# FINDING ALL EQUILIBRIA

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). I wish to emphasize five features of the algorithm:

(1) It works only on GSC. But then GSC are common in many areas of economics—Vives's (1999) textbook attests to that. See also Milgrom and Roberts (1990), Milgrom and Shannon (1992), Topkis (1998), and Vives (1990) for many economic examples of GSC.
(2) 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). Pure-strategy equilibria always exist in GSC (Topkis 1979, Vives 1990).
(3) It is fast. For example, it needs less than 5 seconds to find all equilibria in a two-player game where each player has 20.000 strategies.
(4) It is simple. I use the algorithm "by hand" on some bimatrix games to show that the algorithm is very simple to apply.
(5) For generic two-player GSC, I show that the algorithm is computationally efficient.

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 essentially only a trivial way of finding *all* pure equilibria: enumerate all strategy profiles and examine them one-by-one to see if they are equilibria. I shall call this way the "trivial algorithm." Not surprisingly, the trivial algorithm is very slow, and computationally

*Date*: January 16, 2003.
*JEL Classification.* C63, C72.
I thank Matt Jackson, Ivana Komunjer, Andy McLennan, John Rust, Ilya Segal, Chris Shannon, and Bernhard von Stengel for comments and suggestions on an early draft.

infeasible on large games. For example, the trivial algorithm needed 15 days to perform a simulation that my algorithm did in 5 minutes (see Section 7).

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 have developed applications of the algorithm that either illustrate how it works or show that the algorithm is efficient. I have not stressed economic applications; the paper is a paper on methodology. Nevertheless, there are many potential applications for the algorithm. I shall mention two examples.

First, the US Department of Justice needs to predict the consequences of mergers between firms; I claim that my algorithm can potentially be of help. They currently postulate a model of a market, and compute a Nash equilibrium before and after the merger of some firms in the market. But their conclusions may be different 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. The models used are often Bertrand models with differentiated products (see for example Werden, Froeb, and Tschantz (2001) or Crooke, Froeb, Tschantz, and Werden (1997)). [1] It turns out that Bertrand models with differentiated products are, under some conditions, GSC (Vives 1999, Milgrom and Shannon 1994). My algorithm could then be used to compare all equilibria before and after the merger.

Second, finding all the equilibria of a game is particularly important for the design of experiments. The designer needs to compare the observed outcomes with the equilibrium predictions. Further, some of the most important experimental studies involve GSC (Cooper, DeJong, Forsythe, and Ross 1990, van Huyck, Battalio, and Beil 1990). The designer has his/her subjects playing a GSC, and needs to find all equilibria of the GSC. The algorithm I present can then be applied in the design of experiments.

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

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 discusses an algorithm for a special class of GSC. Section 7 presents computational results for simulations of GSC.

## 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 $\phi$ 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 $\Gamma$—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 $\Gamma$ is understood I shall write $\beta_i$ for $\beta_{i,\Gamma}$ and $\beta$ for $\beta_\Gamma$.

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 $\Gamma$. When $\Gamma$ 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

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

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

*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 $i$, $\beta_i$ 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 $\Gamma$ 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 $\Gamma$, 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 $\beta$ 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 5.** *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 6.** ***(Topkis 1979)*** $\underline{T}(\inf S)$ *stops at the smallest Nash equilibrium of* $\Gamma$*, and* $\overline{T}(\sup S)$ *stops at the largest Nash equilibrium of* $\Gamma$.

See Topkis (1979) (or Topkis (1998)) for a proof of Lemma 6.

*Remark* 7. 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.

## 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, $\Gamma$. Suppose that player 1 has strategy set $S_1 = \{1, 2, \dots 15\}$, and player 2 has $S_2 = \{1, 2, \dots 11\}$ (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. I do not specify the players' payoffs because it is not necessary to understand how the algorithm works, but suppose that we have calculated the players' best-response functions (to makes things simple, assume best-responses are everywhere unique), $\beta_1$ and $\beta_2$. The game's best-response function is $\beta$, where $\beta(s_1, s_2) = (\beta_1(s_2), \beta_2(s_1))$. Because $\Gamma$ is a GSC, $\beta_1, \beta_2$ and $\beta$ are monotone increasing functions (Lemma 2).

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 $\beta$ is monotone, we have

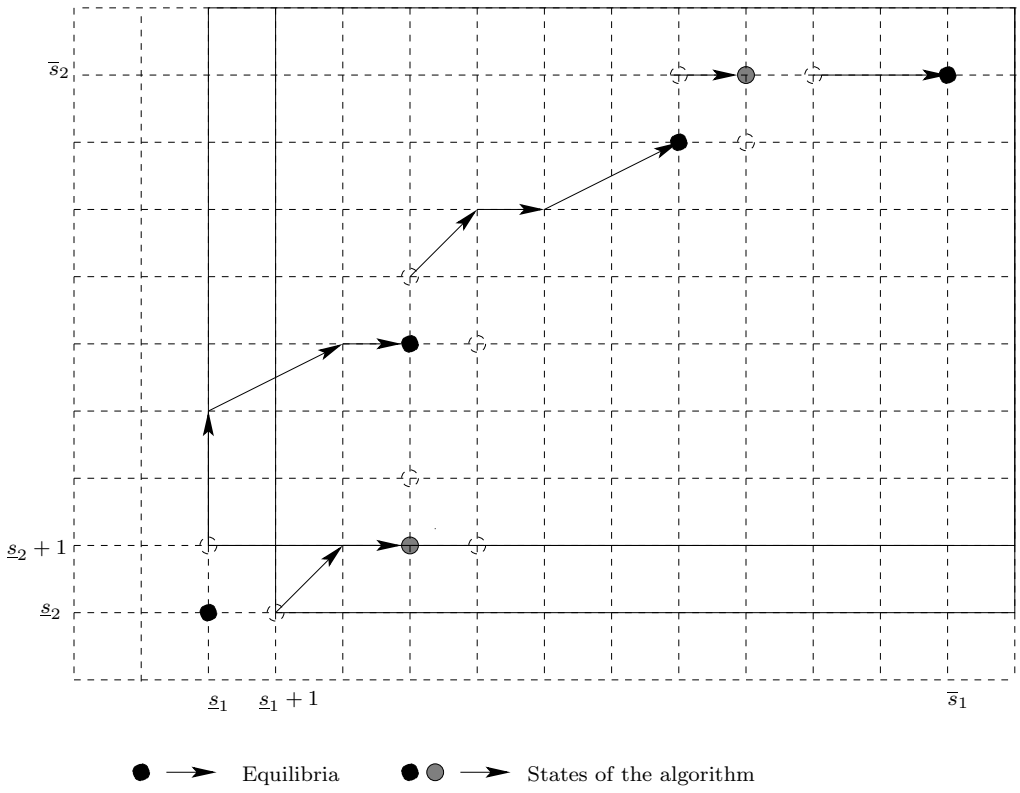


FIGURE 1. The algorithm in a two-player game.

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 $\beta$ 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.

It turns out that $\underline{s}$ is the *smallest* Nash equilibrium in $\Gamma$: Let $s^*$ be any other equilibrium, and note that $(1,1) \leq s^*$. Monotonicity of $\beta$ implies that $\beta(1,1) \leq \beta(s^*) = s^*$. Then, iterating $\beta$ 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 ◯ 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 $\beta$ 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 ◯ to the right of $(\underline{s}_1, \underline{s}_2)$, and the strategy space in $\Gamma^r(\underline{s}_1 + 1, \underline{s}_2)$ is the interval $[(\underline{s}_1 + 1, \underline{s}_2), (15, 11)]$ shown with non-dotted lines in the figure. Use RT to find $s^2$, the smallest Nash equilibrium in $\Gamma^r(\underline{s}_1 + 1, \underline{s}_2)$.
(3) Check if $s^1$ and $s^2$ are Nash equilibria of $\Gamma$. First consider $s^1$. Because $s^1$ is an equilibrium of $\Gamma^r(\underline{s}_1, \underline{s}_2 + 1)$, and $\beta$ 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 $s^k$s; note that the algorithm starts at larger and larger ◯-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 $\Gamma$) 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 $\beta$ 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, I have larger and larger $\hat{s}$'s, but I normally also have larger and larger confirmed equilibria; 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 $\Gamma$. 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 $s^k$s 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

I need a notational convention: $e^{d_i}_l$ is 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 8.** *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 do deviate from $\hat{s}_j$ to a strategy in the set*
$$\left\{z \in S_j : s^*_j \leq z \text{ and } (s_i + e^{d_i}_l, s_{-i})_j \not\leq z\right\}.$$
*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 9.** *The set* $\hat{\mathcal{E}}$ *produced by Algorithm 8 coincides with the set* $\mathcal{E}$ *of Nash equilibria of* $\Gamma$*.*

*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 8. 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 8.

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 6, $s'$ 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 $(s_i + e_l^{d_i}, s_{-i})_j \not\leq z_j$ 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 8 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 6 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 8 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* 10. A modification of Algorithm 8 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 9 says that Algorithm 8 works. In the rest of the paper I show that it is efficient.

## 5. Examples

I present two examples. The examples serve two purposes: First, they show how the algorithm works. Second, they show why it is likely to be fast. In fact, the second example suggests what features of a game would make the algorithm be inefficient, which motivates the sufficient conditions for the algorithm to be efficient in Section 6 and the simulations in Section 7.

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 8 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.

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 $\Gamma$. 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 $\{(1,1),(2,3),(4,4)\}$.

| | $\mathcal{M}$ | $\hat{\mathcal{E}}$ |
|---|---|---|
| 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

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 the players are indifferent between many different strategies. I show in Section 6 that, if we rule out indifference altogether, the algorithm is guaranteed to be efficient. Thus, generically in two-player games, the algorithm is efficient.

Example 2 suggests that, if players are indifferent at some points in the strategy space, but not for a very large set of strategies, the algorithm is likely to be fast. In Section 7 I simulate games that are guaranteed to have some indifference, and show that the algorithm is indeed still very fast.

## 6. Two-player games with strict preferences

Let $\Gamma$ be a two-player game where players have strict preferences and $d_1 = d_2 = 1$. I present a simple version of Algorithm 8 that finds all the equilibria of $\Gamma$. I can bound the complexity of this simple version of Algorithm 8. The complexity in the worst-case calculation is low relative to the (best- or worst-case) complexity of the trivial algorithm.

**Algorithm 11.** *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* $m + (1,1) \leq \overline{s}$*, then do steps 1-4:*

(1) *Let* $s^*$ *be a maximal element in*

$$\left\{\tilde{s} \in \hat{\mathcal{E}} : \tilde{s} \leq m + (1,1)\right\}.$$

(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 do 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 11 makes an *iteration* each time it does steps 1-4. Say that Algorithm 11 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 12.** *Algorithm 11 finds all Nash equilibria in* $O(r\overline{K}^2)$ *time, and does at most* $\underline{K}$ *iterations.*

*Proof.* First I prove that Algorithm 11 is well-behaved and stops when it says that it stops. Let $M \subseteq S$ be the set of states visited by Algorithm 11. Note that, for all $m \in M$, $m + (1,1) \leq m'$, 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 $\Gamma$ 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 9, 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 11 produces one and only one element of $M$, so there are no more iterations than there are elements in $M$. Second, $M \subseteq \{1, \dots K_1\} \times \{1, \dots K_2\}$, 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 $\{1, \dots K_1\}$ or $\{1, \dots K_2\}$. Thus, $M$ has not more than $\min\{K_1, K_2\} = \underline{K}$ elements.

Now I shall prove that Algorithm 11 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 11, even if it means fewer iterations of RT; this is because each iteration of Algorithm 11 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 $M = \{(i,i) : i = 1, \dots K\}$. 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 $\Gamma$. Hence, each iteration of Algorithm 11 involves at worst $2i+2(K-i)=2K$ payoff-function evaluations. We assumed that $M=\{(i,i):i=1,\dots K\}$, 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 11 is $O(r\overline{K}^2)$. □

*Remark* 13. The bounds in Theorem 12 say how Algorithm 11 performs compared to the trivial algorithm: Assume that $K_1=K_2=K$. The trivial algorithm requires $2K^3$ function evaluations—it needs to check if $(s_1,s_2)$ is an equilibrium for $K^2$ different values of $(s_1,s_2)$, and each check requires two best-response calculations, i.e. $2K$ function evaluations. So the trivial algorithm finds all equilibria in $O(rK^3)$ time. This is the effort that the trivial algorithm *must* make. The bound in Theorem 12 is from the worst-case calculation.

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 $n$. For any $s_n\in S_n$, the resulting $(n-1)$-player game is also a GSC, and von Stengel's suggestion can thus be applied to my algorithm as well. From Theorem 12 and recursion, one concludes that Algorithm 8 is still better than the trivial algorithm.

## 7. Performance

I evaluate the performance of Algorithm 8, 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 8 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 8 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 8 is faster the smaller is the number of

| Algorithm | $K$ | Avg. Eq. | Time | Games | Total Time |
|---|---|---|---|---|---|
| trivial | 1.000 | 28.8 | 43.6 min. | 500 | 15.1 days |
| 8 | 1.000 | 28.8 | 0.6 sec. | 500 | 5 min. |
| 8 | 3.000 | 39.7 | 0.4 sec. | 2.000 | 13.3 min. |
| 8 | 20.000 | 64.0 | 4.7 sec. | 2.000 | 2.6 hours |
| 8 | 40.000 | 64.3 | 15.5 sec. | 2.000 | 8.6 hours |

TABLE 3. Simulations (first two results are from a slower computer).

equilibria, and I want to evaluate Algorithm 8 using games where it does not have an apriori advantage.

I use two-player games, where 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 $\alpha_i$ and $\beta_i$ 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 6); the second summand also helps in getting multiple equilibria. The third and fourth summand are variants of polynomial terms that I found—by trial and error—often produce multiple equilibria. Note that, with these payoffs, the game is a GSC.

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

7.2. **Results.** I now discuss the results of a series of simulations. The results are in Table 3. I first compare the performance of Algorithm 8 and the trivial algorithm. Then I discuss what the simulations say about Algorithm 8 in general.

Consider the first two lines of the table: I used the algorithms on 500 simulated games, using $K = 1.000$—so each player had 1.000 strategies to choose from. In each individual game, the parameters $\alpha_i$ and $\beta_i$ were generated at pseudo-random from a uniform distribution on $[0, 1]$; I used Algorithm 8 and the trivial algorithm on the resulting game. On average, Algorithm 8 needed 0.6 seconds to find all equilibria of $\Gamma$. On average, the trivial algorithm needed 43.6 minutes to do the same work. Just to stress the difference, note that Algorithm 8 needed only 5 minutes to find the

equilibria of the 500 games while the trivial algorithm needed more than 15 days to do the same work!

Table 3 contains the results of other simulations as well. (The first two simulations in the table are from a slower computer than the others—I used a computer that could be dedicated to the simulation for 15 days. [2]) Algorithm 8 is fast, even on very large problems. On average, Algorithm 8 needs only 4.7 seconds to find all equilibria in a game where each player has 20.000 strategies, and 15.5 seconds when each player has 40.000 strategies. Again, the results are particularly striking if one looks at the "Total Time" column: Algorithm 8 allows one to simulate 2.000 large games in a few hours.

Using the calculation in Remark 13, and the results for $K = 1.000$, we can find the time that it would take the trivial algorithm to find all equilibria of one game with $K = 20.000$: 241 days. Further, when $K = 40.000$ it would take the trivial algorithm 5 years and 3 months to find the equilibria of *one* game!

Note that the speed of Algorithm 8 does make a difference in the analysis of these games. The average number of equilibria stabilizes around 64, after several increases in $K$. We can then infer that the original game with continuous strategy spaces has on average about 64 equilibria. If we were limited to the trivial algorithm, we would be unable to work with a fine enough grid to infer the number of equilibria of the limiting game.

The graph in Figure 3 has more information. The graph plots the individual simulations for the $K = 40.000$ case. First, Algorithm 8 never needs more than 100 seconds to find the equilibria; and it needs less than one minute for all but a few games. Second, note that more equilibria require more time, and that the relation between the number of equilibria and time is approximately linear—in fact, a linear regression has an $R^2$ of 0.93.

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 8 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.

[2]The computer used in the first two simulations is a Sun Ultra 5 with a 360 MHz CPU and 512 MB Ram. I did the other simulations on a Linux Dell Precision PC with a 1.8 GHz Xeon CPU and 512 MB Ram.

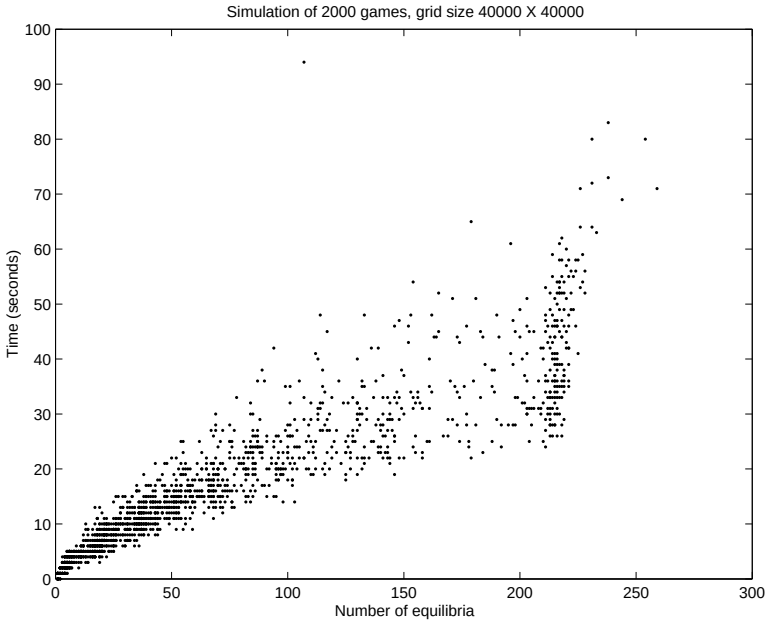


FIGURE 3. Relation between time and number of equilibria

## REFERENCES

AHO, A. V., J. E. HOPCROFT, AND J. D. ULLMAN (1974): *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA.

COOPER, R. W., D. V. DEJONG, R. FORSYTHE, AND T. W. ROSS (1990): "Selection Criteria in Coordination Games: Some Experimental Results," *American Economic Review*, 80(1), 218–233.

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., 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.

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.

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.

VAN HUYCK, J. B., R. C. BATTALIO, AND R. O. BEIL (1990): "Tacit Coordination Games, Strategic Uncertainty, and Coordination Failure," *American Economic Review*, 80(1), 234–248.

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.

HSS 228-77, CALIFORNIA INSTITUTE OF TECHNOLOGY, PASADENA CA 91125.
*E-mail address*: fede@hss.caltech.edu
*URL*: http://www.hss.caltech.edu/~fede/