Sparse Games Are Hard

Xi Chen[1], Xiaotie Deng[2], and Shang-Hua Teng[3]

[1] Department of Computer Science, Tsinghua University, Beijing
xichen00@mails.tsinghua.edu.cn
[2] Department of Computer Science, City University of Hong Kong, Hong Kong
deng@cs.cityu.edu.hk
[3] Department of Computer Science, Boston University, Boston
steng@cs.bu.edu

Abstract. A two-player game is sparse if most of its payoff entries are zeros. We show that the problem of computing a Nash equilibrium remains **PPAD**-hard to approximate in fully polynomial time for sparse games. On the algorithmic side, we give a simple and polynomial-time algorithm for finding exact Nash equilibria in a class of sparse win-lose games.

1 Introduction

Motivated by the growing possibilities in both Internet applications and network computations, Game Theory has attracted a great deal of attention from Theoretical Computer Science community. Central to such game theoretical applications is the problem of computing a Nash equilibrium in a non-cooperative game.

A series of significant progress on the complexity of this problem was initiated by a recent work of Daskalakis, Goldberg, and Papadimitriou [1,2] who introduced a reduction technique and showed that a Nash equilibrium in a four-player game is hard to find, unless **PPAD** [3] is in **P**. Shortly afterward, this hardness result was extended to three-player games [4,5]. Chen and Deng [6] finally settled a long-term open problem, and proved that computing a Nash equilibrium in a two-player game is **PPAD**-complete.

These breakthrough work left the problem of computing approximate Nash equilibria with less than exponential accuracy as a central remaining open question in the area of computing Nash equilibria. In a recent paper [7], we solved this problem by showing that two-player games do not have a fully polynomial-time approximation scheme unless **PPAD** is in **P**. Hence, it is unlikely that the $n^{O(\log n/\epsilon^2)}$-time algorithm of Lipton, Markakis, and Mehta [8], the fastest algorithm known today for approximating Nash equilibria, can be further improved to poly$(n, 1/\epsilon)$. This result also implies that, unlike the simplex algorithm for zero-sum two-player games [9], the smoothed complexity of the classical Lemke-Howson algorithm for non-zero-sum two-player games is not polynomial, unless **PPAD** $\subseteq$ **RP**. Thus the average-case polynomial-time result of Barany, Vempala, and Vetta [10] is not likely extendible to the smoothed model. Recently, Chen, Teng, and Valiant [11] extended this result and proved that win-lose two-player games, in which the payoff entries are either 0 or 1, are **PPAD**-hard to approximate in fully polynomial time.

A two-player game is specified by two $m \times n$ matrices $\mathbf{A} = (a_{i,j})$ and $\mathbf{B} = (b_{i,j})$. They state the payoffs when the first player makes a choice of a row and the second player makes a choice of a column. In general, each player can pick a distribution over its choices in advance, and during the playing time, selects a choice according to this distribution, simultaneously. The concept of a Nash equilibrium captures the notion of rational play in such non-cooperative games. It is rather a strong notion of rationality, stating the condition that neither player can gain by changing its own distribution, when the opponent's distribution is revealed. Each two-player game has at least one Nash equilibrium [12].

In this paper, we consider sparse games in which most of the payoff entries are zeros. Particularly, we focus on sparse two-player games in which each row and column of the two payoff matrices has at most a constant number of non-zero entries. We prove that a Nash equilibrium in such sparse games is equally hard to compute and essentially equally hard to approximate as in general two-player games. Our result shows that sparsity alone does not make game easier to solve and that sparse two-player games do not have a fully polynomial-time approximation scheme unless $\mathbf{PPAD} \subseteq \mathbf{P}$.

To establish our complexity result, we construct a set of new arithmetic and logic gadgets, for the reduction from a discrete Brouwer's fixed point problem to an equilibrium computation problem. These new gadgets enable us to reduce the degree of influence in the simulation of arithmetic and logic computations in two-player games, resulting in hard sparse instances.

On the positive side, we give a polynomial-time algorithm for computing an exact Nash equilibrium for a subclass of sparse win-lose games. Our algorithm takes advantage of the 0-1 payoff structure and effectively reduces the computation of a Nash equilibrium of a two-player win-lose game to the computation of an equilibrium in a smaller game. We were informed by the conference committee that Codenotti, Leoncini, and Resta [13] very recently and independently obtained the same result for finding Nash equilibria in this subclass of sparse win-lose games. As our algorithm appears to be simpler than theirs, we decide to keep our algorithm and its analysis in this conference version.

2 Sparse Two-Player Games and Our Main Result

Definition 1 (Sparse Normalized Games). *A bimatrix game $\mathcal{G} = (\mathbf{A}, \mathbf{B})$ is normalized if every entry of matrices $\mathbf{A}$ and $\mathbf{B}$ is between -1 and 1. A matrix $\mathbf{A}$ is row (column) sparse if there are at most 10 nonzero entries in every row (column). $\mathbf{A}$ is sparse if it is both row sparse and column sparse. A two-player game $\mathcal{G} = (\mathbf{A}, \mathbf{B})$ is sparse if both $\mathbf{A}$ and $\mathbf{B}$ are sparse.*

We use $\mathbb{P}^n$ to denote the set of all *probability vectors* in $\mathbb{R}^n$, i.e., non-negative vectors whose entries sum to 1. Recall that an *ϵ-approximate Nash equilibrium* of game $(\mathbf{A},\mathbf{B})$ is a pair $(\mathbf{x}^* \in \mathbb{P}^m, \mathbf{y}^* \in \mathbb{P}^n)$ such that, for all probability vectors $\mathbf{x} \in \mathbb{P}^m, \mathbf{y} \in \mathbb{P}^n$,

$$(\mathbf{x}^*)^T \mathbf{A}\mathbf{y}^* \geq \mathbf{x}^T \mathbf{A}\mathbf{y}^* - \epsilon \text{ and } (\mathbf{x}^*)^T \mathbf{B}\mathbf{y}^* \geq (\mathbf{x}^*)^T \mathbf{B}\mathbf{y} - \epsilon.$$

Following [7], an *ϵ-well-supported Nash equilibrium* of game $(\mathbf{A},\mathbf{B})$ is a pair $(\mathbf{x}^*,\mathbf{y}^*)$, such that for all i,j, $\langle \mathbf{b}_i|\mathbf{x}^*\rangle > \langle \mathbf{b}_j|\mathbf{x}^*\rangle + \epsilon \Rightarrow y_j^* = 0$, and $\langle \mathbf{a}_i|\mathbf{y}^*\rangle > \langle \mathbf{a}_j|\mathbf{y}^*\rangle + \epsilon \Rightarrow x_j^* = 0$, where $\mathbf{a}_i$ and $\mathbf{b}_i$ denote the i^{th} row of $\mathbf{A}$ and the i^{th} column of $\mathbf{B}$, respectively. Motivated by the next lemma proved in [7]. we define the following search problem called Sparse Bimatrix.

Lemma 1 ([7]). *In a normalized game $(\mathbf{A},\mathbf{B})$, for every $0 \leq \epsilon \leq 1$, (1) every ϵ-well-supported Nash equilibrium $(\mathbf{x},\mathbf{y})$ is also an ϵ-approximate Nash equilibrium; (2) from every $\epsilon^2/(8n)$-approximate Nash equilibrium $(\mathbf{u},\mathbf{v})$, one can find in polynomial time an ϵ-well-supported Nash equilibrium $(\mathbf{x},\mathbf{y})$.*

Definition 2 (Sparse Bimatrix**).** *The input instance is a bimatrix game $\mathcal{G} = (\mathbf{A},\mathbf{B})$ which is both normalized and sparse. $\mathbf{A}$ and $\mathbf{B}$ are $n \times n$ matrices.*

The output is an n^{-6}-well-supported Nash equilibrium of game $\mathcal{G}$.

Our main result is the following theorem.

Theorem 1 (Main). *Problem* Sparse Bimatrix *is* **PPAD***-complete.*

Clearly, Sparse Bimatrix belongs to **PPAD** [6]. To prove its completeness, we will reduce the **PPAD**-complete problem Brouwer^f [7] to it, where $f(n) = 3$. We also notice that, in contrast, a $(10/n)$-approximate Nash equilibrium of a sparse normalized game can be found in polynomial time.

3 Review of the Reduction in [7]

In this section, we review the reduction in [7], from Brouwer^f to the problem of finding an n^{-6}-well-supported Nash equilibrium in a normalized game.

Let $U = (C, 0^{3n})$ be an input instance of Brouwer^f, where C is a boolean circuit. Let m be the smallest integer such that $2^m > \text{Size}[C] > n$. Here we let $\text{Size}[C]$ denote the number of gates plus the number of input and output variables in C. In the reduction, we construct a game $\mathcal{G}^U = (\mathbf{A}^U, \mathbf{B}^U)$ in polynomial time, where $\mathbf{A}^U$ and $\mathbf{B}^U$ are $N \times N = 2^{6m+1} = 2K$ matrices, satisfying

Property $\mathbf{P}_1$: $|a_{i,j}^U|, |b_{i,j}^U| \leq N^3$ for all i,j: $1 \leq i,j \leq N$;

Property $\mathbf{P}_2$: From every ϵ-well-supported Nash equilibrium of $\mathcal{G}^U$, where $\epsilon = 2^{-18m} = 1/K^3$, one can find a panchromatic simplex P of circuit C in polynomial time.

Then we normalize $\mathcal{G}^U$ to obtain $\overline{\mathcal{G}^U} = (\overline{\mathbf{A}^U}, \overline{\mathbf{B}^U})$ by setting $\overline{\mathbf{A}^U} = \mathbf{A}^U/N^3$ and $\overline{\mathbf{B}^U} = \mathbf{B}^U/N^3$. **Property $\mathbf{P}_2$** implies that, from any $1/N^6$-well-supported Nash equilibrium of $\overline{\mathcal{G}^U}$, one can find a panchromatic simplex of circuit C efficiently.

As a result, the problem of finding an n^{-6}-well-supported Nash equilibrium in a normalized bimatrix game is **PPAD**-hard.

The construction of $\mathcal{G}^U$ starts with a zero-sum game $\mathcal{G}^* = (\mathbf{A}^*, \mathbf{B}^*)$ called Matching Pennies with payoff parameter $M = 2^{18m+1} = 2K^3$. $\mathbf{A}^*$ is a $K \times K$ block diagonal matrix, where each block is a 2×2 matrix of all M's, and $\mathbf{B}^* = -\mathbf{A}^*$. Ultimately, we obtain $\mathcal{G}^U$ by perturbing the payoff entries of $\mathcal{G}^*$.

At a high level, we partition the rows of $\mathcal{G}^*$ and hence of $\mathcal{G}^U$ into K groups: the i^{th} group consists of rows $2i-1, 2i$. Every row group $(2i-1, 2i)$ is referred as an *arithmetic node* v. Let V_A denote the set of all such nodes ($|V_A| = K$), and $\mathcal{C}_A$ denote the one-to-one correspondence from V_A to $\{1, 2...K\}$ such that v corresponds to the $\mathcal{C}_A(v)^{th}$ row group, for all $v \in V_A$. We also partition the columns of $\mathcal{G}^*$ into K groups: the j^{th} group consists of columns $2j-1, 2j$, and every group is referred as an *internal node* w. Let V_I denote the set of internal nodes and $\mathcal{C}_I$ denote the one-to-one correspondence from V_I to $\{1, 2...K\}$.

Let $(\mathbf{x} \in \mathbb{P}^N, \mathbf{y} \in \mathbb{P}^N)$ be a profile of mixed strategies. For each $v \in V_A$, we let $\mathbf{x}[v] = x_{2k-1}$ and $\mathbf{x}_C[v] = x_{2k-1} + x_{2k}$ denote the *value* and *capacity* of v in $(\mathbf{x}, \mathbf{y})$, respectively, where $k = \mathcal{C}_A(v)$. For each $w \in V_I$, we let $\mathbf{y}[w] = y_{2t-1}$ and $\mathbf{y}_C[w] = y_{2t-1} + y_{2t}$ denote the *value* and *capacity* of w in $(\mathbf{x}, \mathbf{y})$, respectively, where $t = \mathcal{C}_I(w)$. For $x, y \in \mathbb{R}$ and $c \in \mathbb{R}^+$, by $x = y \pm c$, we mean that $y - c \le x \le y + c$. All our perturbations of $\mathcal{G}^*$ have the following nice property.

Lemma 2 ([7]). *Let* $(\mathbf{A}, \mathbf{B})$ *be a game with* $0 \le \mathbf{A} - \mathbf{A}^*, \mathbf{B} - \mathbf{B}^* \le 1$. *For any* $t \le 1$, *let* $(\mathbf{x}, \mathbf{y})$ *be a t-well-supported Nash equilibrium of* $(\mathbf{A}, \mathbf{B})$, *then it must satisfy constraint* $\mathcal{P} = [\,\mathbf{x}_C[v] = 1/K \pm \epsilon,\ \mathbf{y}_C[w] = 1/K \pm \epsilon,\ \forall\ v \in V_A, w \in V_I\,]$.

To construct $\mathcal{G}^U$, we transform the prototype game $\mathcal{G}^*$ by adding "gadget" games: we first build a collection of gadgets $\mathcal{S}^U = \{T_1..., T_l\}$ for some $l < K$. Each $T \in \mathcal{S}^U$ defines [7] an $N \times N$ "gadget" game $(\mathbf{L}[T], \mathbf{R}[T])$. We then build game $\mathcal{G}^U$ by invoking function BuildGame on $\mathcal{S}^U$. BuildGame takes a collection $\mathcal{S}$ of gadgets and returns a bimatrix game $(\mathbf{A}, \mathbf{B})$ as

$$\mathbf{A} = \mathbf{A}^* + \textstyle\sum_{T \in \mathcal{S}} \mathbf{L}[T] \quad \text{and} \quad \mathbf{B} = \mathbf{B}^* + \textstyle\sum_{T \in \mathcal{S}} \mathbf{R}[T].$$

A gadget T is a 6-tuple (G, v_1, v_2, v, c, w). Here G is the type of the gadget where $G \in \{G_\zeta, G_{\times\zeta}, G_=, G_+, G_-, G_<, G_\wedge, G_\vee, G_\neg\}$. $v_1 \in V_A \cup \{nil\}$ and $v_2 \in V_A \cup \{nil\}$ are the first and second input nodes of T, respectively. $v \in V_A$ is the output node, and $w \in V_I$ is the internal node. Parameter $c \in \mathbb{R} \cup \{nil\}$ is only used in G_ζ and $G_{\times\zeta}$ gadgets: when $G = G_\zeta$, $0 \le c \le 1/K - \epsilon$; when $G = G_{\times\zeta}$, $0 \le c \le 1$; otherwise, $c = nil$.

Every gadget $T = (G, v_1, v_2, v, c, w)$ implements an arithmetic or logic constraint $\mathcal{P}[T]$, which requires the values of nodes v, v_1 and v_2 to satisfy certain functional relationship. All the nine types of constraints are listed in Figure 1. Among the nine types of gadgets, $G_\wedge, G_\vee$ and $G_\neg$ are logic gadgets. They are used to simulate the logic gates in C. Associated with probability vectors $(\mathbf{x}, \mathbf{y})$, the value of $v \in V_A$ represents boolean 1 ($\mathbf{x}[v] =_B 1$) if $\mathbf{x}[v] = \mathbf{x}_C[v]$; it represents boolean 0 ($\mathbf{x}[v] =_B 0$) if $\mathbf{x}[v] = 0$.

$$\begin{array}{ll}
G_+: & \mathcal{P}[T] = [\, \mathbf{x}[v] = \min(\mathbf{x}[v_1] + \mathbf{x}[v_2], \mathbf{x}_C[v]\,) \pm \epsilon \,] \\
G_\zeta: & \mathcal{P}[T] = [\, \mathbf{x}[v] = c \pm \epsilon \,] \\
G_{\times\zeta}: & \mathcal{P}[T] = [\, \mathbf{x}[v] = \min(\,c\mathbf{x}[v_1], \mathbf{x}_C[v]\,) \pm \epsilon \,] \\
G_=: & \mathcal{P}[T] = [\, \mathbf{x}[v] = \min(\mathbf{x}[v_1], \mathbf{x}_C[v]\,) \pm \epsilon \,] \\
G_<: & \mathcal{P}[T] = [\, \mathbf{x}[v] =_B 1 \text{ if } \mathbf{x}[v_1] < \mathbf{x}[v_2] - \epsilon;\ \mathbf{x}[v] =_B 0 \text{ if } \mathbf{x}[v_1] > \mathbf{x}[v_2] + \epsilon \,] \\
G_-: & \mathcal{P}[T] = [\, \min(\mathbf{x}[v_1] - \mathbf{x}[v_2], \mathbf{x}_C[v]\,) - \epsilon \le \mathbf{x}[v] \le \max(\mathbf{x}[v_1] - \mathbf{x}[v_2], 0\,) + \epsilon \,] \\
G_\neg: & \mathcal{P}[T] = [\, \mathbf{x}[v] =_B 0 \text{ if } \mathbf{x}[v_1] =_B 1;\ \mathbf{x}[v] =_B 1 \text{ if } \mathbf{x}[v_1] =_B 0 \,] \\
G_\vee: & \mathcal{P}[T] = \begin{bmatrix} \mathbf{x}[v] =_B 1 \text{ if } \mathbf{x}[v_1] =_B 1 \text{ or } \mathbf{x}[v_2] =_B 1; \\ \mathbf{x}[v] =_B 0 \text{ if } \mathbf{x}[v_1] =_B 0 \text{ and } \mathbf{x}[v_2] =_B 0 \end{bmatrix} \\
G_\wedge: & \mathcal{P}[T] = \begin{bmatrix} \mathbf{x}[v] =_B 0 \text{ if } \mathbf{x}[v_1] =_B 0 \text{ or } \mathbf{x}[v_2] =_B 0; \\ \mathbf{x}[v] =_B 1 \text{ if } \mathbf{x}[v_1] =_B 1 \text{ and } \mathbf{x}[v_2] =_B 1 \end{bmatrix}
\end{array}$$

Fig. 1. Constraint $\mathcal{P}[T]$, where $T = (G, v_1, v_2, v, c, w)$

The collection $\mathcal{S}^U$ we construct is *valid*, that is, for each pair $T = (G, v_1, v_2, v, w, c)$ and $T' = (G', v'_1, v'_2, v', c', w')$ in $\mathcal{S}^U$, $v \neq v'$ and $w \neq w'$. In [7], we prove the following two lemmas for valid collections of gadgets.

Lemma 3 ([7]). *Let $\mathcal{S}$ be a valid collection and $\mathcal{G} = (\mathbf{A}, \mathbf{B}) =$* BuildGame$(\mathcal{S})$*, then we have $0 \le \mathbf{A} - \mathbf{A}^*, \mathbf{B} - \mathbf{B}^* \le 1$. So, by Lemma 2, each ϵ-well-supported Nash equilibrium of $\mathcal{G}$ satisfies constraint $\mathcal{P}$.*

Lemma 4 ([7]). *Let $\mathcal{S}$ be a valid collection of gadgets, and $(\mathbf{x}, \mathbf{y})$ be any ϵ-well-supported Nash equilibrium of* BuildGame$(\mathcal{S})$*, then for each $T \in \mathcal{S}$, constraint $\mathcal{P}[T]$ as defined in Figure 1 is satisfied by $(\mathbf{x}, \mathbf{y})$.*

Property $\mathbf{P}_1$ follows directly from Lemma 3. From Lemma 3 and 4, every ϵ-well-supported Nash equilibrium of $\mathcal{G}^U$ satisfies a set of $|\,\mathcal{S}^U\,| + 1$ constraints: $\{\,\mathcal{P}, \mathcal{P}[T_1], ..., \mathcal{P}[T_l]\,\}$, which can be used to prove **Property $\mathbf{P}_2$**.

4 The New Reduction

Although the prototype game $\mathcal{G}^*$ is sparse (for each row and column, there are exactly two nonzero entries), $\mathcal{G}^U$ constructed in [7] is not always sparse:

1. There are three types of "bad" gadgets used in the construction of $\mathcal{G}^U$: G_ζ, $G_\wedge$ and $G_\vee$. For every $T = (G, v_1, v_2, v, c, w) \in \mathcal{S}^U$ with $G \in \{\,G_\zeta, G_\wedge, G_\vee\,\}$, every entry in the $(2\mathcal{C}_I(w))^{th}$ column of matrix $\mathbf{R}[T]$ is non-zero [7]. As a result, $\mathbf{B}^U$ is not column sparse.
2. There exist some arithmetic nodes $v \in V_A$ which are used by more than 5 gadgets in $\mathcal{S}^U$ as one of their input nodes. Suppose they are $T_1 ... T_k \in \mathcal{S}^U$, then in both the $(2\mathcal{C}_A(v) - 1)^{st}$ and $(2\mathcal{C}_A(v))^{th}$ rows of $\sum_{1 \le i \le k} \mathbf{R}[T_i]$, we have $2k > 10$ non-zero entries [7]. As a result, $\mathbf{B}^U$ is not row sparse.

In this section, we will reduce problem Brouwer^f to Sparse Bimatrix. The reduction is very similar to the one in [7]. We will develop new "gadget" games to overcome the first obstacle above. Then we will perturb the prototype game $\mathcal{G}^*$ to build a sparse game $\mathcal{H}^U$ which satisfies both **Property** $\mathbf{P}_1$ and $\mathbf{P}_2$. One can normalize the sparse game $\mathcal{H}^U$ to prove Theorem 1.

4.1 New Gadgets and Constraints

To build game $\mathcal{H}^U$, we transform the prototype game $\mathcal{G}^* = (\mathbf{A}^*, \mathbf{B}^*)$ by adding "gadget" games. We first build a collection $\mathcal{T}^U = \{T_1, ..., T_l\}$ of gadgets. For every gadget T, we construct a "gadget" game $(\mathbf{M}[T], \mathbf{N}[T])$ according to Figure 2. Given any collection of gadgets $\mathcal{T}$, one can construct a two-player game $(\mathbf{A}, \mathbf{B}) = \text{BuildGame}(\mathcal{T})$ by setting

$$\mathbf{A} = \mathbf{A}^* + \textstyle\sum_{T\in\mathcal{T}} \mathbf{M}[T] \quad \text{and} \quad \mathbf{B} = \mathbf{B}^* + \textstyle\sum_{T\in\mathcal{T}} \mathbf{N}[T].$$

From $\mathcal{T}^U$, we obtain game $\mathcal{H}^U = \text{BuildGame}(\mathcal{T}^U)$.

Here a gadget is a 7-tuple $T = (G, v_1, v_2, v_3, v, c, w)$. $v_3 \in V_A \cup \{nil\}$ is the auxiliary input node of T, while the meanings of all the other components are the same as those in the previous reduction. In the new reduction, we have totally eleven types of gadgets: $G \in \{G_+, G_-, G_=, G_<, G_{\times\zeta}, G_\neg, G_\zeta^*, G_\wedge^*, G_\vee^*, G_H, G_{B=}\}$. Similarly, every gadget T implements an arithmetic or logic constraint $\mathcal{R}[T]$, which requires the values of v_1, v_2, v_3, v to satisfy certain functional relationship. Before describing constraints $\mathcal{R}[T]$ for each type of gadgets, we claim the following two lemmas, whose proofs are very similar to those of Lemma 3 and Lemma 4 in [7]. Here a collection $\mathcal{T}$ is *valid* if for every pair $T = (G, v_1, v_2, v_3, v, c, w)$ and $T' = (G', v_1', v_2', v_3', v', c', w')$ in $\mathcal{T}$, $v \neq v'$ and $w \neq w'$.

Lemma 5. *Let $\mathcal{T}$ be a valid collection and* $(\mathbf{A}, \mathbf{B}) = \text{BuildGame}(\mathcal{T})$*, then we have* $0 \leq \mathbf{A} - \mathbf{A}^*, \mathbf{B} - \mathbf{B}^* \leq 1$*. So from Lemma 2, every ϵ-well-supported Nash equilibrium of* $(\mathbf{A}, \mathbf{B})$ *satisfies constraint* $\mathcal{P}$*.*

Lemma 6 (Gadget Constraints). *Let $\mathcal{T}$ be a valid collection of gadgets, and* $(\mathbf{x}, \mathbf{y})$ *be an ϵ-well-supported Nash equilibrium of* BuildGame$(\mathcal{T})$*, then for each $T \in \mathcal{T}$, constraint $\mathcal{R}[T]$ is satisfied by* $(\mathbf{x}, \mathbf{y})$*.*

By Lemma 5 and 6, every ϵ-well-supported Nash equilibrium $(\mathbf{x}, \mathbf{y})$ of game BuildGame$(\mathcal{T})$ satisfies $|\mathcal{T}| + 1$ constraints: $\{\mathcal{P}, \mathcal{R}[T], T \in \mathcal{T}\}$. Let $T = (G, v_1, v_2, v_3, v, c, w)$ be a gadget in $\mathcal{T}$, then $\mathcal{R}[T]$ on $(\mathbf{x}, \mathbf{y})$ is described as follows:

- If type $G \in \{G_{\times\zeta}, G_=, G_+, G_-, G_<, G_\neg\}$, then $v_3 = nil$. We have $\mathbf{M}[T] = \mathbf{L}[T']$ and $\mathbf{N}[T] = \mathbf{R}[T']$, where $T' = (G, v_1, v_2, v, c, w)$, and naturally, constraint $\mathcal{R}[T]$ is the same as $\mathcal{P}[T']$.
- If $G = G_{B=}$, then $v_1 \in V_A$ and $v_2 = v_3 = c = nil$. $(\mathbf{x}, \mathbf{y})$ satisfies constraint $\mathcal{R}[T] = [\,\mathbf{x}[v] =_B 1 \text{ if } \mathbf{x}[v_1] =_B 1;\ \mathbf{x}[v] =_B 0 \text{ if } \mathbf{x}[v_1] =_B 0\,]$.
- If $G = G_\zeta^*$, then the auxiliary input node $v_3 \in V_A$, $v_1 = v_2 = nil$, and $0 \leq c \leq 1/K - \epsilon$. $(\mathbf{x}, \mathbf{y})$ satisfies $\mathcal{R}[T] = [\,\mathbf{x}[v] = c \pm 4\epsilon \text{ if } \mathbf{x}[v_3] = 1/(2K) \pm \epsilon\,]$. So, $\mathcal{R}[T]$ is very close to constraint $\mathcal{P}[T']$, where $T' = (G_\zeta, nil, nil, v, c, w)$, when the value of the auxiliary input node v_3 in $(\mathbf{x}, \mathbf{y})$ is close to $1/(2K)$.

Construction of $\mathbf{M}[T]$ and $\mathbf{N}[T]$, where $T = (G, v_1, v_2, v_3, v, c, w)$

Set $\mathbf{M}[T] = (M_{i,j}) = \mathbf{N}[T] = (N_{i,j}) = 0$

$k = \mathcal{C}_A(v)$, $k_1 = \mathcal{C}_A(v_1)$, $k_2 = \mathcal{C}_A(v_2)$, $k_3 = \mathcal{C}_A(v_3)$ and $t = \mathcal{C}_I(w)$

G_+ : $M_{2k-1,2t-1} = M_{2k,2t} = N_{2k_1-1,2t-1} = N_{2k_2-1,2t-1} = N_{2k-1,2t} = 1$

G_- : $M_{2k-1,2t-1} = M_{2k,2t} = N_{2k_1-1,2t-1} = N_{2k_2-1,2t} = N_{2k-1,2t} = 1$

$G_=$: $M_{2k-1,2t-1} = M_{2k,2t} = N_{2k_1-1,2t-1} = N_{2k-1,2t} = 1$

$G_<$: $M_{2k-1,2t} = M_{2k,2t-1} = N_{2k_1-1,2t-1} = N_{2k_2-1,2t} = 1$

$G_{\times\zeta}$: $M_{2k-1,2t-1} = M_{2k,2t} = N_{2k-1,2t} = 1$, $N_{2k_1-1,2t-1} = c$

$G_\neg$: $M_{2k-1,2t} = M_{2k,2t-1} = N_{2k_1-1,2t-1} = N_{2k_1,2t} = 1$

G^*_ζ : $M_{2k-1,2t} = M_{2k,2t-1} = 1$, $N_{2k-1,2t-1} = 1/2$, $N_{2k_1-1,2t} = Kc$

$G^*_\wedge$: $M_{2k-1,2t-1} = M_{2k,2t} = N_{2k_3-1,2t} = 1$, $N_{2k_1-1,2t-1} = N_{2k_2-1,2t-1} = 1/3$

$G^*_\vee$: $M_{2k-1,2t-1} = M_{2k,2t} = N_{2k_1-1,2t-1} = N_{2k_2-1,2t-1} = N_{2k_3-1,2t} = 1$

$G_{B=}$: $M_{2k-1,2t-1} = M_{2k,2t} = N_{2k_1-1,2t-1} = N_{2k_1,2t} = 1$

G_H : $M_{2k-1,2t} = M_{2k,2t-1} = N_{2k-1,2t-1} = N_{2k,2t} = 1$

Fig. 2. Construction of "Gadget" Game $(\mathbf{M}[T], \mathbf{N}[T])$

– If $G = G^*_\vee$, then $v_1, v_2, v_3 \in V_A$ and $c = nil$. $(\mathbf{x}, \mathbf{y})$ satisfies constraint $\mathcal{R}[T]$

$$\left[\ \mathbf{x}[v_3] = \frac{1}{2K} \pm \epsilon \implies \left\{ \begin{array}{c} \mathbf{x}[v] =_B 1 \text{ if } \mathbf{x}[v_1] =_B 1 \text{ or } \mathbf{x}[v_2] =_B 1 \\ \mathbf{x}[v] =_B 0 \text{ if } \mathbf{x}[v_1] =_B 0 \text{ and } \mathbf{x}[v_2] =_B 0 \end{array} \right\} \right].$$

Similarly, if $G = G^*_\wedge$, then $(\mathbf{x}, \mathbf{y})$ must satisfy constraint $\mathcal{R}[T]$

$$\left[\ \mathbf{x}[v_3] = \frac{1}{2K} \pm \epsilon \implies \left\{ \begin{array}{c} \mathbf{x}[v] =_B 1 \text{ if } \mathbf{x}[v_1] =_B 1 \text{ and } \mathbf{x}[v_2] =_B 1 \\ \mathbf{x}[v] =_B 0 \text{ if } \mathbf{x}[v_1] =_B 0 \text{ or } \mathbf{x}[v_2] =_B 0 \end{array} \right\} \right].$$

Clearly, constraint $\mathcal{R}[T]$ is the same as $\mathcal{P}[T']$ where $T' = (G_\vee \text{ or } G_\wedge, v_1, v_2, v, c, w)$, when the value of v_3 in $(\mathbf{x}, \mathbf{y})$ is close to $1/(2K)$.

– If type $G = G_H$, then $v_1 = v_2 = v_3 = nil$. $(\mathbf{x}, \mathbf{y})$ satisfies $\mathcal{R}[T] = [\ \mathbf{x}[v] = 1/2K \pm \epsilon\ ]$. We will use G_H gadgets to "generate" auxiliary nodes for G^*_ζ, $G^*_\wedge$ and $G^*_\vee$ gadgets to simulate the old G_ζ, $G_\wedge$ and $G_\vee$ gadgets used in [7].

We next show that, if a valid collection $\mathcal{T}$ also satisfies the following property, then bimatrix game BuildGame($\mathcal{T}$) must be sparse.

Definition 3. *Collection $\mathcal{T}$ is said to be* sparse *if for every v^* in V_A, there exist at most two gadgets $T = (G, v_1, v_2, v_3, v, c, w) \in \mathcal{T}$ such that $v^* \in \{v_1, v_2, v_3\}$.*

Lemma 7. *If $\mathcal{T}$ is both valid and sparse, then* BuildGame($\mathcal{T}$) *is sparse.*

```
COPY_A($\mathcal{T}; v; v_1, v_2, ..., v_k$)
1: pick unused nodes $v'_1, v'_2, ..., v'_{k-1} \in V_A$ and $w', w'', w_1 ..., w_{k-1}, w'_1 ..., w'_{k-2} \in V_I$
2: INSERT($\mathcal{T}, (G_=, v, nil, nil, v_1, nil, w')$) and INSERT($\mathcal{T}, (G_=, v, nil, nil, v'_1, nil, w'')$)
3: for $i$ from 1 to $k-1$, INSERT($\mathcal{T}, (G_=, v'_i, nil, nil, v_{i+1}, nil, w_i)$)
4: for $i$ from 1 to $k-2$, INSERT($\mathcal{T}, (G_=, v'_i, nil, nil, v'_{i+1}, nil, w'_i)$)
```

Fig. 3. Function COPY$_A$

Proof. For each $T = (G, v_1, v_2, v_3, v, c, w) \in \mathcal{T}$, $(\mathbf{M}[T], \mathbf{N}[T])$ satisfies:

Property 1. *Let $k = \mathcal{C}_A(v)$, $t = \mathcal{C}_I(w)$ and $k_i = \mathcal{C}_A(v_i)$ for every $1 \le i \le 3$. In matrices $\mathbf{M}[T] = (M_{i,j})$ and $\mathbf{N}[T] = (N_{i,j})$, only the following entries are possibly nonzero: $\{M_{2k-1,2t-1}, M_{2k-1,2t}, M_{2k,2t-1}, M_{2k,2t}\}$ and $\{N_{2l-1,2t-1}, N_{2l-1,2t}, N_{2l,2t-1}, N_{2l,2t}$, where $l \in \{k_1, k_2, k_3, k\}\}$, and all these entries are in $[0, 1]$.*

Property 1 follows directly from the construction of $\mathbf{M}[T]$ and $\mathbf{N}[T]$ in Figure 2. Let $(\mathbf{A} = (a_{i,j}), \mathbf{B} = (b_{i,j})) = \text{BUILDGAME}(\mathcal{T})$.

Let v be an arithmetic node in V_A and $k = \mathcal{C}_A(v)$. According to Property 1, for any $1 \le j \le 2K$, $a_{2k,j} \ne a^*_{2k,j}$ implies that there exists a gadget $T \in \mathcal{T}$ whose output node is v and internal node w satisfies $j \in \{2\mathcal{C}_I(w), 2\mathcal{C}_I(w) - 1\}$. Since $\mathcal{T}$ is valid, there can be at most one gadget whose output node is v and thus, there are at most two integers $1 \le j \le 2K$ such that $a_{2k,j} \ne a^*_{2k,j}$. On the other hand, the $(2k)^{th}$ row of $\mathbf{A}^*$ has exactly two nonzero entries. As a result, the number of nonzero entries in the $(2k)^{th}$ row of $\mathbf{A}$ is at most four. The case for the $(2k-1)^{st}$ row can be proved similarly, and thus, $\mathbf{A}$ is row sparse. One can prove similarly that both $\mathbf{A}$ and $\mathbf{B}$ are column sparse.

Let v be an arithmetic node in V_A and $k = \mathcal{C}_A(v)$. According to Property 1, $b_{2k,j} \ne b^*_{2k,j}$ implies there exists a gadget $T = (G, v_1, v_2, v_3, v, c, w) \in \mathcal{T}$ such that $v \in \{v_1, v_2, v_3, v\}$ and $j \in \{2\mathcal{C}_I(w), 2\mathcal{C}_I(w) - 1\}$. Since $\mathcal{T}$ is both valid and sparse, there can be at most three gadgets $T = (G, v_1, v_2, v_3, v, c, w) \in \mathcal{T}$ such that $v \in \{v_1, v_2, v_3, v\}$, and at most six integers j such that $b_{2k,j} \ne b^*_{2k,j}$. So the number of nonzero entries in the $(2k)^{th}$ row of $\mathbf{B}$ is at most eight. The case for the $(2k-1)^{st}$ row can be proved similarly, and thus, $\mathbf{B}$ is row sparse.

4.2 The Copy Network

In this subsection, we build a network of gadgets which will be referred to as a *copy network*. Let us define some notations that will be useful.

Let $\mathcal{T}$ be a valid collection of gadgets. An arithmetic node $v \in V_A$ (or an internal node $w \in V_I$) is *unused* in $\mathcal{T}$ if none of the gadgets in $\mathcal{T}$ uses v (or w) as its output node (or internal node). We use UNUSED$[\mathcal{T}]$ to denote the number of unused nodes $v \in V_A$ in $\mathcal{T}$. Suppose $T \notin \mathcal{T}$ is a gadget such that $\mathcal{T} \cup \{T\}$ is still valid. We use INSERT$(\mathcal{T}, T)$ to denote the insertion of gadget T into $\mathcal{T}$.

```
1: set $\mathcal{T} = \emptyset$
2: for every gadget $T = (G, v_1, v_2, v, c, w) \in \mathcal{S}^U$ constructed in [7] do
3:     if $G \in \{G_{\times\zeta}, G_=, G_+, G_-, G_<, G_\neg\}$ then
4:         $\textsc{Insert}(\mathcal{T}, (G, v_1, v_2, nil, v, c, w))$
5:     else [ if $G = G_\zeta$ (or $G_\wedge, G_\vee$), we use $G^*$ to denote $G^*_\zeta$ (or $G^*_\wedge, G^*_\vee$) ]
6:         pick nodes $v' \in V_A$ and $w' \in V_I$, which are unused in both $\mathcal{S}^U$ and $\mathcal{T}$
7:         $\textsc{Insert}(\mathcal{T}, (G_H, nil, nil, nil, v', nil, w'))$, $\textsc{Insert}(\mathcal{T}, (G^*, v_1, v_2, v', v, c, w))$
```

Fig. 4. Step 1: from $\mathcal{S}^U$ to $\mathcal{T}$

```
1: set $\mathcal{T}^U = \mathcal{T}$
2: for every $v \in V_A$ which is used by $k > 2$ gadgets in $\mathcal{T}^U$ as their input nodes do
3:     suppose these gadgets are $T_1, T_2, ..., T_k \in \mathcal{T}^U$
4:     pick $k$ nodes $v_1, v_2, ..., v_k \in V_A$ which are unused in $\mathcal{T}^U$
5:     for every $1 \le i \le k$, replace the $v$ in $T_i \in \mathcal{T}^U$ by $v_i$
6:     if we intend to store a boolean value in $v$ (which should be clear from [7])
7:         $\textsc{Copy}_B(\mathcal{T}^U; v; v_1, v_2, ..., v_k)$
8:     else
9:         $\textsc{Copy}_A(\mathcal{T}^U; v; v_1, v_2, ..., v_k)$
```

Fig. 5. Step 2: from $\mathcal{T}$ to $\mathcal{T}^U$

Let $\mathcal{T}$ be a valid collection with $\textsc{Unused}[\mathcal{T}] \ge 2k-1$, and $k \ge 3$. Let $v \in V_A$, and $v_1, v_2, ..., v_k \in V_A$ be k unused nodes in $\mathcal{T}$. We insert $2k-1$ gadgets into $\mathcal{T}$ by invoking the function $\textsc{Copy}_A(\mathcal{T}; v; v_1, v_2, ..., v_k)$ in Figure 3. We let $\mathcal{T}'$ denote the collection $\mathcal{T}$ after executing $\textsc{Copy}_A(\mathcal{T}; v; v_1, v_2, ..., v_k)$, then

Lemma 8. *In every ϵ-well-supported Nash equilibrium $(\mathbf{x}, \mathbf{y})$ of bimatrix game $\textsc{BuildGame}(\mathcal{T}')$, $\mathbf{x}[v_i] = \mathbf{x}[v] \pm 3t\epsilon$ for all $1 \le t \le k$.*

Furthermore, by replacing every $G_=$ gadget in $\textsc{Copy}_A$ with a $G_{B=}$ gadget, we immediately get a function $\textsc{Copy}_B(\mathcal{T}; v; v_1, v_2 ... v_k)$ for inserting a boolean copy network into $\mathcal{T}$, such that

Lemma 9. *In every ϵ-well-supported Nash equilibrium $(\mathbf{x}, \mathbf{y})$ of bimatrix game $\textsc{BuildGame}(\mathcal{T}')$, if $\mathbf{x}[v] =_B b$ where $b \in \{0, 1\}$, then $\mathbf{x}[v_t] =_B b$, $\forall\ 1 \le t \le k$.*

4.3 Construction of $\mathcal{T}^U$ and $\mathcal{H}^U$

Let $U = (C, 0^{3n})$ be an input instance of search problem $\textsc{Brouwer}^f$, and $\mathcal{S}^U$ be the collection of gadgets constructed in [7]. We now convert it into a new collection $\mathcal{T}^U$ that is both valid and sparse, such that $\mathcal{H}^U = \textsc{BuildGame}(\mathcal{T}^U)$

satisfies both **Property** $\mathbf{P}_1$ and $\mathbf{P}_2$. Notice that, since $\mathcal{T}^U$ is valid and sparse, game $\mathcal{H}^U$ is sparse by Lemma 7. We build $\mathcal{T}^U$ with a two-step construction:

Step 1 [Figure 4]. We build a collection $\mathcal{T}$ by replacing each $G_\zeta, G_\wedge$ and $G_\vee$ gadget in $\mathcal{S}^U$ with two gadgets: one G_H gadget and one $G_\zeta^*, G_\wedge^*$ or $G_\vee^*$ gadget. Note that, every $G_\zeta^*, G_\wedge^*$ or $G_\vee^*$ gadget in $\mathcal{T}$ has a "private" G_H gadget.
Step 2 [Figure 5]. For every $v \in V_A$ which is used by $k > 2$ gadgets in $\mathcal{T}$ as one of their input nodes, we pick k unused nodes $v_1, ..., v_k$ in V_A and insert a copy network to connect v with $v_1, ..., v_k$. Then, each of the k gadgets gets one node in $\{v_1, v_2, ..., v_k\}$ as its input node.

4.4 Correctness of the Reduction

The main item we need to check carefully is the number of nodes used in $\mathcal{T}^U$. The number of nodes used in $\mathcal{S}^U$ is $O(\text{Size}\,[C]^4)$ [7]. Thus the number of nodes used in $\mathcal{T}$ is also $O(\text{Size}\,[C]^4)$. On the other hand, every $T \in \mathcal{T}$ can appear in line 3 of Figure 5 for at most three times, since T only has three input nodes. The number of nodes used in $\mathcal{T}^U$ is still $O(\text{Size}\,[C]^4) \ll K$. So we always have $\text{UNUSED}[\mathcal{T}] > 0$ and $\text{UNUSED}[\mathcal{T}^U] > 0$, during the construction of $\mathcal{T}$ and $\mathcal{T}^U$.

Because $\mathcal{S}^U$ is valid, one can check that $\mathcal{T}^U$ is both valid and sparse. As a corollary of Lemma 5, $\mathcal{H}^U$ satisfies **Property** $\mathbf{P}_1$. Following the line of proof in [7], we can show that, with the same procedure used in [7], one can recover a panchromatic simplex of C from every ϵ-well-supported Nash equilibrium of two-player game $\mathcal{H}^U$. Therefore, $\mathcal{H}^U$ also satisfies **Property** $\mathbf{P}_2$.

By Lemma 7, we know that game $\mathcal{H}^U$ is sparse. Finally, we get a reduction from problem BROUWER^f to SPARSE BIMATRIX, and Theorem 1 is proven.

5 An Algorithm for Very Sparse Win-Lose Games

In this section, we describe an algorithm for finding exact Nash equilibria in a class of very sparse win-lose games.

A bimatrix game $\mathcal{G} = (\mathbf{A}, \mathbf{B})$ is a *win-lose game* if every entry of $\mathbf{A}$ and $\mathbf{B}$ is either 0 or 1. A win-lose game $\mathcal{G} = (\mathbf{A}, \mathbf{B})$ is *very sparse* if in each row of $\mathbf{A}$ and each column of $\mathbf{B}$, there are at most *two* non-zero entries. We use $\mathcal{P}$ to denote the set of very sparse win-lose games. First we define a subclass $\mathcal{Q}$ of $\mathcal{P}$. Every game in $\mathcal{Q}$ has an exact Nash equilibrium that can be computed easily.

Definition 4. *Let* $\mathbf{A}$ *be a* $\{0,1\}$*-matrix. The row* i *of* $\mathbf{A}$ *is said to be dominated if one of the following conditions is true:* **1).** *all the entries in it are zero;* **2).** *only one entry* $a_{i,j} = 1$ *is non-zero, and there exists another* $i' \neq i$ *such that* $a_{i',j} = 1$. *Similarly, the column* j *of matrix* $\mathbf{B}$ *is dominated if the row* j *of* $\mathbf{B}^T$ *is dominated. A bimatrix game* $\mathcal{G} = (\mathbf{A}, \mathbf{B}) \in \mathcal{P}$ *belongs to* $\mathcal{Q}$ *if none of the rows of* $\mathbf{A}$ *is dominated, and none of the columns of* $\mathbf{B}$ *is dominated.*

For every game $\mathcal{G} = (\mathbf{A}, \mathbf{B}) \in \mathcal{Q}$ where $\mathbf{A}$ and $\mathbf{B}$ are $n \times m$ matrices, we build a pair of vectors $(\mathbf{x}^* \in \mathbb{R}^n, \mathbf{y}^* \in \mathbb{R}^m)$ as follows: **1)** For each $1 \leq j \leq m$, if there

```
SparseWinLose ( $\mathcal{G} = (\mathbf{A}, \mathbf{B}) \in \mathcal{P}$ )
 1: if $n = 1$ or $m = 1$ then
 2:     output a Nash equilibrium of $\mathcal{G}$
 3: else if $\mathcal{G} \in \mathcal{Q}$ then
 4:     output a Nash equilibrium of $\mathcal{G}$ using Lemma 10
 5: else if the row $i$ of $\mathbf{A}$ is dominated then
 6:     $(\mathbf{x}', \mathbf{y}') =$ SparseWinLose ( $\mathcal{G}'$ ), $\mathcal{G}'$ is obtained by deleting row $i$ from $\mathcal{G}$
 7:     output a Nash equilibrium of $\mathcal{G}$ using Lemma 11
 8: else [ assume the column $j$ of $\mathbf{B}$ is dominated ]
 9:     $(\mathbf{x}', \mathbf{y}') =$ SparseWinLose ( $\mathcal{G}'$ ), $\mathcal{G}'$ is obtained by deleting column $j$ from $\mathcal{G}$
10:     output a Nash equilibrium of $\mathcal{G}$ using Lemma 12
```

Fig. 6. An Algorithm for Very Sparse Win-Lose Games

exists an $1 \leq i \leq n$ such that the row i of $\mathbf{A}$ has exactly one non-zero entry: $a_{i,j} = 1$, then $y_j^* = 2$, otherwise $y_j^* = 1$; **2)** For each $1 \leq i \leq n$, if there is an $1 \leq j \leq m$ such that the column j of matrix $\mathbf{B}$ has exactly one nonzero entry: $b_{i,j} = 1$, then $x_i^* = 2$, otherwise $x_i^* = 1$.

Lemma 10. *For every $\mathcal{G} = (\mathbf{A}, \mathbf{B}) \in \mathcal{Q}$, let $(\mathbf{x}^* \in \mathbb{R}^n, \mathbf{y}^* \in \mathbb{R}^m)$ be the pair of vectors constructed above, then $(\mathbf{x}, \mathbf{y})$ is a Nash equilibrium of $\mathcal{G}$ where*

$$x_i = x_i^* \Big/ \sum_{1 \leq k \leq n} x_k^* \quad \textit{and} \quad y_j = y_j^* \Big/ \sum_{1 \leq k \leq m} y_k^*, \quad \forall\ 1 \leq i \leq n,\ 1 \leq j \leq m.$$

Proof. From the definition of $\mathcal{Q}$, one can check that, for every $1 \leq i \leq n$ and $1 \leq j \leq m$, $\langle \mathbf{A}_i | \mathbf{y} \rangle = 2 / \sum_{1 \leq k \leq m} y_k^*$ and $\langle \mathbf{x} | \mathbf{B}_j \rangle = 2 / \sum_{1 \leq k \leq n} x_k^*$, where $\mathbf{A}_i$ denotes the i^{th} row vector of matrix $\mathbf{A}$, and $\mathbf{B}_j$ denotes the j^{th} column vector of $\mathbf{B}$. This implies that $(\mathbf{x}, \mathbf{y})$ is an exact Nash equilibrium of game $\mathcal{G}$.

Our algorithm is recursive. If the input game is small ($n = 1$ or $m = 1$) or belongs to $\mathcal{Q}$, then a Nash equilibrium can be found easily. Otherwise, we delete one row or column from $\mathcal{G}$, and obtain a smaller game $\mathcal{G}'$. From every Nash equilibrium $(\mathbf{x}', \mathbf{y}')$ of the new game $\mathcal{G}'$, one can "recover" a solution to the old one quickly. The algorithm is supported by the following two lemmas. Here we use $\mathcal{G} = (\mathbf{A}, \mathbf{B})$ to denote a game in $\mathcal{P}$, where $\mathbf{A}$ and $\mathbf{B}$ are $n \times m$ matrices.

Lemma 11. *If the row k of $\mathbf{A}$ is dominated, letting $(\mathbf{x}' \in \mathbb{P}^{n-1}, \mathbf{y}' \in \mathbb{P}^m)$ be an exact Nash equilibrium of $\mathcal{G}'$, where $\mathcal{G}'$ is obtained by deleting row k from game $\mathcal{G}$, then $(\mathbf{x}, \mathbf{y})$ is a Nash equilibrium of $\mathcal{G}$, where $\mathbf{y} = \mathbf{y}'$, $x_k = 0$, $x_i = x_i'$ for all $1 \leq i < k$, and $x_i = x_{i-1}'$ for all $k < i \leq n$.*

Lemma 12. *If the column k of $\mathbf{B}$ is dominated, letting $(\mathbf{x}' \in \mathbb{P}^n, \mathbf{y}' \in \mathbb{P}^{m-1})$ be a Nash equilibrium of $\mathcal{G}'$, where $\mathcal{G}'$ is obtained by deleting column k from game*

$\mathcal{G}$, then $(\mathbf{x}, \mathbf{y})$ is a Nash equilibrium of $\mathcal{G}$, where $\mathbf{x} = \mathbf{x}'$, $y_k = 0$, $y_i = y'_i$ for all $1 \leq i < k$, and $y_i = y'_{i-1}$ for all $k < i \leq m$.

Acknowledgements

We thank Li-Sha Huang for a discussion on market equilibria which led us to study this problem. Part of Teng's work was done while visiting Microsoft Beijing Research Lab. His work is supported by the NSF grants CCR-0311430 and ITR CCR-0325630. Part of Deng's work was done while visiting Tsinghua University.

References

1. Goldberg, P., Papadimitriou, C.: Reducibility among equilibrium problems. (In: STOC 2006)
2. Daskalakis, C., Goldberg, P., Papadimitriou, C.: The complexity of computing a nash equilibrium. (In: STOC 2006)
3. Papadimitriou, C.: On the complexity of the parity argument and other inefficient proofs of existence. Journal of Computer and System Sciences (1994) 498–532
4. Chen, X., Deng, X.: 3-nash is ppad-complete. ECCC, TR05-134 (2005)
5. Daskalakis, C., Papadimitriou, C.: Three-player games are hard. ECCC, TR05-139 (2005)
6. Chen, X., Deng, X.: Settling the complexity of two-player nash-equilibrium. (In: FOCS 2006)
7. Chen, X., Deng, X., Teng, S.H.: Computing nash equilibria: Approximation and smoothed complexity. (In: FOCS 2006)
8. Lipton, R.J., Markakis, E., Mehta, A.: Playing large games using simple strategies. (In: ACM EC 2003) 36–41
9. Spielman, D.A., Teng, S.H.: Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. J. ACM **51** (2004) 385–463
10. Barany, I., Vempala, S., Vetta, A.: Nash equilibria in random games. (In: FOCS 2005)
11. Chen, X., Teng, S.H., Valiant, P.A.: The approximation complexity of win-lose games. Manuscript: Tsinghua-BU-MIT (2006)
12. Nash, J.: Equilibrium point in n-person games. Porceedings of the National Academy of the USA **36** (1950) 48–49
13. Codenotti, B., Leoncini, M., Resta, G.: Efficient computation of nash equilibria for very sparse win-lose bimatrix games. (In: ESA 2006)