Electronic Colloquium on Computational Complexity, Report No. 37 (2006)

On the Complexity of 2D Discrete Fixed Point Problem

Xi Chen[1] and Xiaotie Deng[2]

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

Abstract. While the 3-dimensional analogue of the Sperner problem in the plane was known to be **PPAD**-complete, the complexity of the **2D-SPERNER** itself is not known to be **PPAD**-complete or not. In this paper, we settle this open problem proposed by Papadimitriou [7] fifteen years ago. This also allows us to derive the computational complexity characterization of a discrete version of the 2-dimensional Brouwer fixed point problem, improving a recent result of Daskalakis, Goldberg and Papadimitriou [2]. Those results will be very useful tools to the study of other related problems.

1 Introduction

The classical Sperner's Lemma [9], which is the combinatorial characterization behind 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 a trichromatic triangle in any admissible 3-coloring, which is one of the most typical problems in **PPAD**, a complexity class introduced by Papadimitriou to characterize a class of mathematical structures with a particular type of proof techniques [8]. Many important problems, such as the Brouwer fixed point, the search versions of Smith's theorem, as well as Borsuk-Ulam theorem, are found in this class [8].

For the Sperner's problem, its 3-dimensional analogue **3D-SPERNER** is the first natural problem ever proved to be **PPAD**-complete [8]. In conclusion of the proof of **PPAD**-completeness of the **3D-SPERNER**, Papadimitriou conjectured that the 2-dimensional case may not also be **PPAD**-complete. Since then, much progress has been made recently toward the solution of this problem: In [5], Grigni described a non-oriented version of **3D-SPERNER** and proved that it is **PPA**-complete. Friedl et al proved in [3, 4] that locally 2-dimensional Sperner's problem is **PPAD**-complete. However, the original 2-dimensional Sperner's problem remains elusive.

In this paper, we settle this problem by proving that **2D-SPERNER** is **PPAD**-complete and close the open problem proposed by Papadimitriou [7] fifteen years ago. Moreover, this result allows us to derive the **PPAD**-complete proof of a discrete version of the 2D Brouwer fixed point problem.

ISSN 1433-8092

Our study is motivated by the lower bound results in [1] and [6] which discussed the algorithmic complexity of finding a discrete Brouwer fixed point in d-dimensional space. The combinatorial structure there is similar to the one here. It was proved that, for any dimension $d \geq 2$, the fixed point problem for the oracle model always requires exponential number of queries. Although the computational models are different, we expect that the complexity hierarchy in the Sperner's problem may have a similar structure with respect to the dimension.

In examination into the **PPAD**-complete proof of **3D-SPERNER**, and those of recent versions of locally 2-dimensional Sperner's problems, we found that the main idea is to embed complete graphs in the 3-dimensional space, or locally 2-dimensional manifolds. In both [8] and [3], the proof of completeness relies on such an embedding which is obviously impossible in the plane. Our approach is different, and the proof can be divided clearly into two steps. First, we define a new search problem called **RLEAFD** (restricted-**LEAFD** where **LEAFD** is the natural complete problem of **PPAD**). While the input graph has the same property as those in problem **LEAFD**, it is guaranteed to be a sub-graph of some predefined planar grid graph. The interesting result followed is that, even with such a strong restriction, **RLEAFD** is still **PPAD**-complete. In the second step, we reduce **RLEAFD** to **2D-SPERNER** and prove the latter is also complete. The main idea represents a better understanding of **PPAD** reductions and can be of general applicability in related problems.

We should in the next section introduce the necessary definitions. We then define a new search problem called **RLEAFD** in section 3 and prove that it is **PPAD**-complete in section 4. In section 5, we show that **RLEAFD** is reducible to **2D-SPERNER** and complete the proof that the latter is also complete in **PPAD**. In addition, we prove that the 2-dimensional case of a discrete version of the Brouwer fixed point problem [2], **2D-BROUWER**, as a corollary, is also **PPAD**-complete. Finally, we conclude with discussion on the significance of our results and potential applications, in particular, to important problems in algorithmic game theory such as the bimatrix game Nash Equilibrium.

2 Preliminaries

2.1 TFNP and PPAD

First, we review the complexity classes **FNP** and **TFNP** of search problems. We also formalize the notion of polynomial-time reductions between problems in **TFNP** [7].

Definition 1 (FNP and TFNP) *Let $R \subset \Sigma^* \times \Sigma^*$ be a polynomial-time computable, polynomially balanced relation (that is, there exists a polynomial $p(n)$ such that for every pair $(x, y) \in R$, we have $|y| \leq p(|x|)$). The* **NP** *search problem Q_R specified by R is this: given input $x \in \Sigma^*$, return a $y \in \Sigma^*$ such that $(x, y) \in R$, if such a y exists, and return the string "no" otherwise. We use* **FNP** *to denote the class of* **NP** *search problems. An* **NP** *search problem Q_R*

is said to be total if for every x, there exists a y such that $(x, y) \in R$. We use **TFNP** *to denote the class of total* **NP** *search problems.*

Definition 2 (Polynomial Reduction) *A search problem $Q_{R_1} \in$* **TFNP** *is* polynomial-time reducible *to another $Q_{R_2} \in$* **TFNP** *if there exists a pair of polynomial-time computable functions (f, g) such that, for every input x of R_1, if y satisfies $(f(x), y) \in R_2$, then $(x, g(y)) \in R_1$.*

We then define a total **NP** search problem called **LEAFD** [8].

Definition 3 (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 every $v \in \{0,1\}^k$, $M(v)$ is an ordered pair (u_1, u_2) where*

$$u_1, u_2 \in \{0,1\}^k \cup \{no\};$$

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 $V = \{0,1\}^k$ in the following way. 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) of graph G which is different from 0^k.

PPAD [7] is the set of total **NP** search problems that are polynomial-time reducible to **LEAFD**. From its definition, **LEAFD** is complete for **PPAD**.

2.2 2D-SPERNER

One of the most interesting problems in **PPAD** is **2D-SPERNER** whose totality is based on the Sperner's Lemma [9].

Lemma 1 (Sperner's Lemma) *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 figure 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 three vertices of the original triangle. The vertex set T_n of the standard $n \times n$ triangulation is defined as

$$T_n = \left\{ \mathbf{p} = (p_1, p_2) \in \mathbb{Z}^2 \;\middle|\; p_1 \geq 0,\ p_2 \geq 0 \text{ and } p_1 + p_2 \leq n \right\}.$$

A 3-coloring of the $n \times n$ triangulation is a function f from vertex set T_n to $\{0, 1, 2\}$. It is said to be admissible if

1. $f(A_i) = i$ for every $i : 0 \leq i \leq 2$;

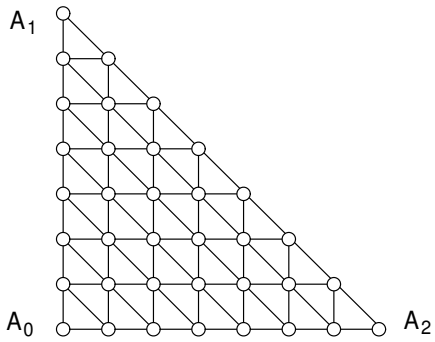

Fig. 1. The standard 7×7 triangulation of a triangle

2. for every $\mathbf{p} \in T_n$ 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 $\mathbf{p}^i \in \mathbb{Z}^d$ for every $0 \leq i \leq 2$. It satisfies either $\mathbf{p}^1 = \mathbf{p}^0 + \mathbf{e}_1$, $\mathbf{p}^2 = \mathbf{p}^0 + \mathbf{e}_2$ or $\mathbf{p}^1 = \mathbf{p}^0 - \mathbf{e}_1$, $\mathbf{p}^1 = \mathbf{p}^0 - \mathbf{e}_2$. In other words, the triangle has a northwest oriented hypotenuse. We use S to denote the set of all such triangles.

Sperner's Lemma guarantees that every admissible 3-coloring f has a trichromatic triangle $\Delta = (\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^2) \in S$, that is, it has all the three colors.

The search problem **2D-SPERNER** is defined as follow.

Definition 4 (2D-SPERNER [7]) *The input instance is a pair $(F, 0^k)$ where F is the description of a polynomial-time Turing machine which generates an admissible 3-coloring f on T_{2^k}. Here $f(\mathbf{p}) = F(\mathbf{p}) \in \{0, 1, 2\}$, for every $\mathbf{p} \in T_{2^k}$.*

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

[7] showed that **2D-SPERNER** belongs to **PPAD**. They also defined a 3-dimensional analogue **3D-SPERNER** of **2D-SPERNER** and proved that it is **PPAD**-complete. The two dimensional case was left as an open problem.

3 Definition of Search Problem RLEAFD

Before the definition of problem **RLEAFD**, we describe a class of planar grid graphs G_1, G_2,... where $G_i = (V_i, E_i)$ is derived from the complete graph K_i of order i.

For every integer $n \geq 1$, we let

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

Informally speaking, G_n can be viewed as a planar embedding of the complete graph K_n with vertex set $\{0, 1, \ldots n-1\}$. Every vertex i of K_n corresponds to the vertex $(0, 6i)$ of G_n. To describe the edge set E_n, we start with $E_n = \emptyset$. For every edge $ij \in K_n$, we define a path E_{ij} in G_n from vertex $(0, 6i)$ to $(0, 6j)$, and

add all the edges in E_{ij} into E_n. As K_n is not a planar graph when $n \geq 5$, there are many vertices of G_n which are at the intersection of two paths $E_{ij}, E_{i'j'}$ added previously. For each of them, we add 4 more edges around it 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 $E_n^1 = \cup_{ij \in K_n} E_{ij}$ where path E_{ij} is defined as follows.

Definition 5 *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 $\mathbf{u}^1, \mathbf{u}^2 ... \ \mathbf{u}^m \in \mathbb{Z}^2$ be all the integral points on segment $\mathbf{p}^1\mathbf{p}^2$ which are labelled 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: $\mathbf{u}^1\mathbf{u}^2$, $\mathbf{u}^2\mathbf{u}^3$, ... $\mathbf{u}^{m-1}\mathbf{u}^m$.*

Definition 6 *For every edge ij in the complete graph K_n where $0 \leq i \neq j < n$, we define a path E_{ij} in G_n as*

$$E_{ij} = E(\mathbf{p}^1\mathbf{p}^2) \cup E(\mathbf{p}^2\mathbf{p}^3) \cup E(\mathbf{p}^3\mathbf{p}^4) \cup E(\mathbf{p}^4\mathbf{p}^5),$$

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 G_n has at most 4 edges (including both incoming and outgoing edges) in E_n^1. Moreover, if $\mathbf{u}$ has 4 edges, then $3 \mid u_1$ and $3 \mid u_2$. We now use $\{\mathbf{u}^i\}_{1 \leq i \leq 8}$ to denote the eight vertices around $\mathbf{u}$. For every $i : 1 \leq i \leq 8$, $\mathbf{u}^i = \mathbf{u} + \mathbf{x}^i$ where

$$\mathbf{x}^1 = (-1, 1), \quad \mathbf{x}^2 = (0, 1), \quad \mathbf{x}^3 = (1, 1), \quad \mathbf{x}^4 = (1, 0),$$
$$\mathbf{x}^5 = (1, -1), \quad \mathbf{x}^6 = (0, -1), \quad \mathbf{x}^7 = (-1, -1) \text{ and } \mathbf{x}^8 = (-1, 0).$$

Every $\mathbf{u} \in V_n$ which has 4 edges in E_n^1 must have 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 $\mathbf{u}^6\mathbf{u}, \mathbf{u}\mathbf{u}^2 \in E_n^1$.

For every vertex $\mathbf{u} \in V_n$ which has 4 edges in E_n^1, there are 4 more edges in E_n^2 around it. 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 figure 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 clearly in figure 2.

Now we have finished the construction of directed graph G_n. An example (graph G_3) is showed in figure 6 in appendix. We can draw it in two steps. In the first step, for every edge $ij \in K_n$, we insert the path E_{ij} into the empty graph. In the second step, we search for vertices of degree 4. For each of them, 4 edges are added according to figure 2. The following lemma concerning the computability of graph G_n is easy to prove.

Lemma 2 *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 in V_n, it outputs all the predecessors and successors of $\mathbf{u}$ in graph G_n.*

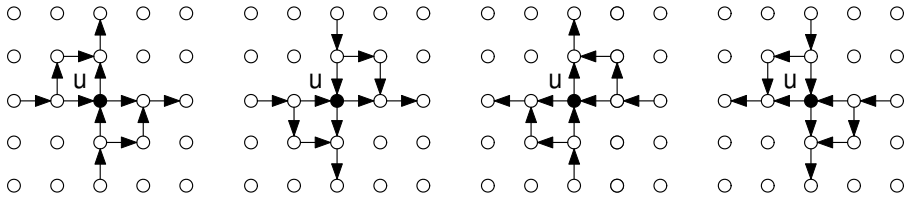

Fig. 2. Summary of cases in the construction of E_n^2

Definition 7 *We use C_n to denote the set of directed graphs $G = (V_n, E)$ such that $E \subset E_n$ and for every vertex $\mathbf{u} \in V_n$, both of its in-degree and out-degree are no more than one.*

The new problem **RLEAFD** is similar to **LEAFD**. The input is a also pair $(K, 0^k)$ where K is the description of a polynomial-time Turing machine that generates a directed graph G of exponential size. The key difference is that, the graph G generated by $(K, 0^k)$ in **RLEAFD** always belongs to C_{2^k}.

Definition 8 (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 $\mathbf{u} \in V_{2^k}$, $K(\mathbf{u})$ is an ordered pair $(\mathbf{u}^1, \mathbf{u}^2)$ where*

$$\mathbf{u}^1, \mathbf{u}^2 \in V_{2^k} \cup \{no\};$$

2. *$K((0,0)) = (no, (1,0))$ and the first component of $K((1,0))$ is $(0,0)$.*

K generates a directed graph $G = (V_{2^k}, E) \in C_{2^k}$ in the following way. 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 is a directed leaf (with in-degree + out-degree = 1) of graph G which is different from the origin $(0,0)$.

By Lemma 2, one can show that **RLEAFD** $\in$ **PPAD**.

4 RLEAFD is PPAD-Complete

In this section, we will describe a polynomial-time reduction from **LEAFD** to **RLEAFD** and prove that **RLEAFD** is complete in **PPAD**.

Let G be a directed graph with vertex set $\{0, 1...n-1\}$ which satisfies that the in-degree and out-degree of every vertex are at most one. We now construct 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. This property plays a vital role in the reduction from **LEAFD** to **RLEAFD**.

Step 1: Starting with an empty graph $(V_n, \emptyset)$, for every edge $ij \in G$, we add all the edges in path E_{ij} into it.

Step 2: For every $\mathbf{u} \in V_n$ which has 4 edges at this moment, we remove all the 4 edges which have $\mathbf{u}$ as an endpoint and add 4 edges around $\mathbf{u}$ using figure 2.

One can check that the in-degree and out-degree of every vertex in $C(G)$ are no more than 1 and thus, $C(G) \in C_n$. For example, figure 7 in appendix shows graph $C(G)$ where $G = (\{0, 1, 2\}, \{02, 21\})$. The following lemma is easy to check.

Lemma 3 *For every vertex $0 \leq k \leq n-1$ of G, it is a directed leaf of G iff $\mathbf{u} = (0, 6k) \in V_n$ is a directed leaf of $C(G)$.*

Finally, we prove that **RLEAFD** is **PPAD**-complete.

Lemma 4 *Search problem* **RLEAFD** *is* **PPAD***-complete.*

Proof. Let $(M, 0^k)$ be an input instance of problem **LEAFD** and G be the directed graph generated. A Turing machine K is described by the algorithm in figure 10, which satisfies the two conditions in Definition 8. It's tedious, but not hard to check that, pair $(K, 0^k)$, as an input of **RLEAFD**, generates graph $C(G) \in C_{2^k}$. Furthermore, as M is polynomial-time, K is also a polynomial-time Turing machine whose representation can be constructed from $(M, 0^k)$ in polynomial time. On the other hand, Lemma 3 shows that, given a directed leaf of $C(G)$, we can locate a directed leaf of G efficiently.

In conclusion, we reduced **LEAFD** to **RLEAFD** and thus, the latter is also **PPAD**-complete.

5 2D-SPERNER is PPAD-Complete

In this section, we reduce **RLEAFD** to **2D-SPERNER** and finish the **PPAD**-complete 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 generates an admissible 3-coloring on $T_{2^{2k+5}}$. Given a trichromatic triangle $\Delta \in S$, a directed leaf of G can be computed efficiently. To clarify the presentation here, we will only 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 will 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 $P_1, P_2, ... P_m$. By the mapping $\mathcal{F}$, every P_i corresponds to a vertex 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 are colored carefully in either 1 or 2. Let $\Delta = (\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^2) \in S$ be a trichromatic triangle and $F(\mathbf{p}^i) = 0$ where $0 \leq i \leq 2$,

then we will prove that $\mathcal{F}^{-1}(\mathbf{p}^i)$ must be 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 every $\mathbf{uv} \in E_{2^k}$, 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 $P = \mathbf{u}^1 ... \mathbf{u}^t$ 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 $I(P) \subset T_{2^{2k+5}}$ as

$$I(P) = \cup_{i=1}^{t-1} I(\mathbf{u}^i \mathbf{u}^{i+1})$$

and vertex set $O(P) \subset T_{2^{2k+5}}$ as

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

If P is a simple path, then we decompose $O(P)$ into $\{\mathbf{s}_P, \mathbf{e}_P\} \cup L(P) \cup R(P)$.

$$\mathbf{s}_P = \mathcal{F}(\mathbf{u}^1) + (\mathbf{u}^1 - \mathbf{u}^2) \quad \text{and} \quad \mathbf{e}_P = \mathcal{F}(\mathbf{u}^t) + (\mathbf{u}^t - \mathbf{u}^{t-1}).$$

Starting at $\mathbf{s}_P$, we enumerate all the vertices in $O(P)$ clockwise as $\mathbf{s}_P, \mathbf{q}^1 ... \mathbf{q}^{n_1}$, $\mathbf{e}_P, \mathbf{r}^1 ... \mathbf{r}^{n_2}$ and

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

If P is a simple cycle, then we decompose $O(P)$ into $L(P) \cup R(P)$ where

$$L(P) = \left\{ \mathbf{p} \in O(P) \;\middle|\; \mathbf{p} \text{ lies on the left side of cycle } P \right\} \quad \text{and}$$
$$R(P) = \left\{ \mathbf{p} \in O(P) \;\middle|\; \mathbf{p} \text{ lies on the right side of cycle } P \right\}.$$

As the graph G which is specified by $(K, 0^k)$ belongs to C_{2^k}, we can uniquely decompose its edge set into $P_1, P_2, ... P_m$. For every $1 \le i \le m$, P_i is either a maximal path, that is, no path in G contains P_i, which contains at least two vertices, or a cycle in graph G. One can prove the following two lemmas.

Lemma 5 *For every $1 \le i \ne j \le m$, we have*

$$\big(I(P_i) \cup O(P_i)\big) \bigcap \big(I(P_j) \cup O(P_j)\big) = \emptyset.$$

Lemma 6 *Let $(K, 0^k)$ be an input instance of* **RLEAFD** *and $G \in C_{2^k}$ be the directed graph specified, we can compute the presentation of a polynomial-time Turing machine M_K in polynomial time. Given an vertex $\mathbf{p} \in T_{2^{2k+5}}$, it outputs an integer $0 \le t \le 4$ which has the following property: Let the decomposition of G be $P_1, P_2, ... P_m$, then*

$$\exists\, i,\ \mathbf{p} \in I(P_i) \ \Rightarrow\ t = 1; \qquad \exists\, i,\ \mathbf{p} \in L(P_i) \ \Rightarrow\ t = 2;$$
$$\exists\, i,\ \mathbf{p} \in R(P_i) \ \Rightarrow\ t = 3; \qquad \exists\, i,\ \mathbf{p} = \mathbf{s}_{P_i} \ \Rightarrow\ t = 4;$$
$$\exists\, i,\ \mathbf{p} = \mathbf{e}_{P_i} \ \Rightarrow\ t = 5; \qquad \textit{otherwise},\ t = 0.$$

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

Fig. 3. Behavior of Turing Machine F

Turing machine F is described by the algorithm in figure 3 below. As M_K is polynomial-time, F is also a polynomial-time Turing machine whose representation can be computed from pair $(K, 0^k)$ in polynomial time.

For example, let $G \in C_2$ be the directed graph generated by pair $(K, 0^1)$, which is illustrated in figure 8. Figure 9 in appendix shows the 3-coloring F on T_{128}, which is constructed from $(K, 0^1)$. As T_{128} contains so many vertices, not all of them are drawn in figure 9. 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 check the following two properties:

1. the 3-coloring specified by F is admissible;
2. let $\Delta = (\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^2) \in S$ be a trichromatic triangle of F and $F(\mathbf{p}^i) = 0$, where $0 \leq i \leq 2$, then vertex $\mathbf{u} = \mathcal{F}^{-1}(\mathbf{p}^i)$ must be 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.*

6 2D-BROUWER is PPAD-Complete

Recently, Daskalakis, Goldberg and Papadimitriou [2] proved that the problem of computing Nash equilibria in games with four players is **PPAD**-complete. In the proof, they define a 3-dimensional Brouwer fixed point problem and proved that it is **PPAD**-complete. By reducing it to 4-NASH, they show that the latter is also complete in **PPAD**.

We now define a new problem **2D-BROUWER** which is the 2-dimensional analogue of the 3-dimensional Brouwer fixed point problem in [2].

For every $n > 1$, we let

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

The boundary of B_n is then the set of points $\mathbf{p} \in B_n$ with $p_i \in \{0, r_i - 1\}$ for some $i = 1$ or 2. For every $\mathbf{p} \in \mathbb{Z}^2$, we let

$$K_{\mathbf{p}} = \left\{ \mathbf{q} = (q_1, q_2) \in \mathbb{Z}^2 \;\middle|\; q_1 = p_1 \text{ or } p_1 + 1 \text{ and } q_2 = p_2 \text{ or } p_2 + 1 \right\}.$$

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,

1. if $p_2 = 0$, then $g(\mathbf{p}) = 2$;
2. if $p_2 \neq 0$ and $p_1 = 0$, then $g(\mathbf{p}) = 0$;
3. otherwise, $g(\mathbf{p}) = 1$.

The search problem **2D-BROUWER** is then defined as follows.

Definition 9 (2D-BROUWER) *The input of the problem* **2D-BROUWER** *is a pair $(F, 0^k)$ where F is the description of 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 point $\mathbf{p} \in B_{2^k}$ such that $K_{\mathbf{p}}$ is trichromatic, that is, $K_{\mathbf{p}}$ has all the three colors.

Notice that the output of **2D-BROUWER** is a set $K_{\mathbf{p}}$ of 4 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 $\Delta \notin S$. In other words, the hypotenuse of Δ might be northeast oriented. So, **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**.

Definition 10 (2D-BROUWER*) *The input of the problem is a pair $(F, 0^k)$ where F is the description of 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 trichromatic triangle $\Delta \in S$ which has all the three colors.

We now construct a polynomial-time reduction from **2D-BROUWER*** to **2D-BROUWER**.

Let $(F, 0^k)$ be the input instance of **2D-BROUWER** and $n = 2^k$. In figure 4, we describe a new Turing machine F' which generates a 3-coloring on B_{3n}. For integers $0 \leq l, k < n$, figure 5 shows the 3-coloring F' on set

$$\Big\{ 3l, 3l+1, 3l+2, 3l+3 \Big\} \times \Big\{ 3k, 3k+1, 3k+2, 3k+3 \Big\} \subset B_{3n}.$$

Clearly, as F is polynomial-time, 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}.

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. 4. The construction of Turing machine F'

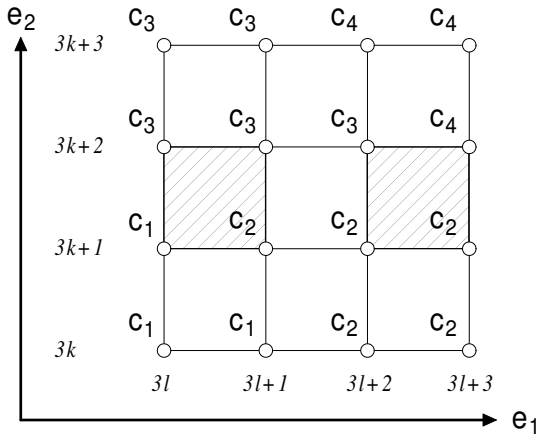

Fig. 5. The 3-coloring generated by Turing machine F'. $c_1 = F(l, k)$, $c_2 = F(l + 1, k)$, $c_3 = F(l, k + 1)$ and $c_4 = F(l + 1, k + 1)$

We now prove that, for every $\mathbf{p} \in B_{3n}$ such that $K_{\mathbf{p}}$ is trichromatic in F', one can find a trichromatic triangle $\Delta = (\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^2) \in S$ in F easily.

Let $p_1 = 3l + i$ and $p_2 = 3k + j$, where $0 \le i, j \le 2$. By examining figure 5, we know that either $(i, j) = (0, 1)$ or $(i, j) = (2, 1)$, since $K_{\mathbf{p}}$ has all the three colors. Figure 5 also shows that

1. if $(i, j) = (0, 1)$, then $(\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^2) \in S$ 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 $(\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^2) \in S$ 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**-complete 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. figure 7), the property of a vertex being a leaf is well maintained.

An important corollary follows Theorem 1 is that, the computation of Brouwer fixed point in 2-dimensional spaces (**2D-BROUWER**) is also complete in **PPAD**. Naturally, our results may serve as a basic tool to the same question on related problems: Can we show more problems complete for **PPA** and **PPAD**? For example, is **2D-TUCKER** [8] **PPAD**-complete? Can we find a natural complete problem for either **PPA** or **PPAD** that doesn't have an explicit Turing machine in the input? For example, is **SMITH** [8] (that is, given a Hamiltonian cycle in a graph with all vertices of odd degree, find another.) **PPA**-complete? Finally and most importantly, what is the relationship between complexity classes **PPA**, **PPAD** and **PPADS**? Indeed, our new fundamental results provide helpful insight into the study of the related problems of the two player Nash Equilibrium as well as its approximations.

References

1. Xi Chen and Xiaotie Deng. On Algorithms for Discrete and Approximate Brouwer Fixed Points. In *STOC 2005*, pages 323–330.
2. C. Daskalakis, P.W. Goldberg, and C.H. Papadimitriou. The Complexity of Computing a Nash Equilibrium. *STOC*, 2006.
3. K. Friedl, G. Ivanyos, M. Santha, and Y. Verhoeven. Locally 2-dimensional Sperner problems complete for the Polynomial Parity Argument classes. *submitted*.
4. K. Friedl, G. Ivanyos, M. Santha, and Y. Verhoeven. On the complexity of Sperner's Lemma. *Research report NI05002-QIS at the Isaac Newton Institute for Mathematical studies*.
5. M. Grigni. A Sperner lemma complete for PPA. *Inform. Process. Lett.*, 77(5-6):255–259, 2001.
6. M.D. Hirsch, C. Papadimitriou and S. Vavasis. Exponential lower bounds for finding Brouwer fixed points. *J.Complexity*, 5:379–416, 1989.
7. C.H. Papadimitriou. On graph-theoretic lemmata and complexity classes. In *In Proceedings 31st Annual Symposium on Foundations of Computer Science*, pages 794–801, 1990.
8. C.H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, pages 498–532, 1994.
9. E. Sperner. Neuer Beweis fur die Invarianz der Dimensionszahl und des Gebietes. *Abhandlungen aus dem Mathematischen Seminar Universitat Hamburg*, 6:265–272, 1928.

Appendix

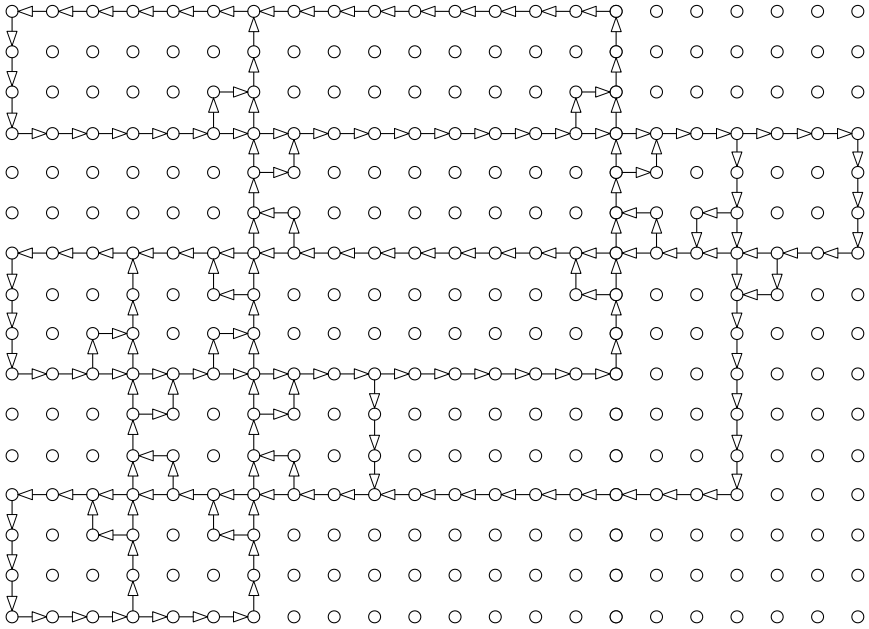

Fig. 6. The planar grid graph G_3

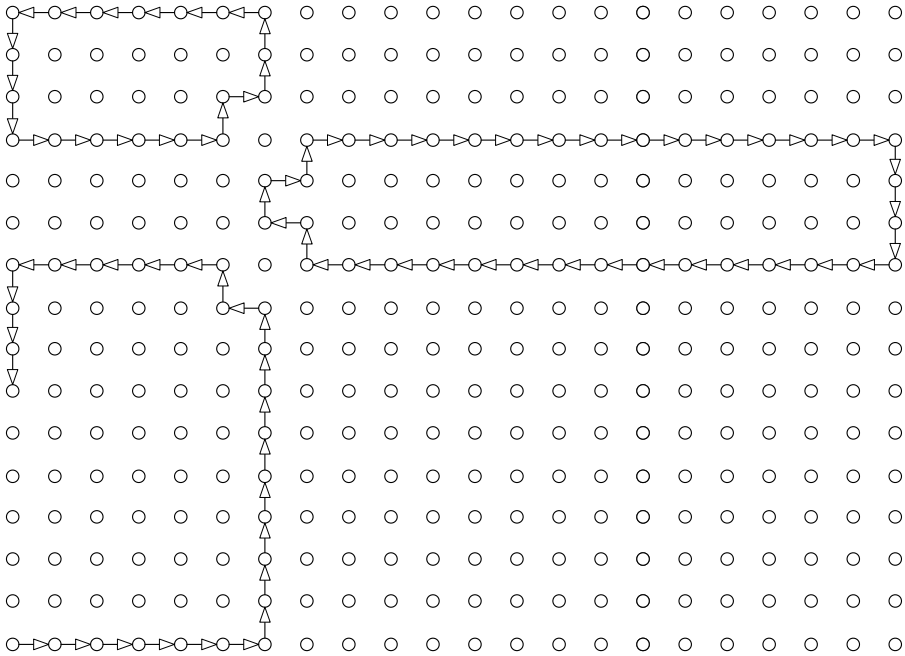

Fig. 7. Graph $C(G) \in C_3$ where $G = (\{0, 1, 2\}, \{02, 21\})$

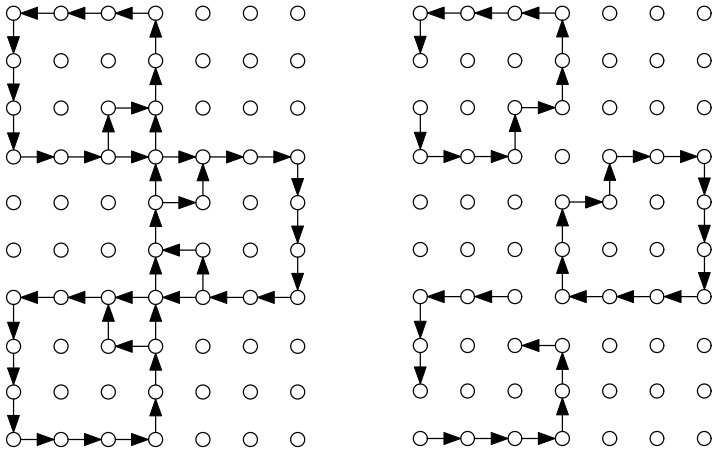

Fig. 8. Graph G_2 and one of its sub-graphs $G \in C_2$ specified by pair $(K, 0^1)$

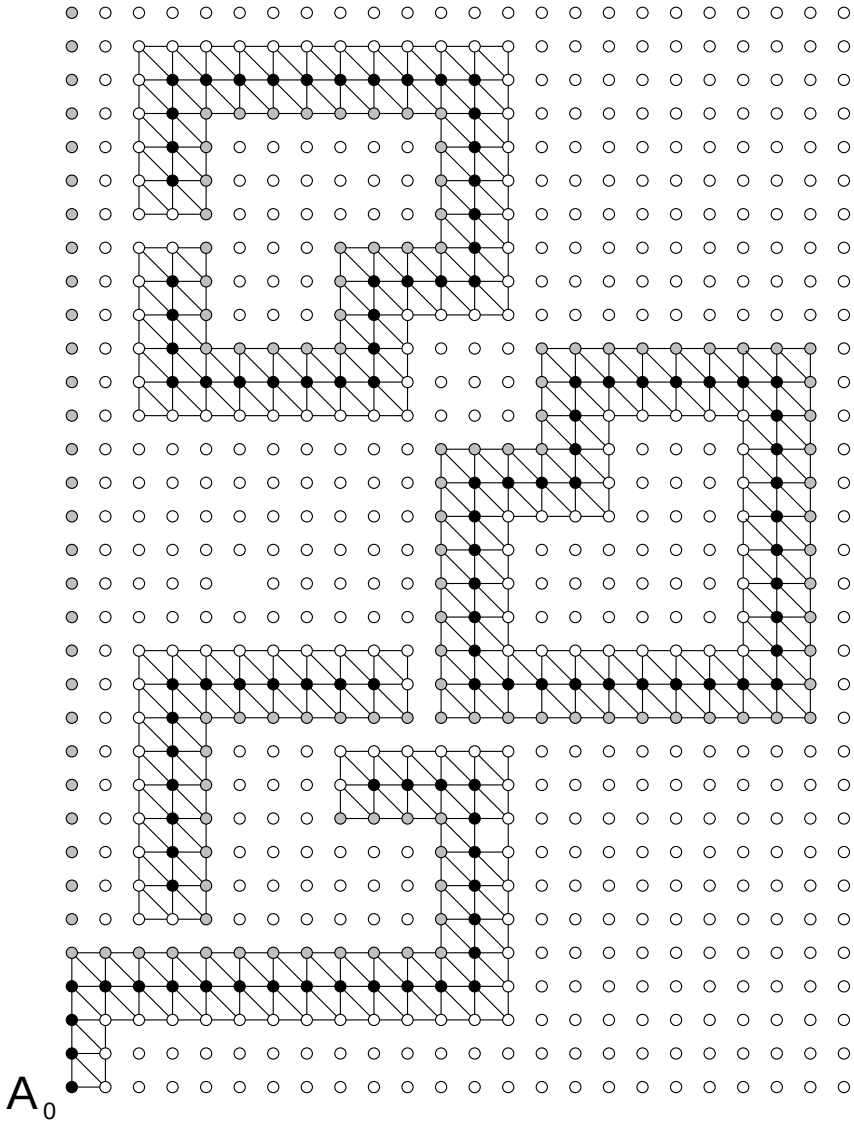

Fig. 9. The admissible 3-coloring F on T_{128} : **black** – **0**, **gray** – **1**, **white** – **2**

Turing Machine K with input $\mathbf{u} = (u_1, u_2) \in V_{2^k}$

let r^1 and r^2 be the nearest integer to $u_1/3$ and $u_2/3$ respectively
let $\mathbf{u}^* = (3r^1, 3r^2)$, $r^1 = i2^k + j$ where $0 \le i, j < 2^k$,
$r^2 = 2i^* + c$ where $c = 0$ or 1
if $r^1 = 0$ and $c = 0$ **then**
    use M to compute the predecessor i' and successor j' of vertex i^* in G
    case $i' \neq$ no, $j' \neq$ no, use figure **11.1**; case $i' \neq$ no, $j' =$ no, use figure **11.2**
    case $i' =$ no, $j' \neq$ no, use figure **11.3**; case $i' =$ no, $j' =$ no, use figure **11.4**
else if $r^1 = 0$ and $c = 1$ **then**
    use M to compute the predecessor i' of vertex i^* in G
    case $i' =$ no, use figure **11.4**; case $i' \neq$ no, use figure **11.5**
else if $r^1 > 0$ and $c = 0$ **then**
    use M to decide whether edge $ij \in G$ or not
    use M to compute the successor j' of vertex i^* in G
    if $j' =$ no or $i^*2^k + j' < r^1$ **then**
        case $ij \in G$ and $i < i^* \le j$, use figure **12.1**;
        case $ij \in G$ and $j < i^* < i$, use figure **12.2**;
        otherwise, use figure **12.3**
    else if $i^*2^k + j' = r^1$ **then**
        case $j' < i^*$, use figure **12.4**; otherwise, use figure **12.5**
    else
        case $ij \in G$ and $i < i^* \le j$, use figure **12.6**
        case $ij \in G$ and $j < i^* < i$, use figure **12.7**
        otherwise, use figure **12.8**
else ($r^1 > 0$ and $c = 1$)
    use M to decide whether edge $ij \in G$ or not
    use M to compute the predecessor i' of vertex i^* in G
    if $i' =$ no or $i'2^k + i^* < r^1$ **then**
        case $ij \in G$ and $i \le i^* < j$, use figure **13.1**;
        case $ij \in G$ and $j < i^* < i$, use figure **13.2**;
        otherwise, use figure **13.3**
    else if $i'2^k + i^* = r^1$ **then**
        case $i' < i^*$, use figure **13.4**; otherwise, use figure **13.5**
    else
        case $ij \in G$ and $i \le i^* < j$, use figure **13.6**
        case $ij \in G$ and $j < i^* < i$, use figure **13.7**
        otherwise, use figure **13.8**

Fig. 10. Construction of Turing machine K

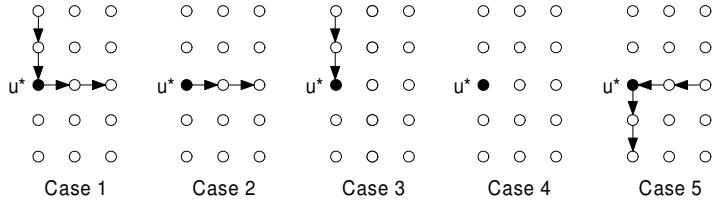

Fig. 11. Cases of $r^1 = 0$

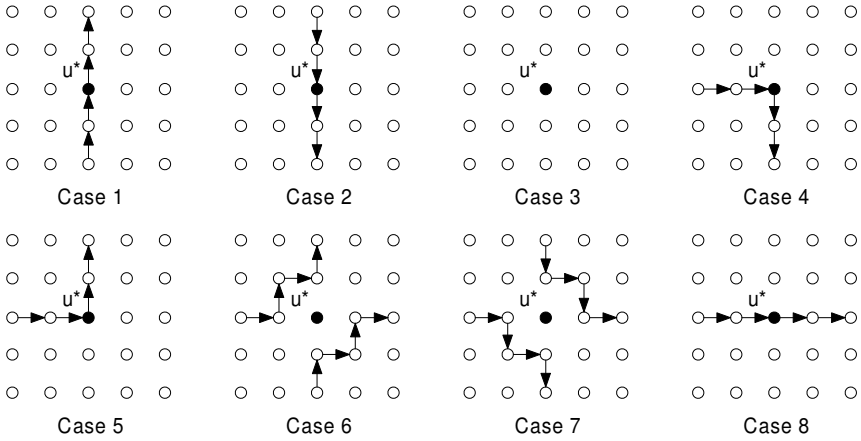

Fig. 12. Cases of $r^1 > 0$ and $c = 0$

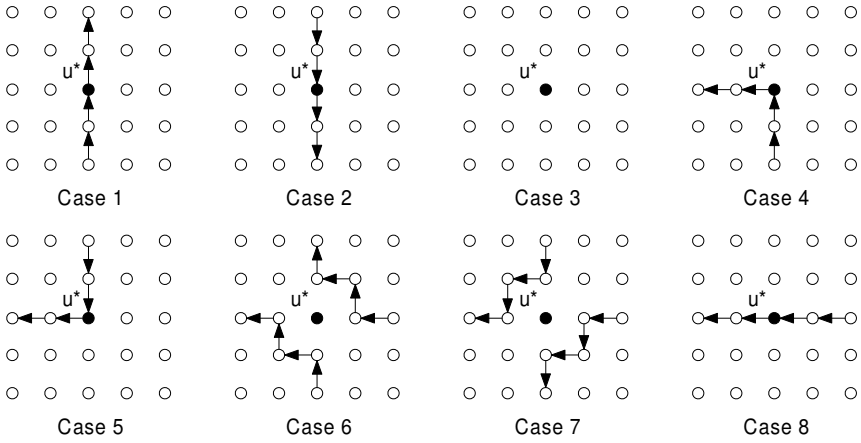

Fig. 13. Cases of $r^1 > 0$ and $c = 1$

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/