Theoretical Computer Science 410 (2009) 4448–4456

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

On the complexity of 2D discrete fixed point problem

Xi Chen [a], Xiaotie Deng [b,*]

[a] Department of Computer Science, Tsinghua University, Beijing, PR China

[b] Department of Computer Science, City University of Hong Kong, Hong Kong SAR, PR China

A B S T R A C T

We study a computational complexity version of the 2D Sperner problem, which states that any three coloring of vertices of a triangulated triangle, satisfying some boundary conditions, will have a trichromatic triangle. In introducing a complexity class **PPAD**, Papadimitriou [C.H. Papadimitriou, On graph-theoretic lemmata and complexity classes, in: Proceedings of the 31st Annual Symposium on Foundations of Computer Science, 1990, 794–801] proved that its 3D analogue is **PPAD**-complete about fifteen years ago. The complexity of **2D-SPERNER** itself has remained open since then.

We settle this open problem with a **PPAD**-completeness proof. The result also allows us to derive the computational complexity characterization of a discrete version of the 2D Brouwer fixed point problem, improving a recent result of Daskalakis, Goldberg and Papadimitriou [C. Daskalakis, P.W. Goldberg, C.H. Papadimitriou, The complexity of computing a Nash equilibrium, in: Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC), 2006]. Those hardness results for the simplest version of those problems provide very useful tools to the study of other important problems in the **PPAD** class.

1. Introduction

The classical lemma of Sperner [10], characterizing the combinatorial nature of Brouwer's fixed point theorem, states that any admissible 3-coloring of any triangulation of a triangle has a trichromatic triangle. Naturally, it defines a search problem **2D-SPERNER** of finding such a triangle in an admissible 3-coloring for an exponential size triangulation, typical of problems in **PPAD**, a complexity class introduced by Papadimitriou to capture mathematical structures with the path-following proof technique [9]. Many important problems, such as the Brouwer fixed point, the search versions of Smith's theorem, as well as the Borsuk–Ulam theorem, belong to this class [9].

The computational complexity issue for those problems is of interest only when the search space is exponential in the input parameter. For problem **2D-SPERNER** as an example, with an input parameter n, we consider a right angled triangle with a side length $N = 2^n$. Its triangulation is into right angled triangles of side length one (see Fig. 1 for an example). There is a (3-coloring) function which, given any vertex in the triangulation, outputs its color in the coloring. The color function is guaranteed to be admissible and is given by a polynomial-time Turing machine. The problem is to find a triangle that has all three colors on its vertices. The 3D analogue **3D-SPERNER** is the first natural problem proved to be complete in **PPAD** [9]. Whether the 2D case is complete or not was left as an open problem. Since then, progress has been made toward the solution of this problem: In [6], Grigni defined a non-oriented version of **3D-SPERNER** and proved that it is **PPA**-complete. Friedl, Ivanyos, Santha, and Verhoeven showed [5,4] that the locally 2D case of Sperner's problem is complete in **PPAD**. Despite those efforts, the original 2D Sperner's problem has remained elusive.

* Corresponding author.
E-mail addresses: csxichen@gmail.com (X. Chen), deng@cs.cityu.edu.hk (X. Deng).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.07.052

In this paper, we prove that **2D-SPERNER** is **PPAD**-complete and thus settle the open problem proposed by Papadimitriou [8] fifteen years ago. Furthermore, this result also allows us to derive the **PPAD**-completeness proof of a discrete version of the 2D fixed point problem (**2D-BROUWER**). Our study is motivated by the complexity results in [1] and [7] for finding a discrete Brouwer fixed point in d-dimensional space with a function oracle. The combinatorial structure there is similar to the one here. It was proven in the oracle model that, for any $d \geq 2$, the fixed point problem unconditionally requires an exponential number (in consistency with d) of queries. Although the computational models in these two problems are different, we moved into the direction of a hardness proof, expecting that the complexity hierarchy in Sperner's problem may have a similar structure with respect to the dimension.

The class **PPAD** is the set of problems that are polynomial-time reducible to the problem called **LEAFD** [9]. It considers a directed graph of an exponential number, in the input parameter n, of vertices, numbered from 0 to $N-1$ where $N = 2^n$. Each vertex has at most one incoming edge and at most one outgoing edge. There is a distinguished vertex, 0, which has no incoming edge and has one outgoing edge. The required output is another vertex for which the sum of its incoming degree and outgoing degree is one. To access the directed graph, we have a polynomial-time Turing machine which, given any vertex as an input, outputs its possible predecessor and successor. In examination into the **PPAD**-completeness proof of problem **3D-SPERNER**, we found that the main idea is to embed complete graphs in 3D search spaces [9]. Such an embedding, obviously impossible in the plane, would allow us to transform any Turing machine which generates a directed graph in **LEAFD** to a Turing machine which produces an admissible coloring on a 3D search space of **3D-SPERNER**.

We take a different approach for the proof which can be clearly divided into two steps. First, we define a new search problem called **RLEAFD** (restricted-**LEAFD**). While the input graph has the same property as those in problem **LEAFD** (that is, both the incoming degree and outgoing degree of every vertex are at most one), it is guaranteed to be a sub-graph of some predefined planar grid graph. Note that even though any input to **LEAFD** is planar, it is not easy to do it within polynomial time. The interesting result obtained is that, even with such a strong restriction, the problem is still complete in **PPAD**. In the second step, we reduce **RLEAFD** to **2D-SPERNER** and prove that the latter is also complete. The main idea represents an improved understanding of **PPAD** reductions and may be of general applicability in related problems.

The completeness result of **2D-SPERNER** allows us to deduce that a discrete version of the 2D Brouwer fixed point problem is also **PPAD**-complete. In the problem, we consider a function g on a 2D grid such that, for every point $\mathbf{p}$ in the grid, $g(\mathbf{p})$ is equal to $\mathbf{p}$ plus an incremental vector with only three possible values: $(1, 0)$, $(0, 1)$ and $(-1, -1)$. A fixed point is a set of four corners of an orthogonal unit square such that incremental vectors at those point include all the three possibilities, an analogue to that of the 3D case introduced in [3]. Such a definition of fixed points, which is different from the original Brouwer fixed point but is related to Sperner's lemma, has a natural connection with approximation [7], and is consistent in spirit with the recent algorithmic studies on discrete fixed points [1]. On a first look at the problem, its natural link to the 2D Sperner problem is only in one direction. We overcome the difficulty in the other direction to show the reduction is indeed complete.

The **PPAD**-completeness of both **2D-SPERNER** and **2D-BROUWER**, in their simplicities, can serve as better benchmarks as well as provide the much needed intuition to derive completeness proofs for complicated problems, such as in the subsequent result of non-approximability (and also the smoothed complexity) of the two-player game Nash Equilibrium problem [2]. In particular, an important key lemma in the non-approximability result is a **PPAD**-completeness proof of a discrete fixed point problem on high-dimensional hypergrids with a constant side length, which can be most conveniently derived from the completeness of the 2D discrete fixed point problem.

2. Preliminaries

2.1. TFNP and PPAD

Definition 1 (*TFNP*). Let $R \subset \{0,1\}^* \times \{0,1\}^*$ be a polynomial-time computable, polynomially balanced relation (that is, there exists a polynomial $p(n)$ such that for every pair $(x, y) \in R$, $|y| \leq p(|x|)$). The **NP** search problem Q_R specified by R is this: given an input $x \in \{0,1\}^*$, return a string $y \in \{0,1\}^*$ such that $(x, y) \in R$, if such a y exists, and return the string "no" otherwise.

An **NP** search problem Q_R is said to be *total* if for every string $x \in \{0,1\}^*$, there exists a $y \in \{0,1\}^*$ such that $(x, y) \in R$. We use **TFNP** to denote the class of total **NP** search problems.

An **NP** search problem $Q_{R_1} \in$ **TFNP** is *polynomial-time reducible* to problem $Q_{R_2} \in$ **TFNP** if there exists a pair of polynomial-time computable functions (f, g) such that, for every input x of Q_{R_1}, if y satisfies $(f(x), y) \in R_2$, then $(x, g(y)) \in R_1$. We now define a total **NP** search problem called **LEAFD** [9].

Definition 2 (*LEAFD*). The input of the problem is a pair $(M, 0^k)$ where M is the description of a polynomial-time Turing machine which satisfies: **(1)** for any $v \in \{0,1\}^k$, $M(v)$ is an ordered pair (u_1, u_2) where $u_1, u_2 \in \{0,1\}^k \cup \{\text{no}\}$; **(2)** $M(0^k) = (no, 1^k)$ and the first component of $M(1^k)$ is 0^k. M generates a directed graph $G = (V, E)$ where $V = \{0,1\}^k$. An edge uv appears in E iff v is the second component of $M(u)$ and u is the first component of $M(v)$.

The output is a directed leaf (with in-degree + out-degree $= 1$) other than 0^k.

PPAD [8] is the set of total **NP** search problems that are polynomial-time reducible to **LEAFD**. From its definition, **LEAFD** is complete for **PPAD**.

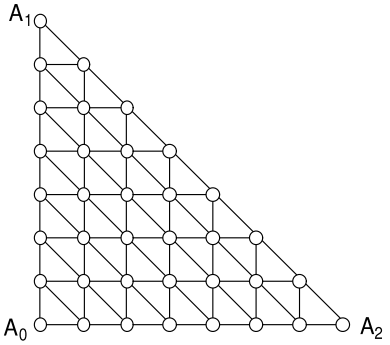

Fig. 1. The standard 7×7 triangulation of a triangle.

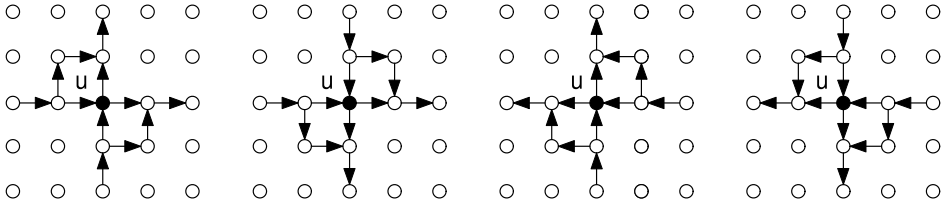

Fig. 2. Summary of cases in the construction of E_n^2.

2.2. Definition of 2D-SPERNER

One of the most interesting problems in **PPAD** is **2D-SPERNER** whose totality is based on Sperner's Lemma [10]: any admissible 3-coloring of any triangulation of a triangle has a trichromatic triangle.

In problem **2D-SPERNER**, we consider the standard $n \times n$ triangulation of a triangle which is illustrated in Fig. 1. Every vertex in the triangulation corresponds to a point in $\mathbb{Z}^2$. Here $A_0 = (0, 0), A_1 = (0, n)$ and $A_2 = (n, 0)$ are the three vertices of the original triangle. The vertex set T_n of the triangulation is defined as $T_n = \{\mathbf{p} \in \mathbb{Z}^2 \mid p_1 \geq 0,\ p_2 \geq 0,\ p_1 + p_2 \leq n\}$. A 3-coloring of the $n \times n$ triangulation is a function f from T_n to $\{0, 1, 2\}$. It is said to be admissible if

1. $f(A_i) = i$, for all $0 \leq i \leq 2$;
2. for every point $\mathbf{p}$ on segment A_iA_j, $f(\mathbf{p}) \neq 3 - i - j$.

A unit size well-oriented triangle is a triple $\Delta = (\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^2)$ where $\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^2 \in \mathbb{Z}^2$. It satisfies either $\mathbf{p}^1 = \mathbf{p}^0 + \mathbf{e}_1$, $\mathbf{p}^2 = \mathbf{p}^0 + \mathbf{e}_2$ or $\mathbf{p}^1 = \mathbf{p}^0 - \mathbf{e}_1, \mathbf{p}^1 = \mathbf{p}^0 - \mathbf{e}_2$, where $\mathbf{e}_1 = (0, 1)$ and $\mathbf{e}_2 = (1, 0)$ are the unit vectors. In other words, the triangle has a northwest oriented hypotenuse. We use S to denote the set of all such triangles.

From Sperner's Lemma, we define problem **2D-SPERNER** as follows.

Definition 3 (*2D-SPERNER [8]*). The input instance is a pair $(F, 0^k)$ where F is a polynomial-time Turing machine which produces an admissible 3-coloring f on T_{2^k}. Here $f(\mathbf{p}) = F(\mathbf{p}) \in \{0, 1, 2\}$, for every vertex $\mathbf{p} \in T_{2^k}$.

The output is a trichromatic triangle $\Delta \in S$ of coloring f.

In [8], it was shown that **2D-SPERNER** is in **PPAD**. They also defined a 3D analogue **3D-SPERNER** of **2D-SPERNER**, and proved that it is **PPAD**-complete. The completeness of the 2D case was left as an open problem.

3. Definition of search problem RLEAFD

Before the definition of **RLEAFD**, we describe a class of planar grid graphs $\{G_i\}_{i \geq 1}$, where $G_n = (V_n, E_n)$ and vertex set

$$V_n = \left\{\mathbf{u} \in \mathbb{Z}^2 | 0 \leq u_1 \leq 3(n^2 - 2), 0 \leq u_2 \leq 3(2n - 1)\right\}.$$

Informally speaking, G_n is a planar embedding of the complete graph K_n with vertex set $\{0, 1 \ldots n - 1\}$. For every $0 \leq i < n$, vertex i of K_n corresponds to the vertex $(0, 6i)$ of G_n. For every edge $ij \in K_n$, we define a path E_{ij} from vertex $(0, 6i)$ to $(0, 6j)$. To obtain the edge set E_n of G_n, we start from an empty graph $(V_n, \emptyset)$, and then add all the paths E_{ij}. There are $O(n^2)$ vertices in V_n, which are at the intersection of two paths added previously. Since K_n is not a planar graph when $n \geq 5$, there is no embedding which can avoid those crossing points. For each of those crossing points, we add four more edges into E_n.

We define E_n formally as follows. E_n can be divided into two parts: E_n^1 and E_n^2 such that $E_n = E_n^1 \cup E_n^2$ and $E_n^1 \cap E_n^2 = \emptyset$. The first part $E_n^1 = \cup_{ij \in K_n} E_{ij}$ and path E_{ij} is defined as follows.

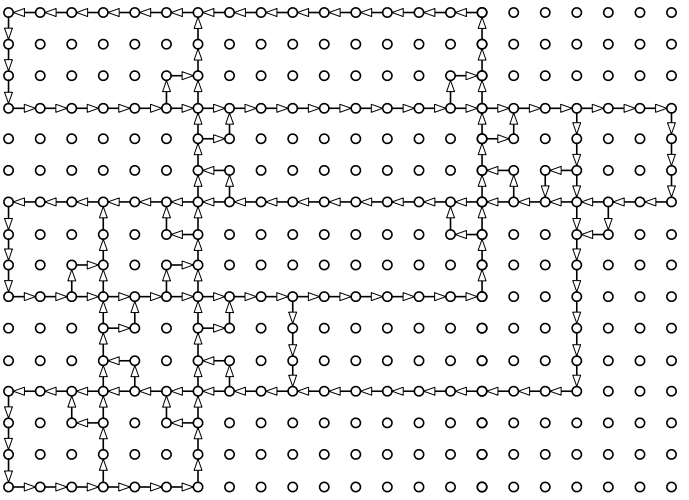

Fig. 3. The planar grid graph G_3.

Definition 4. Let $\mathbf{p}^1, \mathbf{p}^2 \in \mathbb{Z}^2$ be two points with the same x-coordinate or the same y-coordinate. Let $\mathbf{u}^1, \mathbf{u}^2 \ldots \mathbf{u}^m \in \mathbb{Z}^2$ be all the integral points on segment $\mathbf{p}^1\mathbf{p}^2$ which are labeled along the direction of $\mathbf{p}^1\mathbf{p}^2$. We use $E(\mathbf{p}^1\mathbf{p}^2)$ to denote the path which consists of $m-1$ directed edges: $\mathbf{u}^1\mathbf{u}^2, \mathbf{u}^2\mathbf{u}^3, \ldots \mathbf{u}^{m-1}\mathbf{u}^m$.

Definition 5. For every edge $ij \in K_n$ with $0 \le i \ne j < n$, we define a path E_{ij} as $E(\mathbf{p}^1\mathbf{p}^2) \cup E(\mathbf{p}^2\mathbf{p}^3) \cup E(\mathbf{p}^3\mathbf{p}^4) \cup E(\mathbf{p}^4\mathbf{p}^5)$, where $\mathbf{p}^1 = (0, 6i)$, $\mathbf{p}^2 = (3(ni+j), 6i)$, $\mathbf{p}^3 = (3(ni+j), 6j+3)$, $\mathbf{p}^4 = (0, 6j+3)$ and $\mathbf{p}^5 = (0, 6j)$.

One can show that, every vertex in V_n has at most 4 edges (including both incoming and outgoing edges) in E_n^1. Moreover, if $\mathbf{u}$ has 4 edges, then $3|u_1$ and $3|u_2$. We now use $\{\mathbf{u}^i\}_{1\le i\le 8}$ to denote the eight vertices around $\mathbf{u}$. For each $1 \le i \le 8$, $\mathbf{u}^i = \mathbf{u} + \mathbf{x}^i$ where $\mathbf{x}^1 = (-1, 1)$, $\mathbf{x}^2 = (0, 1)$, $\mathbf{x}^3 = (1, 1)$, $\mathbf{x}^4 = (1, 0)$, $\mathbf{x}^5 = (1, -1)$, $\mathbf{x}^6 = (0, -1)$, $\mathbf{x}^7 = (-1, -1)$ and $\mathbf{x}^8 = (-1, 0)$. If $\mathbf{u} \in V_n$ has 4 edges in E_n^1, then it must satisfy the following two properties:

1. either edges $\mathbf{u}^4\mathbf{u}, \mathbf{u}\mathbf{u}^8 \in E_n^1$ or $\mathbf{u}^8\mathbf{u}, \mathbf{u}\mathbf{u}^4 \in E_n^1$;
2. either edges $\mathbf{u}^2\mathbf{u}, \mathbf{u}\mathbf{u}^6 \in E_n^1$ or $\mathbf{u}^6\mathbf{u}, \mathbf{u}\mathbf{u}^2 \in E_n^1$.

Now for every vertex $\mathbf{u} \in V_n$ which has four edges in E_n^1, we add four more edges into E_n. For example, if $\mathbf{u}^4\mathbf{u}, \mathbf{u}\mathbf{u}^8, \mathbf{u}^2\mathbf{u}, \mathbf{u}\mathbf{u}^6 \in E_n^1$ (that is, the last case in Fig. 2), then $\mathbf{u}^4\mathbf{u}^5, \mathbf{u}^5\mathbf{u}^6, \mathbf{u}^2\mathbf{u}^1, \mathbf{u}^1\mathbf{u}^8 \in E_n^2$. All the four possible cases are summarized in Fig. 2.

An example (graph G_3) is showed in Fig. 3. We can draw it in two steps. In the first step, for each $ij \in K_3$, we add path E_{ij} into the empty graph. In the second step, we search for vertices of degree four. For each of them, 4 edges are added according to Fig. 2. One can prove the following property of G_n.

Lemma 1. *Every vertex in G_n has at most 4 edges. There is a polynomial-time Turing machine M^* such that, for every input instance $(n, \mathbf{u})$ where $\mathbf{u} \in V_n$, it outputs all the predecessors and successors of vertex $\mathbf{u}$ in graph G_n.*

We use C_n to denote the set of graphs $G = (V_n, E)$ such that $E \subset E_n$ and for every $\mathbf{u} \in V_n$, both of its in-degree and out-degree are no more than one.

The new search problem **RLEAFD** is similar to **LEAFD**. The only difference is that, in **RLEAFD**, the directed graph G generated by the input pair $(K, 0^k)$ always belongs to C_{2^k}. By Lemma 1, one can prove that **RLEAFD** $\in$ **PPAD**.

Definition 6 (*RLEAFD*). The input instance is a pair $(K, 0^k)$ where K is the description of a polynomial-time Turing machine which satisfies: **(1)** for every vertex $\mathbf{u} \in V_{2^k}$, $K(\mathbf{u})$ is an ordered pair $(\mathbf{u}^1, \mathbf{u}^2)$ where $\mathbf{u}^1, \mathbf{u}^2 \in V_{2^k} \cup \{\text{no}\}$; **(2)** $K(0, 0) = (no, (1, 0))$ and the first component of $K(1, 0)$ is $(0, 0)$. K generates a directed graph $G = (V_{2^k}, E) \in C_{2^k}$. An edge $\mathbf{uv}$ appears in E iff $\mathbf{v}$ is the second component of $K(\mathbf{u})$, $\mathbf{u}$ is the first component of $K(\mathbf{v})$ and edge $\mathbf{uv} \in E_{2^k}$.

The output of the problem is a directed leaf other than $(0, 0)$.

4. Problem RLEAFD is PPAD-Complete

In this section, we will describe a reduction from **LEAFD** to **RLEAFD** and prove the latter is also complete in **PPAD**.

Let G be a directed graph with vertex set $\{0, 1 \ldots n-1\}$ which satisfies that the in-degree and out-degree of every vertex are at most one. We now build a graph $C(G) \in C_n$ in two steps. An important observation here is that $C(G)$ is not a planar embedding of G, as the structure of G is mutated dramatically in $C(G)$. However, it preserves the leaf nodes of G and does not create any new leaf node. Graph $C(G)$ is constructed as follows.

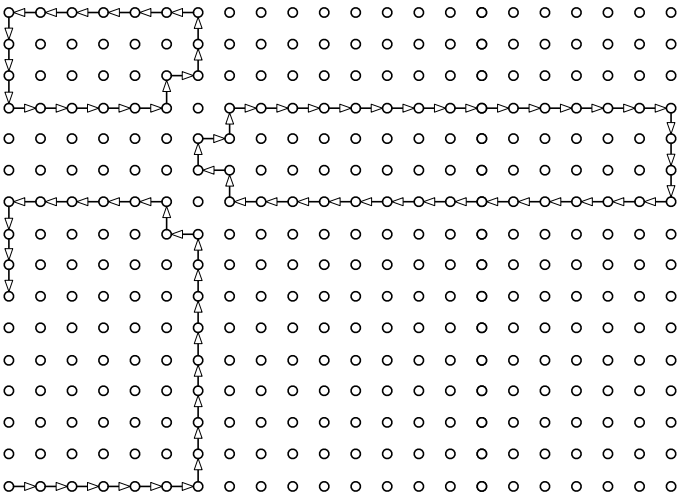

Fig. 4. Graph $C(G) \in C_3$.

1. Starting from an empty graph $(V_n, \emptyset)$, for every $ij \in G$, add path E_{ij};
2. For every $\mathbf{u} \in V_n$ of degree 4, remove all the four edges which have $\mathbf{u}$ as an endpoint and add four edges around $\mathbf{u}$ using Fig. 2.

One can check that, for each vertex in graph $C(G)$, both of its in-degree and out-degree are no more than one, and thus, we have $C(G) \in C_n$. For example, Fig. 4 shows $C(G)$ where $G = (\{0, 1, 2\}, \{02, 21\})$. The following lemma is easy to check.

Lemma 2. *Let G be a directed graph with vertex set $\{0, \ldots n-1\}$ which satisfies that the in-degree and out-degree of every vertex are at most one. For every vertex $0 \le k \le n-1$ of G, it is a directed leaf of G iff $\mathbf{u} = (0, 6k) \in V_n$ is a directed leaf of $C(G)$. On the other hand, if $\mathbf{u} \in V_n$ is a directed leaf of $C(G)$, then $u_1 = 0$ and $6|u_2$.*

Lemma 3. *Search problem* **RLEAFD** *is* **PPAD**-*complete.*

Proof. Let $(M, 0^k)$ be an input instance of **LEAFD** and G be the directed graph specified by M. It is tedious, but not hard, to show that one can build a Turing machine K from M in polynomial time, such that

1. K satisfies all the conditions in Definition 6; and
2. As an input of **RLEAFD**, the graph generated by K is exactly $C(G) \in C_{2^k}$.

On the other hand, Lemma 2 shows that, given any directed leaf of $C(G)$, we can locate a directed leaf of G easily. In this way, we get a reduction from **LEAFD** to **RLEAFD** and the lemma follows. □

5. 2D-SPERNER is PPAD-Complete

In this section, we present a polynomial-time reduction from **RLEAFD** to **2D-SPERNER** and finish the completeness proof of **2D-SPERNER**.

Let $(K, 0^k)$ be an input instance of **RLEAFD** and $G \in C_{2^k}$ be the directed graph generated by K. We will build a polynomial-time Turing machine F that defines an admissible 3-coloring on $T_{2^{2k+5}}$. Given a trichromatic triangle $\Delta \in S$ of F, a directed leaf of G can be found easily. To clarify the presentation here, we use $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ to denote vertices in V_{2^k}, and $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$ to denote vertices in $T_{2^{2k+5}}$.

To construct F, we first define a mapping $\mathcal{F}$ from V_{2^k} to $T_{2^{2k+5}}$. Since $G \in C_{2^k}$, its edge set can be uniquely decomposed into a collection of paths and cycles $P_1, P_2, \ldots P_m$. By using $\mathcal{F}$, every P_i is mapped to a set $I(P_i) \subset T_{2^{2k+5}}$. Only vertices in $I(P_i)$ have color 0 (with several exceptions around A_0). All the other vertices in $T_{2^{2k+5}}$ are colored carefully with either 1 or 2. Let $\Delta \in S$ be a trichromatic triangle of F and $\mathbf{p}$ be the point in Δ with color 0, then the construction of F guarantees that $\mathcal{F}^{-1}(\mathbf{p}) \in V_{2^k}$ is a directed leaf of G, which is different from $(0, 0)$.

Firstly, the mapping $\mathcal{F}$ from V_{2^k} to $T_{2^{2k+5}}$ is defined as $\mathcal{F}(\mathbf{u}) = \mathbf{p}$ where $p_1 = 3u_1 + 3$ and $p_2 = 3u_2 + 3$. For each $\mathbf{uv} \in E_{2^k}$, we use $I(\mathbf{uv})$ to denote the set of four vertices in $T_{2^{2k+5}}$ which lie on the segment between $\mathcal{F}(\mathbf{u})$ and $\mathcal{F}(\mathbf{v})$. Let $P = \mathbf{u}^1 \ldots \mathbf{u}^t$ be a simple path or cycle in G_{2^k} where $t > 1$ (if P is a cycle, then $\mathbf{u}^1 = \mathbf{u}^t$), then we define $I(P) = \cup_{i=1}^{t-1} I(\mathbf{u}^i\mathbf{u}^{i+1})$ and $O(P) \subset T_{2^{2k+5}}$ as

$$O(P) = \left\{\mathbf{p} \in T_{2^{2k+5}} \text{ and } \mathbf{p} \notin I(P) \mid \exists\, \mathbf{p}' \in I(P),\ ||\mathbf{p} - \mathbf{p}'||_\infty = 1\right\}.$$

If P is a simple path, then we can further decompose $O(P)$ into $\{\mathbf{s}_P, \mathbf{e}_P\} \cup L(P) \cup R(P)$. Here $\mathbf{s}_P = \mathcal{F}(\mathbf{u}^1) + (\mathbf{u}^1 - \mathbf{u}^2)$ and $\mathbf{e}_P = \mathcal{F}(\mathbf{u}^t) + (\mathbf{u}^t - \mathbf{u}^{t-1})$. Starting from $\mathbf{s}_P$, we enumerate vertices in $O(P)$ clockwise as $\mathbf{s}_P$, $\mathbf{q}^1 \ldots \mathbf{q}^{n_1}$, $\mathbf{e}_P$, $\mathbf{r}^1 \ldots \mathbf{r}^{n_2}$, then

$$L(P) = \left\{\mathbf{q}^1, \mathbf{q}^2 \ldots \mathbf{q}^{n_1}\right\} \quad \text{and} \quad R(P) = \left\{\mathbf{r}^1, \mathbf{r}^2 \ldots \mathbf{r}^{n_2}\right\}.$$

Turing Machine F with input $\mathbf{p} = (p_1, p_2) \in T_{2^{2k+5}}$

1: **if** $p_1 = 0$ **then**
2: case $p_2 \leq 3$, output 0; case $p_2 > 3$, output 1
3: **else if** $p_1 = 1$ **then**
4: case $p_2 = 3$, output 0; case $p_2 = 4$, output 1; otherwise, output 2
5: **else if** $p_1 = 2$ and $p_2 = 3$ **then**
6: output 0
7: let $t = M_K(\mathbf{p})$. case $t = 1$, output 0; case $t = 2$, output 1; otherwise, output 2

Fig. 5. Behavior of turing machine F.

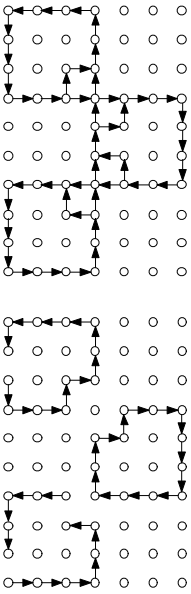

Fig. 6. G_2 and $G \in C_2$.

If P is a simple cycle, then we decompose $O(P)$ into $L(P) \cup R(P)$ where $L(P)$ contains all the vertices on the left side of the cycle and $R(P)$ contains all the vertices on the right side of the cycle.

As the graph G specified by $(K, 0^k)$ belongs to C_{2^k}, we can uniquely decompose its edge set into $P_1, \ldots P_m$. For every $i : 1 \leq i \leq m$, P_i is either a maximal path (that is, no path in G contains P_i), or a cycle in graph G. For both cases, the length of P_i is at least 1. One can prove the following two lemmas.

Lemma 4. *For all $i, j : 1 \leq i \neq j \leq m$, $\big(I(P_i) \cup O(P_i)\big) \cap \big(I(P_j) \cup O(P_j)\big) = \emptyset$.*

Lemma 5. *Let $(K, 0^k)$ be an input instance of problem* **RLEAFD** *and $G \in C_{2^k}$ be the directed graph specified, we can construct a polynomial-time TM M_K in polynomial time with the following properties. Given any vertex $\mathbf{p} \in T_{2^{2k+5}}$, it outputs an integer t: $0 \leq t \leq 5$. Let the unique decomposition of graph G be $P_1, P_2 \ldots P_m$, then: if $\exists\, i$, $\mathbf{p} \in I(P_i)$, then $t = 1$; if $\exists\, i$, $\mathbf{p} \in L(P_i)$, then $t = 2$; if $\exists\, i$, $\mathbf{p} \in R(P_i)$, then $t = 3$; if $\exists\, i$, $\mathbf{p} = \mathbf{s}_{P_i}$, then $t = 4$; if $\exists\, i$, $\mathbf{p} = \mathbf{e}_{P_i}$, then $t = 5$; otherwise, $t = 0$.*

Finally, Turing machine F is described by the algorithm in Fig. 5. For example, let $G \in C_2$ be the graph generated by $(K, 0^1)$, which is illustrated in Fig. 6, then Fig. 7 shows the 3-coloring F on T_{128}. As T_{128} contains so many vertices, not all of them are drawn in Fig. 7. For every omitted vertex $\mathbf{p} \in T_{128}$, if $p_1 = 0$, then $F(\mathbf{p}) = 1$, otherwise, $F(\mathbf{p}) = 2$.

One can prove the following two properties of TM F: **(1)** the 3-coloring f specified by F is admissible; **(2)** let $\Delta \in S$ be a trichromatic triangle and $\mathbf{p}$ be the vertex in Δ with color 0, then $\mathbf{u} = \mathcal{F}^{-1}(\mathbf{p})$ is a directed leaf of G, which is different from $(0, 0)$. By these two properties, we get the following theorem.

Theorem 1. *Search problem* **2D-SPERNER** *is* **PPAD**-*complete.*

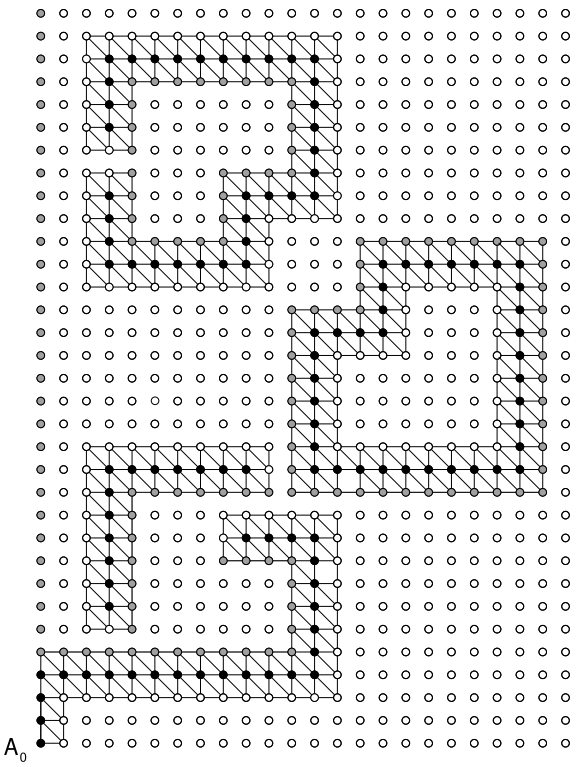

Fig. 7. F: black − 0, gray − 1, white − 2.

6. 2D-BROUWER is PPAD-complete

Recently, Daskalakis, Goldberg and Papadimitriou [3] proved that the problem of computing Nash equilibria in games with four players is **PPAD**-complete. In the proof, they define a 3D Brouwer fixed point problem and proved it is **PPAD**-complete. By reducing it to 4-NASH, they show that the latter is also complete in **PPAD**.

In this section, we first define a new problem **2D-BROUWER** which is a 2D analogue of the 3D problem [3]. By reducing **2D-SPERNER** to **2D-BROUWER**, we prove the latter is also **PPAD**-complete.

For every $n > 1$, we let

$$B_n = \left\{\mathbf{p} = (p_1, p_2) \in \mathbb{Z}^2 \mid 0 \le p_1 < n-1 \text{ and } 0 \le p_2 < n-1\right\}.$$

The boundary of B_n is the set of points $\mathbf{p} \in B_n$ with $p_i \in \{0, n-1\}$ for some $i \in \{1, 2\}$. For every $\mathbf{p} \in \mathbb{Z}^2$, we let $K_{\mathbf{p}} = \{\mathbf{q} \in \mathbb{Z}^2 \mid q_i = p_i \text{ or } p_i + 1,\ \forall\, i \in \{1, 2\}\}$. A 3-coloring of B_n is a function g from B_n to $\{0, 1, 2\}$. It is said to be valid if for every $\mathbf{p}$ on the boundary of B_n: if $p_2 = 0$, then $g(\mathbf{p}) = 2$; if $p_2 \neq 0$ and $p_1 = 0$, then $g(\mathbf{p}) = 0$; otherwise, $g(\mathbf{p}) = 1$.

Definition 7 (*2D-BROUWER*). The input instance of **2D-BROUWER** is a pair $(F, 0^k)$ where F is a polynomial-time TM which produces a valid 3-coloring g on B_{2^k}. Here $g(\mathbf{p}) = F(\mathbf{p}) \in \{0, 1, 2\}$ for every $\mathbf{p} \in B_{2^k}$. The output is a point $\mathbf{p} \in B_{2^k}$ such that $K_{\mathbf{p}}$ is trichromatic, that is, $K_{\mathbf{p}}$ has all the three colors.

The reason we relate this discrete problem to Brouwer's fixed point theorem is as follows. Let $\mathcal{G}$ be a continuous map from $[0, n-1] \times [0, n-1]$ to itself. If $\mathcal{G}$ satisfies a Lipschitz condition with a large enough constant, then we can construct a valid 3-coloring g on B_n such that:

1. For every point $\mathbf{p} \in B_n$, $g(\mathbf{p})$ only depends on $\mathcal{G}(\mathbf{p})$;
2. Once getting a point $\mathbf{p} \in B_n$ such that $K_{\mathbf{p}}$ is trichromatic, one can immediately locate an approximate fixed point of map $\mathcal{G}$.

Details of the construction can be found in [1].

Notice that the output of **2D-BROUWER** is a set $K_{\mathbf{p}}$ of four points which have all the three colors. Of course, one can pick three vertices in $K_{\mathbf{p}}$ to form a trichromatic triangle Δ, but it's possible that $\Delta \notin S$. Recall that every triangle in S has a northwest oriented hypotenuse. In other words, the hypotenuse of the trichromatic triangle in $K_{\mathbf{p}}$ might be northeast oriented. As a result, **2D-BROUWER** could be easier than **2D-SPERNER**.

Motivated by the discussion above, we define a problem **2D-BROUWER*** whose output is similar to **2D-SPERNER**. One can reduce **2D-SPERNER** to **2D-BROUWER*** easily and prove the latter is complete in **PPAD**.

Turing Machine F' with input $\mathbf{p} = (p_1, p_2) \in B_{3n}$

1: let $p_1 = 3l + i$ and $p_2 = 3k + j$, where $0 \le i, j \le 2$
2: **if** $(i, j) = (0, 0), (1, 0)$ or $(0, 1)$ **then**
3: $\quad F'(\mathbf{p}) = F(\mathbf{q})$ where $q_1 = l$ and $q_2 = k$
4: **else if** $(i, j) = (1, 1), (2, 0)$ or $(2, 1)$ **then**
5: $\quad F'(\mathbf{p}) = F(\mathbf{q})$ where $q_1 = l + 1$ and $q_2 = k$
6: **else** [when $j = 2$]
7: $\quad F'(\mathbf{p}) = F(\mathbf{q})$ where $q_1 = l$ and $q_2 = k + 1$

Fig. 8. The construction of Turing machine F'.

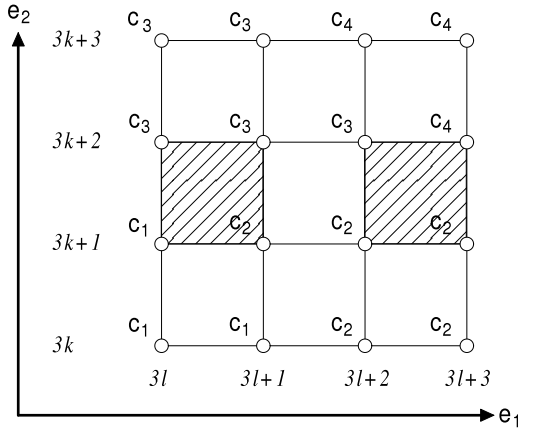

Fig. 9. F': $c_1 = F(l, k)$, $c_2 = F(l + 1, k)$, $c_3 = F(l, k + 1)$ and $c_4 = F(l + 1, k + 1)$.

Definition 8 (*2D-BROUWER**). The input instance is a pair $(F, 0^k)$ where F is a polynomial-time Turing machine which generates a valid 3-coloring g on B_{2^k}. Here $g(\mathbf{p}) = F(\mathbf{p}) \in \{0, 1, 2\}$ for every $\mathbf{p} \in B_{2^k}$.

The output is a trichromatic triangle $\Delta \in S$ which has all the three colors.

We now give a reduction from **2D-BROUWER*** to **2D-BROUWER**.

Let $(F, 0^k)$ be an input pair of **2D-BROUWER***, and $n = 2^k$. In Fig. 8, we describe a new Turing machine F' which generates a new 3-coloring on B_{3n}. For $0 \le l, k < n$, Fig. 9 shows the 3-coloring produced by F' on $\{3l, 3l+1, 3l+2, 3l+3\} \times \{3k, 3k+1, 3k+2, 3k+3\} \subset B_{3n}$. Clearly, F' is also a polynomial-time Turing machine, which can be constructed from F in polynomial time. Besides, F' generates a valid 3-coloring on B_{3n}. We now prove that, for every $\mathbf{p} \in B_{3n}$ such that set $K_{\mathbf{p}}$ is trichromatic in F', one can recover a trichromatic triangle $\Delta \in S$ in F easily.

Let $p_1 = 3l + i$ and $p_2 = 3k + j$, where $0 \le i, j \le 2$. By examining Fig. 9, we know that either $(i, j) = (0, 1)$ or $(i, j) = (2, 1)$. Furthermore,

1. if $(i, j) = (0, 1)$, then $\Delta = (\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^2) \in S$ is a trichromatic triangle in F, where $\mathbf{p}^0 = (k, l)$, $\mathbf{p}^1 = \mathbf{p}^0 + \mathbf{e}_1$ and $\mathbf{p}^2 = \mathbf{p}^0 + \mathbf{e}_2$;
2. if $(i, j) = (2, 1)$, then $\Delta = (\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^2) \in S$ is a trichromatic triangle in F, where $\mathbf{p}^0 = (k + 1, l + 1)$, $\mathbf{p}^1 = \mathbf{p}^0 - \mathbf{e}_1$ and $\mathbf{p}^2 = \mathbf{p}^0 - \mathbf{e}_2$.

Finally, we get an important corollary of Theorem 1.

Theorem 2. *Search problem* **2D-BROUWER** *is* **PPAD***-complete.*

7. Concluding remarks

All the **PPAD**-completeness proofs of Sperner's problems before rely heavily on embeddings of complete graphs in the standard subdivisions. That is, edges in the complete graph correspond to independent paths which are composed of neighboring triangles or tetrahedrons in the standard subdivision. Such an embedding is obviously impossible in the plane, as complete graphs with order no less than 5 are not planar. We overcome this difficulty by placing a carefully designed

gadget (which looks like a switch with two states) at each intersection of two paths. While the structure of the graph is mutated dramatically (e.g. Fig. 4), the property of a vertex being a leaf is well maintained.

An important corollary of the **PPAD**-completeness of **2D-SPERNER** is that, the computation of discrete Brouwer fixed points in 2D spaces (**2D-BROUWER**) is also **PPAD**-complete. Our new proof techniques may provide helpful insight into the study of other related problems: Can we show more problems complete for **PPA** and **PPAD**? For example, is **2D-TUCKER** [9] **PPAD**-complete? Can we find a natural complete problem for either **PPA** or **PPAD** that does not have an explicit Turing machine in the input? For example, is **SMITH** [9] **PPA**-complete? Finally and most importantly, what is the relationship between complexity classes **PPA**, **PPAD** and **PPADS**?

Acknowledgements

The first author's work is supported by the National Natural Science Foundation of China (Grant 60553001), and the National Basic Research Program of China (Grant 2007CB807900, 2007CB807901). The second author's work is supported by an SRG grant (No. 7001838) of City University of Hong Kong.

References

[1] X. Chen, X. Deng, On algorithms for discrete and approximate Brouwer fixed points, in: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, STOC, 2005, pp. 323–330.
[2] X. Chen, X. Deng, S.-H. Teng, Computing Nash equilibria: Approximation and smoothed complexity, in: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, FOCS, 2006.
[3] C. Daskalakis, P.W. Goldberg, C.H. Papadimitriou, The complexity of computing a Nash equilibrium, in: Proceedings of the 38th Annual ACM Symposium on Theory of Computing, STOC, 2006.
[4] K. Friedl, G. Ivanyos, M. Santha, F. Verhoeven, On the black-box complexity of Sperner's lemma, in: Proceedings of the 15th International Symposium on Fundamentals of Computation Theory, 2005, pp. 245–257.
[5] K. Friedl, G. Ivanyos, M. Santha, F. Verhoeven, Locally 2-dimensional Sperner problems complete for polynomial parity argument classes, in: Proceedings of the 6th Italian Conference on Algorithms and Complexity, 2006, pp. 380–391.
[6] M. Grigni, A Sperner lemma complete for PPA, Information Processing Letters 77 (5–6) (2001) 255–259.
[7] M.D. Hirsch, C.H. Papadimitriou, S. Vavasis, Exponential lower bounds for finding Brouwer fixed points, Journal of Complexity 5 (1989) 379–416.
[8] C.H. Papadimitriou, On graph-theoretic lemmata and complexity classes, in: Proceedings of the 31st Annual Symposium on Foundations of Computer Science, 1990, pp. 794–801.
[9] C.H. Papadimitriou, The complexity of the parity argument and other inefficient proofs of existence, Journal of Computer and System Sciences 48 (1994) 498–532.
[10] E. Sperner, Neuer beweis fur die invarianz der dimensionszahl und des gebietes, Abhandlungen aus dem Mathematischen Seminar Universitat Hamburg 6 (1928) 265–272.