulations of GSC. Section 8 explains why the algorithm is normally fast. Section 9 discusses an algorithm for a special class of GSC.

[2]The software they use is in `http://mba.vanderbildt.edu/luke.froeb/software/`

2. Preliminary Definitions and Results

2.1. **Basic Definitions and Notation.** Let $X \subseteq \mathbf{R}^n$, and $x, y \in \mathbf{R}^n$. Denote the vector $(\max\{x_i, y_i\})$ by $x \vee y$, and the vector $(\min\{x_i, y_i\})$ by $x \wedge y$. Say that X is a *lattice* if, whenever $x, y \in X$, $x \wedge y, x \vee y \in X$.

If X is a lattice, a function $f : X \to \mathbf{R}$ is *quasi-supermodular* if for any $x, y \in X$, $f(x) \geq f(x \wedge y)$ implies $f(x \vee y) \geq f(y)$ and $f(x) > f(x \wedge y)$ implies $f(x \vee y) > f(y)$. Quasi-supermodularity is an ordinal notion of complementarities; it was introduced by Milgrom and Shannon (1994). Let $T \subseteq \mathbf{R}^m$. A function $f : X \times T \to \mathbf{R}$ satisfies the *single-crossing condition* in (x, t) if whenever $x < x'$ and $t < t'$, $f(x, t) \leq f(x', t)$ implies that $f(x, t') \leq f(x', t')$ and $f(x, t) < f(x', t)$ implies that $f(x, t') < f(x', t')$.

For two subsets A, B of X, say that A is smaller than B in the *strong set order* if $a \in A$, $b \in B$ implies $a \wedge b \in A, a \vee b \in B$. Let $\phi : X \twoheadrightarrow X$ be a correspondence. Say that ϕ is *increasing in the strong set order* if, whenever $x \leq y$, $\phi(x)$ is smaller in the strong set order than $\phi(y)$. A detailed discussion of these concepts is in Topkis (1998).

An n-player normal-form game (a game, for short) is a collection $\Gamma = \{(S_i, u_i) : i = 1, \ldots n\}$, where each player i is characterized by a set of possible strategies, S_i, and a payoff function $u_i : S \to \mathbf{R}$, where $S = \times_{j=1}^n S_j$. Say that players have *strict preferences* if, for all i and $s_{-i} \in S_{-i}$, the function $s_i \mapsto u_i(s_i, s_{-i})$ is one-to-one.

For each player i, let $\beta_{i,\Gamma}$ denote i's *best-response correspondence* in Γ—the correspondence defined by

$$\beta_{i,\Gamma}(s) = \operatorname{argmax}_{\tilde{s}_i \in S_i} u_i(\tilde{s}_i, s_{-i}).$$

And let $\beta_\Gamma(s) = \times_{i=1}^n \beta_{i,\Gamma}(s)$ denote the game's best-response correspondence. When Γ is understood I shall write β_i for $\beta_{i,\Gamma}$ and β for β_Γ.

A point $s \in S$ is a *Nash equilibrium* if $s \in \beta(s)$. Let $\mathcal{E}(\Gamma)$ be the set of all Nash equilibria of Γ. When Γ is understood, I shall write $\mathcal{E}$ for $\mathcal{E}(\Gamma)$.

2.2. **The Model.** Say that a game $\Gamma = \{(S_i, u_i) : i = 1, \ldots n\}$ is a *finite game of strategic complementarities* (GSC) if, for each i,

- $S_i \subseteq \mathbf{R}^{d_i}$ is a finite lattice,
- $s_i \mapsto u_i(s_i, s_{-i})$ is quasi-supermodular for all s_{-i},
- and $(s_i, s_{-i}) \mapsto u_i(s_i, s_{-i})$ satisfies the single-crossing property.

The positive integer d_i is the number of dimensions of player i's strategies. I shall assume, in addition, that

- $S_i = \{1, 2, \ldots K_i\}^{d_i}$.

The assumption that $S_i = \{1, 2, \ldots K_i\}^{d_i}$ simplifies notation, but I should stress that all my results hold for arbitrary finite GSC.

The set of Nash equilibria of a GSC is a complete lattice (Zhou, 1994).

Remark 1. One can think of the model as a discretized version of a game with continuous strategy spaces, where each S_i is an interval in some Euclidean space of dimension d_i. For an example, see Section 7.

2.3. **Auxiliary results.** First, GSC have monotone best-response correspondences:

Lemma 2. ***(Milgrom and Shannon, 1994)*** *For all i, β_i is increasing in the strong set order, and* $\inf \beta_i(s), \sup \beta_i(s) \in \beta_i(s)$.

See Milgrom and Shannon (1994) for a proof.

Second, I need some results and notation for games where we restrict the strategies that players can choose: For each $s_i \in S_i$, let $S_i^r(s_i) = \{\tilde{s}_i \in S_i : s_i \leq \tilde{s}_i\}$ be the strategy space obtained by letting i choose any strategy in S_i, as long as it is larger than s_i. For each strategy profile $s = (s_1, \dots s_n) \in S$, let $S^r(s) = \times_{i=1}^n S_i^r(s_i)$. Denote by $\Gamma^r(s)$ the game where each player i is constrained to choosing a strategy larger than s_i. Then,

$$\Gamma^r(s_1, \dots s_n) = \left\{ (S_i^r(s_i), u_i|_{S_i^r(s_i)}) : i = 1, \dots n \right\}.$$

The following lemmata are trivial.

Lemma 3. *If Γ is a GSC, then so is $\Gamma^r(s)$, for any strategy profile $s \in S$.*

Lemma 4. *If s is a Nash equilibrium of Γ, and $z \leq s$, then s is a Nash equilibrium of $\Gamma^r(z)$.*

Lemma 3 and Lemma 4 follow immediately from the definitions of GSC and of Nash equilibrium.

Third, I shall exploit some previous results on finding equilibria in GSC. The method of iterating β until an equilibrium is found is normally attributed to Robinson (1951). Topkis (1979) proved that the method works in GSC. I call this method the "Robinson-Topkis algorithm."

Algorithm 1. *The following are three variants of the Robinson-Topkis algorithm.*

- $\underline{T}(s)$*: Start with $s^0 = s$. Given $s^k \in S$, let $s^{k+1} = \inf \beta_\Gamma(s^k)$. Stop when $s^k = s^{k+1}$.*
- $\overline{T}(s)$*: Start with $s^0 = s$. Given $s^k \in S$, let $s^{k+1} = \sup \beta_\Gamma(s^k)$. Stop when $s^k = s^{k+1}$.*
- $T^r(s)$*: Do algorithm $\underline{T}(s)$ in $\Gamma^r(s)$.*

Lemma 5. ***(Topkis, 1979)*** $\underline{T}(\inf S)$ *stops at the smallest Nash equilibrium of Γ, and* $\overline{T}(\sup S)$ *stops at the largest Nash equilibrium of Γ.*

See Topkis (1979) (or Topkis (1998)) for a proof of Lemma 5.

Remark 6. Note that $\underline{T}(\inf S)$ is faster than "iterating $\inf \beta_\Gamma(s^k)$" suggests. When the algorithm has to find $\inf \beta_\Gamma(s^k)$, it knows that searching in the interval $[s_k, \sup S]$ is enough. The sequence $\{s_k\}$ is monotone increasing, so each iteration of $\underline{T}(\inf S)$ is faster the previous iteration. A similar thing happens to $\overline{T}(s)$ and $T^r(s)$.

A "round-robin" version of RT—where players take turns in best-responding instead of jointly best-responding in each iteration—is faster than the version above (see Topkis (1979)). I use the version above because its notation is easier. All results in the paper hold if "round-robin" RT is substituted for RT. In fact, the results reported in Section 7 are from the round-robin implementation.

3. How it works

"In the authors' experience, an important idea in organizing the analysis of a game by hand is to find one equilibrium, then ask how other equilibria might differ from this one; there is currently no substantiation of this wisdom in theory or computational experience." (McKelvey and McLennan, 1996, p. 28)

I shall use an example to explain how the algorithm works. The explanation shows that the algorithm is a—rudimentary—substantiation of McKelvey and McLennan's wisdom.

Consider a two-player GSC, Γ. Suppose that player 1 has strategy set $S_1 = \{1, 2, \ldots 15\}$, and player 2 has $S_2 = \{1, 2, \ldots 11\}$ (the numbers do not matter, they just happen to give a nice picture in Figure 1). The players' joint strategy space, $S_1 \times S_2$, is in Figure 1. Suppose that we have calculated the players' best-response functions (to make things simple, assume best-responses are everywhere unique), β_1 and β_2. The game's best-response function is β, where $\beta(s_1, s_2) = (\beta_1(s_2), \beta_2(s_1))$. Because Γ is a GSC, β_1, β_2 and β are monotone increasing functions (Lemma 2). I do not specify the players' payoffs, they are not needed in the explanation.

First we need to understand how the Robinson-Topkis (RT from now on) algorithm works. RT starts at the smallest strategy profile, $(1,1)$, and iterates the game's best-response function until two iterations are the same. Since $(1,1)$ is smaller than $\beta(1,1)$, and β is monotone, we have that $\beta(1,1)$ is smaller than $\beta(\beta(1,1)) = \beta^2(1,1)$. Similarly, $\beta^2(1,1)$ is smaller than $\beta^3(1,1)$, and so on—iterating β we get a monotone increasing sequence in S. Now, S is finite, so there must be an iteration k such that $\beta^k(1,1) = \beta^{k-1}(1,1)$. But then of course $\beta^k(1,1) = \beta(\beta^{k-1}(1,1))$, so $\underline{s} = \beta^{k-1}(1,1)$ is a Nash equilibrium.

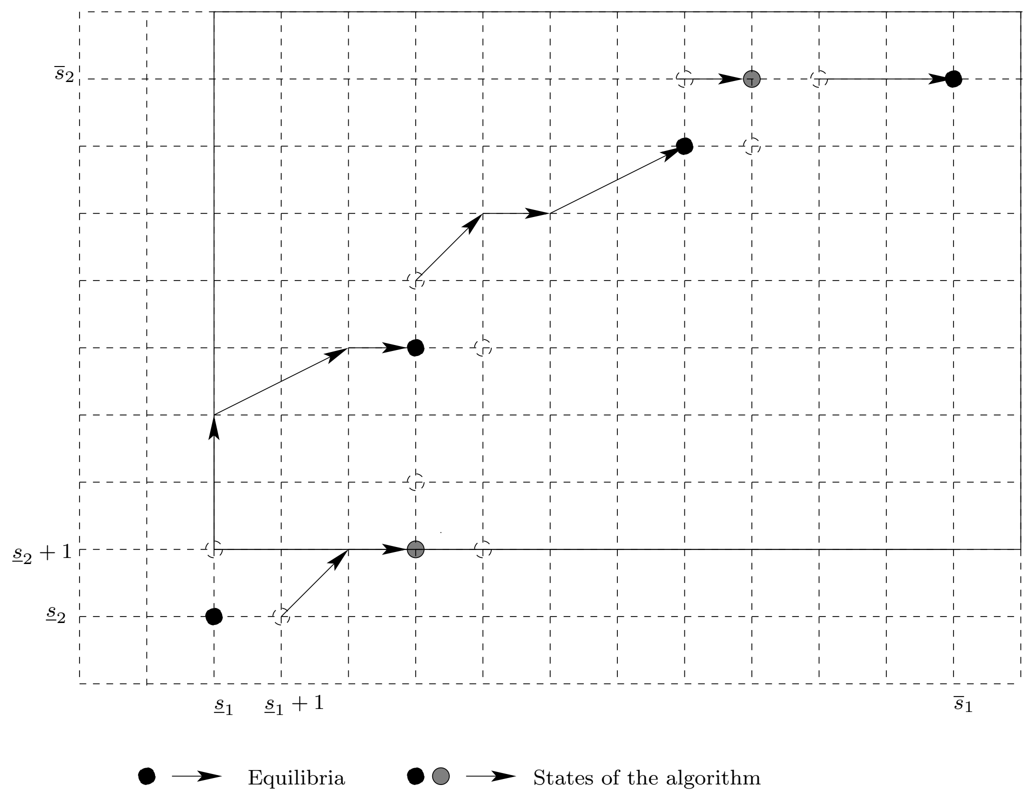

FIGURE 1. The algorithm in a two-player game.

It turns out that $\underline{s}$ is the *smallest* Nash equilibrium in Γ: Let s^* be any other equilibrium, and note that $(1,1) \leq s^*$. Monotonicity of β implies that $\beta(1,1) \leq \beta(s^*) = s^*$. Then, iterating β we get

$$\underline{s} = \beta^{k-1}(1,1) \leq \beta^{k-1}(s^*) = s^*.$$

In a similar way, RT finds the game's largest Nash equilibrium $\overline{s}$ by iterating the game's best-response function starting from the largest strategy profile, $(15,11)$.

I now describe informally the algorithm that I propose. Then I explain heuristically why it works. I develop these ideas in full generality in Section 4.

The algorithm consists of the following steps:

(1) Find the smallest ($\underline{s}$) and largest ($\overline{s}$) Nash equilibrium using RT—note $\underline{s}$ and $\overline{s}$ in Figure 1.
(2) Consider $\Gamma^r(\underline{s}_1, \underline{s}_2 + 1)$, the game where player 1 is restricted to choosing a strategy larger than $\underline{s}_1$, and player 2 is restricted to choosing a strategy larger than $\underline{s}_2 + 1$. The strategy profile $(\underline{s}_1, \underline{s}_2 + 1)$ is indicated in the figure with a circle $\bigcirc$ above $(\underline{s}_1, \underline{s}_2)$, and the strategy

space in $\Gamma^r(\underline{s}_1, \underline{s}_2+1)$ is the interval $[(\underline{s}_1, \underline{s}_2+1), (15, 11)]$ shown with non-dotted lines in the figure. Now use RT to find s^1, the smallest Nash equilibrium in $\Gamma^r(\underline{s}_1, \underline{s}_2+1)$. Each iteration of β is shown with an arrow in the figure, and s^1 is the black disk reached after three iterations.

Similarly, consider $\Gamma^r(\underline{s}_1+1, \underline{s}_2)$, the game where player 1 is restricted to choosing a strategy larger than $\underline{s}_1+1$, and player 2 is restricted to choosing a strategy larger than $\underline{s}_2$. The strategy profile $(\underline{s}_1+1, \underline{s}_2)$ is indicated in the figure with a circle $\bigcirc$ to the right of $(\underline{s}_1, \underline{s}_2)$. Use RT to find s^2, the smallest Nash equilibrium in $\Gamma^r(\underline{s}_1+1, \underline{s}_2)$; s^2 should be a black disk at this point, I explain in next step why it is gray.

(3) Check if s^1 and s^2 are Nash equilibria of Γ. First consider s^1. Because s^1 is an equilibrium of $\Gamma^r(\underline{s}_1, \underline{s}_2+1)$, and β is monotone increasing, we *only* need to check that $\underline{s}_2$ is not a profitable deviation for player 2. Similarly, to check if s^2 is an equilibrium we only need to check that $\underline{s}_1$ is not a profitable deviation for player 1. I explain below why these checks are sufficient. Let us assume that s^1 passes the check while s^2 fails, this is indicated in the figure by drawing s^2 as a gray circle.

(4) Do steps 2 and 3 for $\Gamma^r(s^1_1, s^1_2+1)$, $\Gamma^r(s^1_1+1, s^1_2)$, $\Gamma^r(s^2_1, s^2_2+1)$, and $\Gamma^r(s^2_1+1, s^2_2)$.

(5) Continue repeating steps 2 and 3 for each Nash equilibrium s^k found, unless s^k is equal to $\overline{s}$. The picture shows what the algorithm does for a selection of s^ks; note that the algorithm starts at larger and larger $\bigcirc$-circles, and that it approaches $\overline{s}$.

I phrased item 3—the "check"-phase—in terms of the first iteration of the algorithm. In general, let s^k be a candidate equilibrium obtained as the smallest equilibrium in some $\Gamma^r(\hat{s})$. To check if s^k is an equilibrium I need to take a confirmed (in Γ) equilibrium s^* with $s^* \leq \hat{s}$ and check that player i does not want to deviate to some strategy in the interval $[s^*_i, \hat{s}_i)$. In the first iteration $\hat{s} = (\underline{s}_1, \underline{s}_2+1)$ and $s^* = \underline{s}$, so I only need to check that player 2 does not want to deviate to $\underline{s}_2$.

Why is this check sufficient? First, $s^*_{-i} \leq s^k_{-i}$, and β is monotone increasing, so $s^*_i = \beta_i(s^*_{-i}) \leq \beta_i(s^k_{-i})$ and hence the best possible deviation—$\beta_i(s^k_{-i})$—is larger than s^*_i. Second, s^k is an equilibrium in $\Gamma^r(\hat{s})$, so no deviations larger than $\hat{s}_i$ are profitable. Thus we only need to check for deviations in the interval $[s^*_i, \hat{s}_i)$. As the iterations progress, we get larger and larger $\hat{s}$'s; but normally we also have larger and larger confirmed equilibria, and thus the intervals $[s^*_i, \hat{s}_i)$ often do not get very large.

I now explain why the algorithm finds all the Nash equilibria of Γ. Suppose that s is an equilibrium, so $\underline{s} \leq s \leq \overline{s}$. If $s = \underline{s}$ or $s = \overline{s}$, then the algorithm finds s in step 1. Suppose that $\underline{s} < s < \overline{s}$, then either $(\underline{s}_1, \underline{s}_2 + 1) \leq s$ or $(\underline{s}_1 + 1, \underline{s}_2) \leq s$ (or both). Suppose that $(\underline{s}_1, \underline{s}_2 + 1) \leq s$, so s is a strategy in $\Gamma^r(\underline{s}_1, \underline{s}_2 + 1)$. Note that s is also an equilibrium of $\Gamma^r(\underline{s}_1, \underline{s}_2 + 1)$: if a player i does not want to deviate from s when allowed to choose any strategy in S_i, she will not want to deviate when only allowed to choose the subset of strategies in $\Gamma^r(\underline{s}_1, \underline{s}_2 + 1)$. But s^1 is the smallest equilibrium in $\Gamma^r(\underline{s}_1, \underline{s}_2 + 1)$, so $s^1 \leq s$. If $s^1 = s$ the algorithm has found s. If $s^1 < s$ then either $(s^1_1, s^1_2 + 1) \leq s$ or $(s^1_1 + 1, s^1_2) \leq s$ (or both). Suppose that $(s^1_1, s^1_2 + 1) \leq s$, then repeating the steps above we will arrive at a new $s^k \leq s$. The sequence of strictly increasing s^ks only stops when s^k reaches $\overline{s}$, so $s < \overline{s}$ implies that there must be a $s^k = s$. Since s is an equilibrium, $s^k = s$ passes the test in item 3; hence the algorithm finds s.

4. The Algorithm

Let $e^{d_i}_l$ be the l-th unit vector in $\mathbf{R}^{d_i}$, i.e. $e^{d_i}_l = (0, \ldots 1, 0 \ldots 0) \in \mathbf{R}^{d_i}$, where 1 is the l-th element of $e^{d_i}_l$.

Algorithm 2. *Find $\underline{s} = \inf \mathcal{E}$ using $\underline{T}(\inf S)$, and $\overline{s} = \sup \mathcal{E}$ using $\overline{T}(\sup S)$. Let $\hat{\mathcal{E}} = \{\underline{s}, \overline{s}\}$. The set of possible states of the algorithm is the power set 2^S, the algorithm starts at state $\{\underline{s}\}$.*

Let the state of the algorithm be $\mathcal{M} \in 2^S$. While $\mathcal{M} \neq \{\overline{s}\}$, repeat the following sub-routine to obtain a new state $\mathcal{M}'$.

Subroutine *Let $\mathcal{M}' = \emptyset$. For each $s \in \mathcal{M}$, $i \in \{1, \ldots n\}$ and l with $1 \leq l \leq d_i$, if $(s_i + e^{d_i}_l, s_{-i}) \leq \overline{s}$, then do steps 1-4:*

1. *Let s^* be a maximal element in*
$$\left\{ \tilde{s} \in \hat{\mathcal{E}} : \tilde{s} \leq (s_i + e^{d_i}_l, s_{-i}) \right\}.$$
2. *Run $T^r(s_i + e^{d_i}_l, s_{-i})$; let $\hat{s}$ be the strategy profile at which it stops.*
3. *Check that no player j wants to deviate from $\hat{s}_j$ to a strategy in the set*
$$\left\{ z \in S_j : s^*_j \leq z \text{ and } (s_i + e^{d_i}_l, s_{-i})_j \nleq z \right\}.$$
If no player wants to deviate, add $\hat{s}$ to $\hat{\mathcal{E}}$.
4. *Add $\hat{s}$ to $\mathcal{M}'$ (Let $\mathcal{M}' = \mathcal{M}' \cup \{\hat{s}\}$).*

Theorem 7. *The set $\hat{\mathcal{E}}$ produced by Algorithm 2 coincides with the set $\mathcal{E}$ of Nash equilibria of Γ.*

Proof. First I shall prove that the algorithm stops after a finite number of iterations, and that it stops when $\mathcal{M} = \{\overline{s}\}$, not before (step "well-behaved"). Then I shall prove that $\hat{\mathcal{E}} \subseteq \mathcal{E}$, and then that $\mathcal{E} \subseteq \hat{\mathcal{E}}$.

Step "well-behaved." Let $M \subseteq 2^S$ be the collection of states visited by Algorithm 2. Let C be the set of maps $z : M \to S$ such that

(1) For all $\mathcal{M} \in M$, $z(\mathcal{M}) \in \mathcal{M}$;
(2) If the algorithm transits from $\mathcal{M}$ to $\mathcal{M}'$, and there is at least one player i and dimension l such that $(z(\mathcal{M})_i + e_l^{d_i}, z(\mathcal{M})_{-i}) \leq \overline{s}$, then $z(\mathcal{M}')$ is obtained from $T^r(z(\mathcal{M})_i + e_l^{d_i}, z(\mathcal{M})_{-i})$ from one such player and dimension in step 2 of Algorithm 2.

Note that, for all $\mathcal{M}$, $\mathcal{M} = \{z(\mathcal{M}) : z \in C\}$.

First I shall prove that the algorithm stops when it reaches state $\{\overline{s}\}$, and not before. I need to prove that $s \leq \overline{s}$ for all $s \in \cup\{\mathcal{M} : \mathcal{M} \in M\}$; which implies that $z(\mathcal{M}) \leq \overline{s}$ for all $z \in C$. Let the state $\mathcal{M}$ transit to state $\mathcal{M}'$. Let $s' \in \mathcal{M}'$, then s' must have been obtained from some $s \in \mathcal{M}$, and some i and l with $(s_i + e_l^{d_i}, s_{-i}) \leq \overline{s}$, by $T^r(s_i + e_l^{d_i}, s_{-i})$ in step 2 of the subroutine. By Lemma 5, s' is the smallest Nash equilibrium in $\Gamma^r(s_i + e_l^{d_i}, s_{-i})$. By Lemma 4, $\overline{s}$ is a Nash equilibrium of $\Gamma^r(s_i + e_l^{d_i}, s_{-i})$, so

$$s \leq (s_i + e_l^{d_i}, s_{-i}) \leq s' \leq \overline{s}.$$

This proves that $s \leq \overline{s}$ for all $s \in \mathcal{M}$, for all $\mathcal{M}$ that transit to some state, and that $s' \leq \overline{s}$ for all $s' \in \mathcal{M}'$ for all states $\mathcal{M}'$ that are obtained by transit from some other state. Unless $\underline{s} = \overline{s}$, these two possibilities cover all states in M, and if $\underline{s} = \overline{s}$ there is nothing to prove.

Now fix a state $\mathcal{M}$. For all $s \in \mathcal{M}$, $s \leq \overline{s}$, so if $\mathcal{M} \neq \{\overline{s}\}$ then there is $s \in \mathcal{M}$ such that $s < \overline{s}$. So there are i and l such that $(s_i + e_l^{d_i}, s_{-i}) \leq \overline{s}$. Thus the algorithm must transit from $\mathcal{M}$ to a new state while $\mathcal{M} \neq \{\overline{s}\}$.

Second, let $z \in C$. Let $\mathcal{M}$ be any state in M, and be $\mathcal{M}'$ be the state that it transits to. I shall prove that, if $z(\mathcal{M}') \neq \overline{s}$, then $z(\mathcal{M}) < z(\mathcal{M}')$. If $z(\mathcal{M}') \neq \overline{s}$ then, by item 2 of the definition of C, there is some i and l such that $z(\mathcal{M}')$ is obtained by $T^r(z(\mathcal{M})_i + e_l^{d_i}, z(\mathcal{M})_{-i})$ in Step 2 of the subroutine. So,

$$z(\mathcal{M}) < (z(\mathcal{M})_i + e_l^{d_i}, z(\mathcal{M})_{-i}) \leq z(\mathcal{M}').$$

Hence $z(\mathcal{M}) < z(\mathcal{M}')$.

Now, $M \subseteq 2^S$, and S is finite, so M and therefore C are finite sets. Each $z \in C$ is strictly increasing until $z(\mathcal{M}) = \overline{s}$, so the binary relation "$\mathcal{M}$ transits to $\mathcal{M}'$" on M is transitive. Thus, eventually $z(\mathcal{M}) = \overline{s}$ for every $z \in C$. But then there is an $\overline{\mathcal{M}} \in M$ such that $z(\overline{\mathcal{M}}) = \overline{s}$ for all $z \in C$, as C is finite. Hence

$$\overline{\mathcal{M}} = \cup\left\{z(\overline{\mathcal{M}}) : z \in C\right\} = \{\overline{s}\},$$

and Algorithm 1 stops at state $\overline{\mathcal{M}}$, after a finite number of steps.

Step $\hat{\mathcal{E}} \subseteq \mathcal{E}$. I shall prove that $\hat{\mathcal{E}} \subseteq \mathcal{E}$ by induction. First, in the initial state, $\{\underline{s}\}$, $\hat{\mathcal{E}} \subseteq \mathcal{E}$ by definition of $\hat{\mathcal{E}}$. Second, suppose that, when the algorithm is in state $\mathcal{M}$, $\hat{\mathcal{E}} \subseteq \mathcal{E}$, and that when the algorithm transits from state $\mathcal{M}$ to $\mathcal{M}'$ $\hat{s}$ is added to $\hat{\mathcal{E}}$. I shall prove that $\hat{s} \in \mathcal{E}$. By induction, this implies that $\hat{\mathcal{E}} \subseteq \mathcal{E}$.

Suppose we obtained $\hat{s}$ by running $T^r(s_i + e_l^{d_i}, s_{-i})$, for some $s \in \mathcal{M}$, and some player i and dimension l. Fix a player j. I shall prove that $\hat{s}_j \in \beta_{j,\Gamma}(\hat{s}_{-j})$ by first finding a strategy $z_j \in \beta_{j,\Gamma}(\hat{s}_{-j})$, and then showing that $u_j(z_j, \hat{s}_{-j}) \leq u_j(\hat{s}_j, \hat{s}_{-j})$.

Let s^* be the maximal element in

$$\left\{\tilde{s} \in \hat{\mathcal{E}} : \tilde{s} \leq (s_i + e_l^{d_i}, s_{-i})\right\}$$

found in step 2 of the algorithm. Note that $\underline{s} \leq (s_i + e_l^{d_i}, s_{-i})$, so the set of $\tilde{s} \in \hat{\mathcal{E}}$ such that $\tilde{s} \leq (s_i + e_l^{d_i}, s_{-i})$ is non-empty; thus s^* is well-defined. We have $s_j^* \in \beta_{j,\Gamma}(s_{-j}^*)$, as s^* is a Nash equilibrium. Let $\tilde{s_j} \in \beta_{j,\Gamma}(\hat{s}_{-j})$. Note that $s_{-j}^* \leq \hat{s}_{-j}$, so Milgrom and Shannon's (1994) Theorem 4 implies that $z_j = \tilde{s_j} \vee s_j^* \in \beta_{j,\Gamma}(\hat{s}_{-j})$.

By definition of z_j, $s_j^* \leq z_j$. First, if $(s_i + e_l^{d_i}, s_{-i})_j \leq z_j$, then $u_j(z_j, \hat{s}_{-j}) \leq u_j(\hat{s}_j, \hat{s}_{-j})$, as $\hat{s}_j \in \beta_{j,\Gamma^r(s_i+e_l^{d_i},s_{-i})}(\hat{s}_{-j})$ because $\hat{s}$ is a Nash equilibrium of $\Gamma^r(s_i + e_l^{d_i}, s_{-i})$. Second, if $(s_i + e_l^{d_i}, s_{-i})_j \nleq z_j$ then $u_j(z_j, \hat{s}_{-j}) \leq u_j(\hat{s}_j, \hat{s}_{-j})$ by step 4 of the subroutine. Hence $u_j(z_j, \hat{s}_{-j}) \leq u_j(\hat{s}_j, \hat{s}_{-j})$, so $z_j \in \beta_{j,\Gamma}(\hat{s}_{-j})$ implies that $\hat{s}_j \in \beta_{j,\Gamma}(\hat{s}_{-j})$. Player j was arbitrary, so $\hat{s} \in \beta_\Gamma(\hat{s})$ and $\hat{s} \in \mathcal{E}$.

Step $\mathcal{E} \subseteq \hat{\mathcal{E}}$. Let $s \in \mathcal{E}$. Suppose, by way of contradiction, that $s \notin \hat{\mathcal{E}}$.

Claim: Let Algorithm 2 transit from state $\mathcal{M}$ to state $\mathcal{M}'$. If there is $z \in \mathcal{M}$ with $z < s$ then there is $z' \in \mathcal{M}'$ with $z' < s$.

Proof of the claim: Since $z < s$, there is i and l such that $z_{il} < s_{il}$. Then s is a strategy profile in $\Gamma^r(z_i + e_l^{d_i}, z_{-i})$. If $\hat{s}$ is the strategy profile found by $T^r(z_i + e_l^{d_i}, z_{-i})$, then Lemma 5 implies that $\hat{s} \leq s$, as s is a Nash equilibrium of $\Gamma^r(z_i + e_l^{d_i}, z_{-i})$. If $\hat{s} = s$ then s would pass the test of step 4 and be added to $\hat{\mathcal{E}}$, but we assumed $s \notin \hat{\mathcal{E}}$ so it must be that $\hat{s} < s$. Set $z' = \hat{s}$, then $z' \in \mathcal{M}'$ by step 5, and the proof of the claim is complete.

Now, $s \notin \hat{\mathcal{E}}$ implies that $s \neq \underline{s}$. Initially $\mathcal{M} = \{\underline{s}\}$ so there is $z (= \underline{s})$ in $\mathcal{M}$ with $z < s$. Using the Claim above inductively, it must be that all stages of Algorithm 2 contain a z with $z < s$. But the final state of the algorithm is $\mathcal{M} = \{\overline{s}\}$; a contradiction, since $s \leq \overline{s}$. □

Remark 8. A modification of Algorithm 2 will make it run faster: Only do step 3 of the subroutine if there is no $s' \in \hat{\mathcal{E}}$ such that $\hat{s} \leq s'$, and $\hat{s} \in S(s_i + e_l^{d_i}, s_{-i})$, for the s, i and l at which s' was found. For, if there is such an s', then we know that $\hat{s} \notin \mathcal{E}$, as $\hat{s} \in \mathcal{E}$ would imply that $\hat{s}$ is an equilibrium of $\Gamma^r(s_i + e_l^{d_i}, s_{-i})$, which contradicts that s' is the smallest equilibrium of $\Gamma^r(s_i + e_l^{d_i}, s_{-i})$.

Theorem 7 says that Algorithm 2 works. In the rest of the paper I show that it is efficient.

5. Examples

I present two examples; they show how the algorithm works, and offer a first glance into when and how it is likely to be fast.

5.1. **Example 1.** Consider the two-player game on the left in Figure 2. Both players have identical strategy sets, $\{1, 2, 3, 4\}$. The strategies are ordered in the natural way: a strategy s_i is larger than strategy s_i' if it is a larger number, so 2 is larger than 1, 4 is larger than 2, and so on. With this order it is straightforward—if tedious—to check that Example 1 is a game with strategic complementarities.

	1	2	3	4
4	0, 3	2, 3	3, 4	5, 5
3	1, 3	3, 3	3, 4	4, 4
2	2, 3	4, 3	4, 4	4, 4
1	4, 4	3, 2	3, 1	3, 0

Example 1

	1	2	3	4
4	0, 0	0, 0	0, 0	0, 0
3	1, 3	1, 2	1, 1	0, 0
2	2, 3	2, 2	2, 1	0, 0
1	3, 3	3, 2	3, 1	0, 0

Example 2

Figure 2. Two examples.

Algorithm 2 starts by finding $\underline{s} = \inf \mathcal{E}$ and $\overline{s} = \sup \mathcal{E}$ by RT: let us first iterate the game's best response function starting at the smallest point in the strategy space, $(1, 1)$. Now, $\beta(1, 1) = (1, 1)$ so $(1, 1) = \inf \beta(1, 1)$, and the RT algorithm returns $\underline{s} = (1, 1)$ as the game's smallest equilibrium. Similarly, it returns $\overline{s} = (4, 4)$ as the game's largest equilibrium. Then, the initial state of the algorithm is $\mathcal{M} = \{(1, 1)\}$, and the initial list of equilibria is $\hat{\mathcal{E}} = \{(1, 1), (4, 4)\}$, see Table 1.

	$\mathcal{M}$	$\hat{\mathcal{E}}$
1	$\{(1, 1)\}$	$\{(1, 1), (4, 4)\}$
2	$\{(2, 3)\}$	$\{(1, 1), (2, 3), (4, 4)\}$
3	$\{(3, 3), (4, 4)\}$	$\{(1, 1), (2, 3), (4, 4)\}$
4	$\{(4, 4)\}$	$\{(1, 1), (2, 3), (4, 4)\}$

Table 1. Iterations in Example 1

The initial state is $\mathcal{M} = \{(1,1)\}$. First, $(1,1)+(1,0) = (2,1) \leq \overline{s}$, so we do steps 1-4 in the subroutine starting at $(2,1)$. $\inf \beta_{\Gamma^r(2,1)}(2,1) = (2,3)$, and $\inf \beta_{\Gamma^r(2,1)}(2,3) = (2,3)$, so RT in game $\Gamma^r(2,1)$ returns $(2,3)$ as the smallest equilibrium in $\Gamma^r(2,1)$. In step 3 we need to check that player 1 does not want to deviate to play strategy 1, but playing strategy 1 would yield her a payoff of 3, while playing strategy 2 yields her a payoff of 4. Since the deviation is not profitable, we add $(2,3)$ to $\hat{\mathcal{E}}$. Second, $(1,1)+(0,1) = (1,2) \leq \overline{s}$, so we do steps 1-4 starting at $(1,2)$: $\inf \beta_{\Gamma^r(1,2)}(1,2) = (2,2)$, $\inf \beta_{\Gamma^r(1,2)}(2,2) = (2,3)$, and $\inf \beta_{\Gamma^r(1,2)}(2,3) = (2,3)$. Graphically, the action of $\inf \beta_{\Gamma^r(1,2)}$ is

$$(1,2) \to (2,2) \to (2,3) \to (2,3).$$

Thus RT returns $(2,3)$ as the smallest equilibrium in $\Gamma^r(1,2)$. To sum up, the result of steps 1-4 is that the algorithm transits to state $\{(2,3)\}$, and the list of equilibria is $\hat{\mathcal{E}} = \{(1,1),(2,3),(4,4)\}$.

Now the state of the algorithm is $\{(2,3)\}$. First, $(2,3)+(1,0) = (3,3) \leq \overline{s}$, so we do steps 1-4 in the subroutine starting at $(3,3)$. Note that $\inf \beta_{\Gamma^r(3,3)}(3,3) = (3,3)$, so RT returns $(3,3)$ as the smallest equilibrium in $\Gamma^r(3,3)$. In step 3 we need to check that player 1 does not want to deviate from strategy 3 to strategy 2. In fact, strategy 2 gives a higher payoff (4) than strategy 3 (3), so $(3,3)$ is not an equilibrium, and we do not add $(3,3)$ to $\hat{\mathcal{E}}$. Second, $(2,3)+(0,1) = (2,4) \leq \overline{s}$, so we do steps 1-4 in the subroutine starting at $(2,4)$. Note that $\inf \beta_{\Gamma^r(2,4)}(2,4) = (4,4)$, so RT returns $(4,4)$ as the smallest equilibrium in $\Gamma^r(4,4)$. We already know that $(4,4)$ is an equilibrium of Γ. The result of steps 1-4 is that the algorithm transits to state $\{(3,3),(4,4)\}$, and the list of equilibria is $\hat{\mathcal{E}} = \{(1,1),(2,3),(4,4)\}$.

The state of the algorithm is now $\{(3,3),(4,4)\}$. Both $(4,4)+(1,0)$ and $(4,4)+(0,1)$ fail to be smaller than $\overline{s}$, so we do not run the subroutine starting from $(4,4)$. Now, steps 1-4 in the subroutine starting from $(3,3)+(1,0) = (4,3)$ or $(3,3)+(0,1) = (3,4)$ give $(4,4)$ as the smallest equilibrium of $\Gamma^r(3,4)$ and $\Gamma^r(4,3)$. So, the final state of the algorithm is $\{(4,4)\}$, and the final list of equilibria is $\{(1,1),(2,3),(4,4)\}$.

5.2. **Example 2.** Now consider the game on the right in Figure 2. RT yields $(1,1)$ as the smallest equilibrium, and $(4,4)$ as the largest equilibrium in Example 2. The initial state of the algorithm is thus $\{(1,1)\}$. We start the subroutine at $(2,1) = (1,1)+(1,0)$ and get back $(2,1)$ as the smallest equilibrium of $\Gamma^r(2,1)$. But player 1 prefers strategy 1 over strategy 2, so $(2,1)$ does not survive step 3. We start the subroutine at $(1,2) = (1,1)+(0,1)$ and get back $(1,2)$ as the smallest equilibrium of $\Gamma^r(1,2)$. But player 1 prefers strategy 1 over strategy 2, so $(1,2)$ does not survive step 3.

If one completes all iterations (shown in Table 2) it is clear that the algorithm stops at all strategy profiles, and discards all but the largest and the smallest equilibria of the game.

	$\mathcal{M}$	$\hat{\mathcal{E}}$
1	$\{(1,1)\}$	$\{(1,1),(4,4)\}$
2	$\{(2,1),(1,2)\}$	$\{(1,1),(4,4)\}$
3	$\{(3,1),(2,2),(1,3)\}$	$\{(1,1),(4,4)\}$
4	$\{(4,1),(3,2),(2,3),(1,4)\}$	$\{(1,1),(4,4)\}$
5	$\{(4,2),(3,3),(2,4)\}$	$\{(1,1),(4,4)\}$
6	$\{(4,3),(3,4)\}$	$\{(1,1),(4,4)\}$
7	$\{(4,4)\}$	$\{(1,1),(4,4)\}$

Table 2. Iterations in Example 2

Example 2 presents a pathological situation; the algorithm is forced to check all strategy profiles of the game. The root of the problem is that, after each iteration, it is optimal for the players to choose their smallest allowed strategies. I argue in Section 6 that the situation will typically not occur in the games that one encounters in applications.

6. How fast is Algorithm 2?

6.1. **Outline.** The rest of the paper establishes that Algorithm 2 is generally very fast. Here is an outline:

- Section 7. I simulate a large class of games, and show that Algorithm 2 finds all equilibria very quickly. The class of games was chosen trying to bias the test against Algorithm 2.
- Section 8. I show that Algorithm 2 is faster when best-responses in each iteration have large increases—I then argue that this will occur for many natural applications of the algorithm.
- Section 9. I present a version of the algorithm that applies to two-player games with strict preferences. This version is faster than Algorithm 2.

In the worst case, Algorithm 2 is slow (see Example 2 of Section 5). But the worst case is—not surprisingly—irrelevant for actual applications.

6.2. **The Benchmark.** I now describe the trivial algorithm, the benchmark against which Algorithm 2 is compared in terms of speed.

Let $\Gamma = \{(S_i, u_i) : i = 1, \dots n\}$ be an n-player game. Fix a player, say i. First, for each s_{-i}, find the set of best-responses by i. Second, check if any player $j \neq i$ wants to deviate from her strategy in s_{-i} when i chooses one of her best-responses. This algorithm is essentially the “underlining” method for finding the Nash equilibria that one teaches first-year students.

Suppose that all players have K strategies, and that best-responses are everywhere unique. Let r be the time required to make a payoff-function evaluation. Note that r is independent of K. The trivial algorithm turns out to require $O(rK^n)$ time: The algorithm performs K payoff-function evaluations to find the best-responses, K^{n-1} times. Thus the algorithm performs K^n payoff-function evaluations. By the accounting procedure in Aho, Hopcroft, and Ullman (1974), the algorithm is $O(rK^n)$.

If best-responses are not unique, the algorithm needs to check if any player $j \neq i$ wants to deviate from her strategy in s_{-i}, for all of i's best-responses. In the worst case, the set of best responses grows at rate K, so the algorithm is $O(rK^{n+1})$.

Note that the $O(rK^n)$ calculation is not an unrealistic worst-case bound. It is the time the trivial algorithm *must* use, as long as best-responses are unique.

For n-player games with $n > 2$, Bernhard von Stengel (personal communication) has suggested a recursive procedure that improves on the trivial algorithm: for each $s_n \in S_n$, fix s_n as the strategy played by player n, and find all equilibria of the resulting $(n-1)$-player game. For each equilibrium found, check if player n wishes to deviate from s_n. In the worst-case calculation, this recursive procedure does not improve on the simple trivial algorithm—but in many games of interest it may speed up the trivial algorithm by saving on calculations of best-response by player n. If Γ is a GSC, for any $s_n \in S_n$, the resulting $(n-1)$-player game is also a GSC. So von Stengel's suggestion can be applied to Algorithm 2 as well. But I do not know if following his suggestion improves over the speed of Algorithm 2 or not.

7. Performance

I evaluate the performance of Algorithm 2, using a class of two-player games, where each player has the interval $[0, 1]$ as her strategy space. The algorithm is fast; I use Algorithm 2 with different discretizations—grids—of $[0, 1]$, and show that, even when the resulting grid is quite small (the number of strategies of each player is quite large), the algorithm is very fast. I use the computations to compare Algorithm 2 to the trivial algorithm.

7.1. **Class of games.**

I use a class of games that tend to have a large number of equilibria—Algorithm 2 is faster the smaller is the number of equilibria, and I want to evaluate Algorithm 2 using games where it does not have

One game			2,000 games		
Strat.	Trivial	Alg. 2	Strat.	Trivial	Alg. 2
20,000	2.8 min	2.96 sec.	20,000	3.8 days	1.6 hours
40,000	12.1 min	10.0 sec.	40,000	16.8 days	5.5 hours
60,000	26.1 min	10.8 sec.	60,000	36.0 days	6.0 hours

TABLE 3. Simulations.

an apriori advantage. The class of games is idiosyncratic, but that is unavoidable: the functional forms that economists normally use give few—or unique—equilibria. With those functional forms, Algorithm 2 is faster still.

The games have two-players, each player i has strategy set $S_i = [0,1]$, and payoff function

$$\begin{aligned} u_i(s_i, s_{-i}) &= -(\alpha_i/10)(s_i - s_{-i})^2 + 200\beta_i \sin(100 s_i) \\ &+ (1/100)\left[(1-\alpha_i)s_i(1+s_{-i}) - (1/2-\beta_i)s_i^2/100\right]. \end{aligned}$$

The parameters α_i and β_i are in $[0,1]$.

I arrived at the above functional form by trying to come up with games that have a fairly large number of equilibria. The first summand is a "pure-coordination term," its role is to produce multiple equilibria. The role of the second summand is to provoke multiple maxima (so that preferences are not strict, see Section 9); the second summand also helps in getting multiple equilibria. [3] The third and fourth summand are variants of polynomial terms that I found—by trial and error—often produce multiple equilibria.

Note that, for all α_i and β_i, $(\{1,2\}, \{S_1, S_2\}, \{u_1, u_2\})$ is a GSC.

I discretized the players' strategy spaces, so each player i chooses a strategy in $S_i = \{k/K : 0 \leq k \leq K\}$. Parameters α_i and β_i are chosen at pseudo-random from $[0,1]$ using a uniform distribution.

7.2. **Results.** The results of the simulations are in Table 3. I first compare the performance of Algorithm 2 and the trivial algorithm. Then I discuss what the simulations say about Algorithm 2 in general.

I simulated a large number of games, and used the algorithms to find the equilibria of each game. In each individual game, the parameters α_i and β_i were generated at pseudo-random from a uniform distribution on $[0,1]$. The average results are in the table on the left.

On average, when each player has 20,000 strategies, Algorithm 2 needed 2.96 seconds to find all equilibria. The trivial algorithm needed 2.8 minutes to do the same work. The table reports the results for 40,000 and 60,000 as well.

[3]Simulations without the second summand are slightly faster, and the games have fewer equilibria.

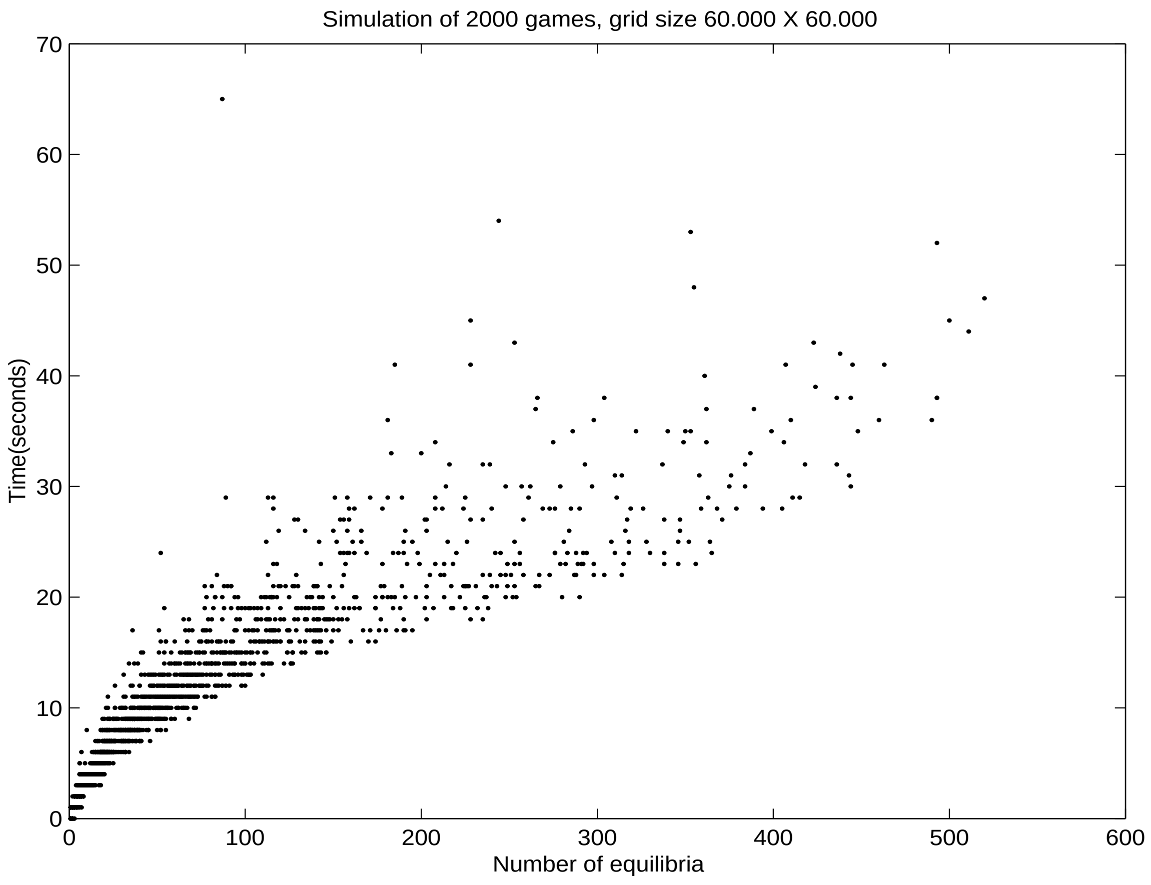

FIGURE 3. Relation between time and number of equilibria.

The table on the right reports how much each algorithm needs to find all the equilibria of 2,000 games—presumably a representative task of the estimation or calibration of an economic model. Already with 20,000 strategies the trivial algorithm needs 3.8 days. Algorithm 2, on the other hand, needs only 1.6 hours.

On the whole, Algorithm 2 is remarkably fast. It finds all equilibria of a game with 3.6×10^9 strategy-profiles in only 10 seconds, and it finds the equilibria of 2,000 such games in 6 hours. The graph in Figure 3 shows that nothing is hidden in the averages; the graph plots all 2,000 games. It shows how many equilibria each game has, and how long it takes for the algorithm to find them. Note that, with one exception, Algorithm 2 never needs more than one minute to find all equilibria.

Table 4 compares the performance of the two algorithms. With 20,000 strategies, Algorithm 2 is 56 times faster than the Trivial algorithm. The

Strat.	Trivial/Alg. 2
20,000	56
40,000	66
60,000	146

Table 4. Comparison of Trivial and Algorithm 2

relative performance of Algorithm 2 improves with the number of strategies, because the performance of Algorithm 2 is sensitive to the number of equilibria more than to the number of strategies.

7.3. **Implementation.** I wrote an implementation in C. The code (and the output from the simulations reported above) can be downloaded from `http://www.hss.caltech.edu/~fede`. The difficulty in implementing Algorithm 2 is that the state, $\mathcal{M}$, of the algorithm is potentially taken from a large set of possible states. Reserving space for the possible values that $\mathcal{M}$ can take may slow down the algorithm considerably. I found a rudimentary solution in my implementation of the algorithm; hopefully a better programmer can write a more efficient implementation.

The average computations on the left in Table 3 are based on 100 simulations in the case of the trivial algorithm, and 2,000 simulations in the case of Algorithm 2. Obviously, the trivial algorithm needs the same amount of time to compute the equilibria of each game. There is a very slight difference in the actual computation of some games, probably due to how the computer organizes the task, or because some payoff functions require slightly more time than others. But all 100 games with 20,000 strategies required essentially 2.8 minutes (the output can be downloaded from the web-page indicated above). The calculations for 2,000 games in the case of the trivial algorithm are thus a "projection."

All the simulations were done on a Linux Dell Precision PC with a 1.8 GHz Xeon CPU and 512 MB Ram. The computer performs 252.2 floating-point operations per second (based on output MFLOPS(1), from Al Aburto's flops.c program).

8. Calculating best-responses

8.1. **The source of slowness.** The examples is Section 5 show that Algorithm 2 can be slow. I shall argue that the examples are in some sense pathological, and that the algorithm is likely to be very fast in most situations—in particular when the source of the game is the discretization of a game with continuous strategy-spaces.

The algorithm is slow when best-response calculations "advance slowly". For example consider a game with two players, each with $K = 10$ strategies:

the numbers $31, 32, \dots 40$. If player 1's best response to 2 playing 31 is 31, then computing 1's best-response to 2 playing 32 requires 10 computations; it requires computing the payoffs from *all* strategies, and finding the maximum. But if 1's best response to 2 playing 31 is 39, then computing 1's best response to 32 requires only 2 computations—best responses are monotone increasing, so it is enough to compare the payoffs from playing 39 and 40.

Hence, if best-responses advance slowly relative to K—as K grows—then Algorithm 2 is relatively slow. On the other hand, if best-responses advance at rate K or faster, the algorithm will be fast. In fact, I shall establish that the algorithm will be linear in the worst case.

When will best-responses advance quickly? Consider an n-player game, and suppose that player i has strategy-space $[0, 1]$. We can apply Algorithm 2 by first discretizing i's strategy-space to $\{k/K : k = 0, \dots K\}$.

Figure 4 illustrates the situation. On the top part of the figure is player i's payoff function $s_i \mapsto u_i(s_i, s_{-i})$, holding opponents' strategies fixed at s_{-i}. In the middle of the figure is i's best-response to s_{-i}, when i is restricted to choosing s_i or a larger strategy—her strategy-space in $\Gamma^r(s_i, s_{-i})$. We select the smallest best response when there is more than one. For strategies $s_i \in [0, s_i^1]$, s_i^1 is the best-response. For strategies $s_i \in (s_i^1, s_i^2)$, it is a best-response to choose s_i; the solution is at a corner. For $s_i \in [s_i^2, s_i^3]$, on the other hand, i's best response is to choose s_i^3. Finally, for $s_i \in (s_i^3, 1]$, it is again optimal for i to be at a corner, and s_i is the unique best-response in the restricted game. The bottom of Figure 4 shows $\beta_{i,\Gamma^r(s_i,s_{-i})}(s_i, s_{-i}) - s_i$.

I now show that best-responses advance quickly over sets of s_i such that $\beta_{i,\Gamma^r(s_i,s_{-i})}(s_i, s_{-i}) - s_i > 0$ (indicated as "Fast" regions in Figure 4), and slowly over sets of s_i such that $\beta_{i,\Gamma^r(s_i,s_{-i})}(s_i, s_{-i}) - s_i = 0$ (indicated as "Slow" regions in Figure 4).

Let $k^*(K)/K$ be i's smallest best-response to s_{-i} in the discretized version of the game—i.e. the game where i has strategy-space $\{k/K : k = 0, \dots K\}$. Assume that i's payoff function is continuous, then the maximum theorem ensures that

$$\inf \beta_{i,\Gamma^r(s_i,s_{-i})}(s_i, s_{-i}) \leq \liminf_{K\to K} k^*(K)/K,$$

as the set of maximizers parameterized by K is upper hemicontinuous.

First, let s_i be such that $\inf \beta_{i,\Gamma^r(s_i,s_{-i})}(s_i, s_{-i}) - s_i > 0$. Then $k^*(K) - s_i$ grows at rate at least K. So the number of strategies that we eliminate from future best-response calculations, $k^*(K)$, grows at rate at least K, and thus best-responses advance quickly. In fact (see 8.2) it will in the worst case advance linearly.

Second, let s_i be such that $\inf \beta_{i,\Gamma^r(s_i,s_{-i})}(s_i, s_{-i}) - s_i = 0$. As K grows, $k^*(K)/K$ will approach s_i, so the number of strategies we eliminate from future calculations will be, in the limit, zero. In practice, the algorithm will

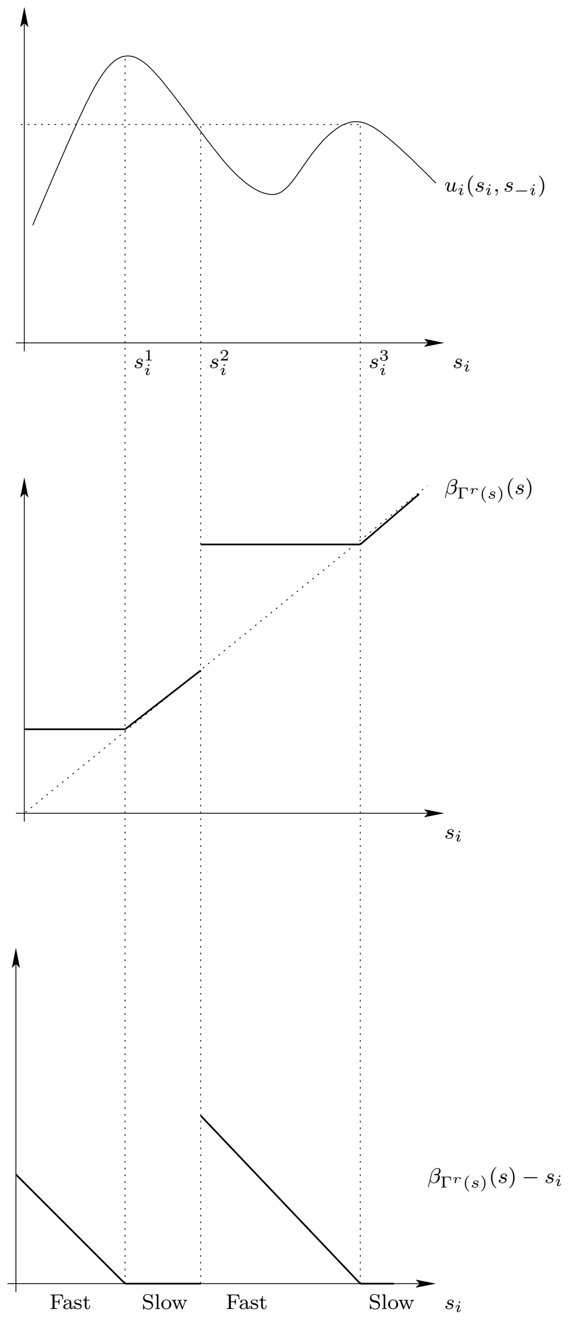

FIGURE 4. Slow and fast regions.

.

advance one-step-at-a-time, until an increase in some other player's strategy pulls the algorithm away from the region.

What do we learn about Algorithm 2? Well, it will advance quickly over fast regions, and require more time in slow regions. The result depends on the game: it depends on the size of the regions, on how much time it spends in each region, and on whether the algorithm will actually end up in these regions. For example, the largest equilibrium often implies that the algorithm does not need to search over the slow regions on the upper side of players' strategy spaces.

Normally, fast regions are important enough, and the algorithm is fast enough over these regions (faster than linear), that the algorithm finds all equilibria very quickly.

8.2. **Bounding how much best-responses change with K.** The argument in Section 8.1 says that slowness depends on how slowly best responses advance. But it is too strong to require that best responses advance quickly globally—normally there are some strategy-profiles at which best responses advance slowly. One possible solution is to control the size of the regions in strategy space where best responses must advance slowly. I develop a different solution below. I consider a family of games, and assume that, at each s in strategy space, the average best-response over the family of games does not advance slowly. It follows that Algorithm 2 is on average linear.

Let $(\Omega, \mathcal{F}, P)$ be a probability space. Let $(\Gamma_\omega, \omega \in \Omega)$ be a family of GSC such that Γ_ω has n players, for all $\omega \in \Omega$, and each player i has strategy-space $\{1, 2, \ldots K\}$. [4]

Assume that there is $\gamma \in (0,1)$ such that, for any strategy-profile s,

$$\gamma(K - s_i) \leq E \inf \beta_{i,\Gamma^r_\omega(s)}(s) - s_i.$$

Note that S_i is finite, so $E \inf \beta_{i,\Gamma^r_\omega(s)}(s)$ is well-defined.

Let $T(\omega)$ be the time required by Algorithm 2 to find all candidate equilibria. I use the accounting procedure in (Aho, Hopcroft, and Ullman, 1974, p. 35–38) to calculate $T(\omega)$, which here amounts to counting the number of payoff-function evaluations that the algorithm performs. Strategy spaces are fixed, so $T(\omega)$ has finite range, and thus $E(T)$ is well-defined.

Proposition 9. *$E(T)$ is $O(nK/\gamma^2)$.*

Proof. Let L be the number of iterations in Algorithm 2. Each iteration l, $l = 1, 2, \ldots L$, has I_l iterations in the RT-algorithm. Each iteration h, $h = 1, \ldots I_l$, implies a best-response calculation for each player i. Fix one player. Let x^h_l be the strategy that is a best response in iteration l of Algorithm 2 and iteration h of the corresponding RT algorithm. Let K_l be the number of strategies we need to consider for that player in iteration l; note that

[4]I restrict the players' strategies to save on notation. One can easily extend the argument to more general strategy spaces.

$K_l = K - x_{l-1}^{I_{l-1}}$. If the calculated best response in iteration $h-1$ was x_l^{h-1}, the best-response calculation is done in $K_l - x_l^{h-1}$ time. We thus need to bound

$$\sum_{l=1}^{L}\sum_{h=0}^{I_l}(K_l - x_l^h).$$

Consider the l-th iteration of the RT algorithm, where each player has K_l strategies. I shall prove that

$$E\sum_{h=0}^{I_l-1}(K_l - x_l^h) \leq EK_l/\gamma$$

by induction. First,

$$\begin{aligned} E\textstyle\sum_{h=0}^{I_l-1}(K_l - x_l^h) &= E\left\{\textstyle\sum_{h=0}^{I_l-2}(K_l - x_l^h) + E\left[(K_l - x^{I_l-1})|x^{I_l-2}\right]\right\} \\ &\leq E\left\{\textstyle\sum_{h=0}^{I_l-2}(K_l - x_l^h) + (1-\gamma)(K_l - x^{I_l-2})\right\}, \end{aligned}$$

as $E\left[(K_l - x^{I_l-1})|x^{I_l-2}\right] \leq (1-\gamma)(K_l - x^{I_l-2})$. Second, let

$$E\sum_{h=0}^{I_l-1}(K_l - x_l^h) \leq E\left\{\sum_{h=0}^{(I_l-1)-(m+1)}(K_l - x_l^h) + \sum_{h=0}^{m}(1-\gamma)^h(K_l - x_l^{I_l-m})\right\}.$$

Then

$$E\left\{\textstyle\sum_{h=0}^{(I_l-1)-(m+1)}(K_l - x_l^h) + \sum_{h=0}^{m}(1-\gamma)^h(K_l - x_l^{I_l-m})\right\}$$
$$=$$
$$E\left\{\textstyle\sum_{h=0}^{(I_l-1)-(m+1)}(K_l - x_l^h) + \sum_{h=0}^{m}(1-\gamma)^h E\left[(K_l - x_l^{I_l-m})|x_l^{I_l-(m+1)}\right]\right\}$$
$$\leq$$
$$E\left\{\textstyle\sum_{h=0}^{(I_l-1)-(m+1)}(K_l - x_l^h) + \sum_{h=0}^{m}(1-\gamma)^h(K_l - x_l^{I_l-m})\right\}.$$

Induction on m now proves

$$E\sum_{h=0}^{I_l-1}(K_l - x_l^h) \leq EK_l\sum_{h=0}^{I_l-1}(1-\gamma)^h = EK_l\left[1-(1-\gamma)^{I_l-1}\right]/g \leq K_l/\gamma$$

Now, by a similar calculation

$$E\sum_{l=1}^{L}K_l/\gamma \leq E\sum_{l=1}^{L}(1-\gamma)^{\sum_{j=0}^{l-1}I_j}K/\gamma \leq E\sum_{l=1}^{L}(1-\gamma)^{l\overline{I}}K/\gamma,$$

where $\overline{I} = \max\{I_l : l = 1, \dots L\}$. Then

$$E\sum_{l=1}^{L} K_l/\gamma \le E\frac{1-(1-\gamma)^{\overline{I}(L+1)}}{1-(1-\gamma)}K/\gamma \le K/\gamma^2.$$

But there are n players, so the expected time used by the algorithm is bounded by

$$nK/\gamma^2.$$

□

9. Two-player games with strict preferences

Let Γ be a two-player game where players have strict preferences and $d_1 = d_2 = 1$. I present a simple version of Algorithm 2 that finds all the equilibria of Γ. I can bound the complexity of this simple version of Algorithm 2.

Algorithm 3. *Find $\underline{s} = \inf\mathcal{E}$ using $\underline{T}(\inf S)$, and $\overline{s} = \sup\mathcal{E}$ using $\overline{T}(\sup S)$. Let $\hat{\mathcal{E}} = \{\underline{s}, \overline{s}\}$. The set of possible states of the algorithm is S, the algorithm starts at state $\underline{s}$.*

Let the state of the algorithm be $m \in S$. While $m \neq \overline{s}$, repeat the following sub-routine to obtain a new state m'.

Subroutine *If $m + (1,1) \le \overline{s}$, then do steps 1-4:*

(1) *Let s^* be a maximal element in*
$$\left\{\tilde{s} \in \hat{\mathcal{E}} : \tilde{s} \le m + (1,1)\right\}.$$
(2) *Run $T^r(m+(1,1))$; let $\hat{s}$ be the strategy profile at which it stops.*
(3) *Check that no player j wants to deviate from $\hat{s}_j$ to a strategy in the interval $\left[s^*_j, (m+(1,1))_j\right]$. If no player wants to deviate, add $\hat{s}$ to $\hat{\mathcal{E}}$.*
(4) *Let $m' = \hat{s}$.*

Say that Algorithm 3 makes an *iteration* each time it does steps 1-4. Say that Algorithm 3 makes a *payoff-function evaluation* each time it calculates u_1 or u_2. Let r be the time required to make a payoff-function evaluation. Note that r is independent of K_1 and K_2. Let $\underline{K} = \min\{K_1, K_2\}$ and $\overline{K} = \max\{K_1, K_2\}$.

Theorem 10. *Algorithm 3 finds all Nash equilibria in $O(r\overline{K}^2)$ time, and does at most $\underline{K}$ iterations.*

Proof. First I prove that Algorithm 3 is well-behaved and stops when it says that it stops. Let $M \subseteq S$ be the set of states visited by Algorithm 3. Note that, for all $m \in M$, $m + (1,1) \le m'$, for all m' obtained at a later iteration of the subroutine. Further, at each iteration of the subroutine there is a unique m found. So, if $m, m' \in M$ then either m' is found after m and

$m+(1,1) \leq m'$, or m is found after m' and $m'+(1,1) \leq m$. Then, M is totally ordered, and for any $m, m' \in M$, if $m \neq m'$ then either $m+(1,1) \leq m'$ or $m'+(1,1) \leq m$. I shall prove below that $m \neq \overline{s}$ implies that $m+(1,1) \leq \overline{s}$; so if $m \neq \overline{s}$ then the algorithm does not stop at m. Since M is finite, the algorithm stops in a finite number of steps, and it stops when the state is $\overline{s}$.

I need to prove the following

Claim. If $s \in \mathcal{E}(\Gamma^r(\tilde{s}))$, then either $s = \inf \mathcal{E}(\Gamma^r(\tilde{s}))$, or

$$\inf \mathcal{E}(\Gamma^r(\tilde{s})) + (1,1) \leq s.$$

Proof of the claim. Suppose that $s \in \mathcal{E}(\Gamma^r(\tilde{s}))$, and that $s \neq \hat{s} = \inf \mathcal{E}(\Gamma^r(\tilde{s}))$. By Lemma 4, $\hat{s} \leq s$. Suppose—without loss of generality—that $s_1 \neq \hat{s}_1$. Now, $\hat{s}_1 \in \beta_{\Gamma^r(\tilde{s})}(\hat{s}_2)$ so $s_2 = \hat{s}_2$ would imply that $\beta_{\Gamma^r(\tilde{s})}(\hat{s}_2)$ has at least two different elements, s_1 and $\hat{s}_1$. Impossible since players in Γ have strict preferences. It must be then that $s_2 \neq \hat{s}_2$. But then $\hat{s} \leq s$ implies $\hat{s}_1 < s_1$ and $\hat{s}_2 < s_2$, so $\hat{s}+(1,1) \leq s$. This proves the claim.

I now prove that $M \ni m \neq \overline{s}$ implies that $m+(1,1) \leq \overline{s}$: let $m' \in M \cup \{\inf S - (1,1)\}$ be the state from which m was obtained by $T^r(m'+(1,1))$. There must be such an m' by definition of m: either m is found in step 2 of the algorithm, or $m = \underline{s}$, and thus m was found by $\underline{T}(\inf S) = T^r(\inf S)$. Now, $m = \inf \mathcal{E}(\Gamma^r(m'+(1,1)))$ and $\overline{s} \in \mathcal{E}(\Gamma^r(m'+(1,1)))$, so the claim and $m \neq \overline{s}$ implies that $m+(1,1) \leq \overline{s}$.

Second, I prove that $\hat{\mathcal{E}} = \mathcal{E}$. The proof that $\hat{\mathcal{E}} \subseteq \mathcal{E}$ is very similar to the proof that $\hat{\mathcal{E}} \subseteq \mathcal{E}$ in Theorem 7, so I omit it. I shall prove that $\mathcal{E} \subseteq \hat{\mathcal{E}}$. Let $s \in \mathcal{E}$ and suppose, by way of contradiction, that $s \notin \hat{\mathcal{E}}$. Let m be some state of the algorithm such that $m \leq s$, we must have $m \neq s$ or s would be added to $\hat{\mathcal{E}}$, since $s \in \mathcal{E}$ implies that s passes the test in step 3. The claim implies that $m'+(1,1) \leq s$, as $m' \neq s$ for the same reason that $m \neq s$.

Now induct on M: $M \ni \underline{s} \leq s$, and if, at some state m, $m \leq s$, then $m'+(1,1) \leq s$ for the state m' that the state transits to. By induction, we must have $\overline{s}+(1,1) \leq s$. A contradiction, as $s \in \mathcal{E}$ implies that $s \leq \overline{s}$.

Now I shall prove that the algorithm needs less than $\underline{K}$ iterations. First, each iteration of Algorithm 3 produces one and only one element of M, so there are no more iterations than there are elements in M. Second, $M \subseteq \{1, \ldots K_1\} \times \{1, \ldots K_2\}$, and for each $m, m' \in M$, $m \neq m'$ then either $m+(1,1) \leq m'$ or $m'+(1,1) \leq m$. Thus M cannot have more elements than either $\{1, \ldots K_1\}$ or $\{1, \ldots K_2\}$. Thus, M has not more than $\min\{K_1, K_2\} = \underline{K}$ elements.

Now I shall prove that Algorithm 3 needs no more than $2\overline{K}^2$ payoff-function evaluations. If $K_1 \neq K_2$, let us change the game: add strictly dominated strategies to the player with the smallest K_i until that player has as many strategies as the other player. The set of equilibria do not change, and besides searching unnecessarily over dominated strategies, neither does the behavior of the algorithm

Let $K = K_1 = K_2$. The worst-case calculation calls for maximizing the number of iterations of Algorithm 3, even if it means fewer iterations of RT; this is because each iteration of Algorithm 3 requires one call to the RT algorithm.

Let $\underline{s} = (1, 1)$ and $\overline{s} = (K, K)$. Suppose that, at each state $m \in M$, the call to $T^r(m + (1, 1))$ in step 2 of the algorithm returns $m + (1, 1)$ as the smallest equilibrium in $\Gamma^r(m+(1, 1))$. This gives $M = \{(i, i) : i = 1, \ldots K\}$. Note that the transition from (i, i) to $(i, i) + (1, 1)$ requires one call to $T^r((i, i) + (1, 1))$ that returns $(i, i) + (1, 1)$; so the call to $T^r((i, i) + (1, 1))$ only involves one calculation of best-responses in $\Gamma^r((i, i) + (1, 1))$, which requires $2(K - i)$ payoff-function evaluations, as each player in $\Gamma^r((i, i) + (1, 1))$. has $K - i$ strategies. The transition from (i, i) to $(i, i) + (1, 1)$ then requires a test in step 3 of the algorithm, this test needs at worst $2i$ function evaluations—in the case that all previous states turned out not to be equilibria of Γ. Hence, each iteration of Algorithm 3 involves at worst $2i + 2(K - i) = 2K$ payoff-function evaluations. We assumed that $M = \{(i, i) : i = 1, \ldots K\}$, so there are at most K iterations. Since $K = \overline{K}$, this proves that the algorithm needs at most $2\overline{K}^2$ payoff-function evaluations.

The argument above and the accounting procedure in (Aho, Hopcroft, and Ullman, 1974, p. 35–38) imply that Algorithm 3 is $O(r\overline{K}^2)$. □

References

Aho, A. V., J. E. Hopcroft, and J. D. Ullman (1974): *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA.

Amir, R., and V. E. Lambson (2000): "On the Effects of Entry in Cournot Markets," *The Review of Economic Studies*, 67(2), 235–254.

Bernstein, F., and A. Federgruen (2004a): "Decentralized Supply Chains with Competing Retailers under Demand Uncertainty," Forthcoming in Management Science.

——— (2004b): "A General Equilibrium Model for Industries with Price and Service Competition," Forthcoming in Operations Research.

Cachon, G. P. (2001): "Stock wars: inventory competition in a two echelon supply chain," *Operations Research*, 49(5), 658–674.

Cachon, G. P., and M. A. Lariviere (1999): "Capacity choice and allocation: strategic behavior and supply chain performance," *Management Science*, 45(8), 1091–1108.

Cachon, G. P., and S. Netessine (2003): "Game Theory in Supply Chain Analysis," Forthcoming in the collecction "Supply Chain Analysis is the eBusiness Era.".

Crooke, P., L. M. Froeb, S. Tschantz, and G. J. Werden (1997): "Properties of Computed Post-Merger Equilibria," Mimeo, Vanderbildt University, Department of Mathematics.

Echenique, F. (2002): "Finding All Equilibria," Caltech Social Science Working Paper 1153.

Echenique, F., and A. Edlin (2002): "Mixed Strategy Equilibria in Games of Strategic Complements are unstable," Working Paper E02-316, UC Berkeley.

Judd, K. L., S. Yeltekin, and J. Conklin (2000): "Computing Supergame Equilibria," mimeo Hoover Institution.

Lippman, S. A., and K. F. McCardle (1997): "The Competitive Newsboy," *Operations Research*, 45(1), 54–65.

McKelvey, R. D., and A. McLennan (1996): "Computation of Equilibria in Finite Games," in *Handbook of Computational Economics*, ed. by H. M. Amman, D. A. Kendrick, and J. Rust, vol. 1. North Holland, Amsterdam.

Milgrom, P., and J. Roberts (1990): "Rationalizability, Learning and Equilibrium in Games with Strategic Complementarities," *Econometrica*, 58(6), 1255–1277.

Milgrom, P., and C. Shannon (1992): "Monotone Comparative Statics," Stanford Institute for Theoretical Economics Working Paper.

——— (1994): "Monotone Comparative Statics," *Econometrica*, 62(1), 157–180.

Netessine, S., and N. Rudi (2003): "Supply Chain choice on the Internet," Mimeo, Upenn, http://www.netessine.com/.

Netessine, S., and R. Shumsky (2003): "Revenue management games: horizontal and vertical competition," Mimeo, Upenn, http://www.netessine.com/.

Robinson, J. (1951): "An Iterative Method of Solving a Game," *The Annals of Mathematics*, 54(2), 296–301.

Topkis, D. M. (1979): "Equilibrium Points in Nonzero-Sum n-Person Submodular Games," *SIAM Journal of Control and Optimization*, 17(6), 773–787.

——— (1998): *Supermodularity and Complementarity.* Princeton University Press, Princeton, New Jersey.

Vives, X. (1990): "Nash Equilibrium with Strategic Complementarities," *Journal of Mathematical Economics*, 19(3), 305–321.

——— (1999): *Oligopoly Pricing.* MIT Press, Cambridge, Massachusetts.

von Stengel, B. (2002): "Computing Equilibria for Two-Person Games," in *Handbook of Game Theory*, ed. by R. J. Aumann, and S. Hart, vol. 3. North Holland, Amsterdam.

von Stengel, B., A. van den Elzen, and D. Talman (2002): "Computing Normal-Form Perfect Equilibria for Extensive Two-Person Games," *Econometrica*, 70(2), 693–715.

Werden, G. J., L. M. Froeb, and S. Tschantz (2001): "The Effects of Merger Synergies on Consumers of Differentiated Products," Mimeo, US Department of Justice, Antitrust Division.

Zhou, L. (1994): "The Set of Nash Equilibria of a Supermodular Game Is a Complete Lattice," *Games and Economic Behavior*, 7(2), 295–300.

HSS 228-77, California Institute of Technology, Pasadena CA 91125.
E-mail address: `fede@caltech.edu`
URL: `http://www.hss.caltech.edu/~fede/`