The Search Version of 2-Dimensional Sperner's Lemma is PPAD-Complete*

Abbas Mehrabian†

University of Waterloo

April 1, 2011

1 Introduction

Consider a triangulation of a triangle $A_0A_1A_2$, and suppose that you colour each vertex of the triangulation by one of the colours 0, 1, or 2. Such a colouring is called a *Sperner colouring* if

1. The vertex A_i gets colour i for $i = 0, 1, 2$.
2. Each vertex on the segment A_iA_j gets either colour i or colour j, for $0 \leq i < j \leq 2$.

The 2-dimensional Sperner's lemma states that if we have a Sperner colouring, then we will have a *multicoloured* small triangle, i.e. a small triangle whose vertices have distinct colours. It is an existential theorem and does not tell us how quickly can we find such a triangle. In this article we study this problem from algorithmic point of view, and show that, it is unlikely that a fast algorithm exists for finding a multicoloured triangle. Indeed we prove that this is a hard problem in a computational complexity sense.

Most of the material is from the paper

> Xi Chen and Xiaotie Deng, On the Complexity of 2D Discrete Fixed Point Problem, ICALP 2006,

*This is the exposition of my lecture on February 27th, for the course CO 739: Topological Methods in Graph Theory, Winter 2011, University of Waterloo.

†amehrabi@uwaterloo.ca

in which the authors prove that the search problem version of the 2-dimensional Sperner's lemma is PPAD-complete. All the figures that appear here, except for the last one, are copied from the version of the above paper that appeared in the electronic colloquium of computational complexity.

Some preliminaries and the definition of the class PPAD appear in Section 2. In Section 3, we define a search problem RLEAFD that will be used to show our main result, and prove that it is PPAD-complete. We prove the main result in Section 4. In Section 5 we mention some other PPAD-complete problems.

2 Total NP search problems and the class PPAD

In this section we formally define our framework, the class PPAD, and the search version of 2-dimensional Sperner's lemma.

Definition (TFNP)**.** Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a relation satisfying the following properties.

(a) Given strings $x, y \in \{0,1\}^*$, it can be checked in polynomial time if $(x,y) \in R$.

(b) There is a polynomial p such that if $(x,y) \in R$ then $|y| \leq p(|x|)$.

(c) For every $x \in \{0,1\}^*$ there exists at least one $y \in \{0,1\}^*$ with $(x,y) \in R$.

The *total NP search problem* associated with R, denoted by Q_R, is the following problem: given an input $x \in \{0,1\}^*$, find a $y \in \{0,1\}^*$ such that $(x,y) \in R$. The term "total" refers to the condition (c) above. We write *TFNP* for the class of total NP search problems.

Next we define the notion of reductions for search problems.

Definition (reduction)**.** Let Q_A and Q_B be total search problems. The problem Q_A is *polynomial time reducible* to Q_B if there exists a pair (f,g) of polynomial time computable functions such that, for every input x of Q_A, if y satisfies $(f(x),y) \in B$, then $(x, g(y)) \in A$.

Let us see why this definition makes sense. Assume that Q_A is reducible to Q_B, and there is an algorithm $\mathcal{B}$ that solves Q_B in polynomial time. We use this to give a polynomial time algorithm $\mathcal{A}$ for Q_A. The algorithm $\mathcal{A}$ works as follows. Let x be an input to the problem Q_A. $\mathcal{A}$ computes $f(x)$ and runs $\mathcal{B}$ on $f(x)$, and suppose that $y = \mathcal{B}(f(x))$. Next $\mathcal{A}$ computes $g(y)$ and outputs it. By definition, we have $(f(x),y) \in B$, and this gives $(x, g(y)) \in A$, so the algorithm $\mathcal{A}$ works correctly.

Let us fix our graph-theoretic notation. Let G be a directed graph. The in-degree (resp. out-degree) of vertex v is the number of vertices that has an edge to (resp. from) v. The maximum in-degree (resp. maximum out-degree) of vertices of G is written $\Delta^-(G)$ (resp. $\Delta^+(G)$). A *leaf* in G is a vertex whose sum of in-degree and out-degree is 1.

A typical NP search problem, whose totality is based on an easy parity argument, is the following.

Definition (LEAFD, a.k.a. End-Of-The-Line)**.** The search problem *LEAFD* is defined as follows. The input is a pair $(M, 0^k)$, where M is a polynomial time Turing machine generating a directed graph G with vertex set $\{0,1\}^k$ such that $\Delta^-(G) = \Delta^+(G) = 1$. More formally, for any $v \in \{0,1\}^k$, $M(v)$ gives the unique in-neighbour and out-neighbour of v if they exist. The graph G also has the property that, the vertex 0^k has in-degree 0, and there is an edge from 0^k to 1^k. The output is a leaf other than 0^k.

The problem LEAFD is the canonical problem whose totality is based on a parity argument in directed graphs. With this view, the class "Polynomial Parity Arguments on Directed graphs (PPAD)" of search problems was defined by Papadimitriou (On graph theoretic lemmata and complexity classes, FOCS 1990) as the following.

Definition (PPAD)**.** *PPAD* is the set of total NP search problems that are polynomial time reducible to LEAFD.

For a complexity class C, problem P is *complete* for C, or *C-complete*, if P is in C, and any problem in C is reducible to P. From its definition, LEAFD is complete for PPAD.

Now, we define the 2D-SPERNER problem, whose totality is based in 2-dimensional Sperner's lemma.

Definition (2D-SPERNER)**.** Let T_n be the standard $n \times n$ triangulation of a triangle. For example, T_7 is illustrated in Fig. 1. The input to the search problem *2D-SPERNER* is a pair $(F, 0^k)$, where F is a polynomial time Turing machine that produces a Sperner 3-colouring on T_{2^k}. More precisely, each vertex p of T_{2^k} has colour $F(p) \in \{0,1,2\}$. The output is a multicoloured triangle.

3 RLEAFD is PPAD-complete

To prove that 2D-SPERNER is PPAD-complete we need to show that 2D-SPERNER is reducible to LEAFD, and LEAFD is reducible to 2D-SPERNER. We will not directly prove that LEAFD can be reduced to 2D-SPERNER, but we will use an in-between

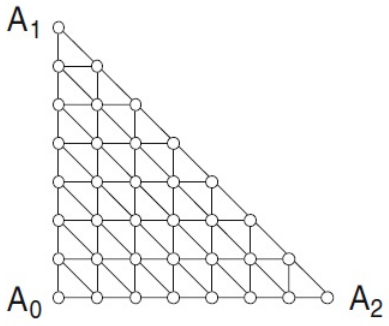


Fig. 1. The standard 7×7 triangulation of a triangle

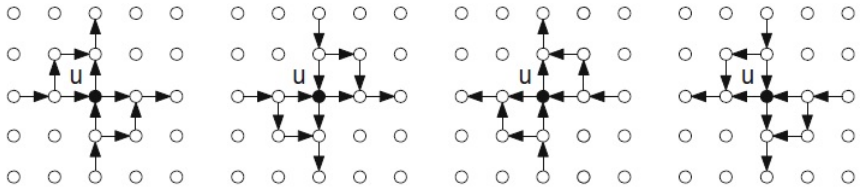


Fig. 2. Summary of cases in the construction of E_n^2

problem called RLEAFD. In this section we define this problem and prove that it is PPAD-complete.

Definition (G_n)**.** The graph G_n is a "grid embedding" of the complete graph on n vertices $0, 1, \ldots, n-1$. Here the term "complete" means that for each $i \neq j$, we have an edge both from i to j and from j to i. Consider a $3n^2 \times 6n$ grid. For each i, $0 \leq i \leq n-1$, vertex i is mapped to the point $(0, 6i)$ of the grid. For every edge ij, we add a path from point $(0, 6i)$ to $(0, 6j)$ in the grid, which is formed by drawing straight lines between the points $(0, 6i), (3(ni + j), 6i), (3(ni + j), 6j + 3), (0, 6j + 3), (0, 6j)$ in the grid. Next, for every point u of the grid that has four incident edges, which corresponds to a crossing of two added paths, we put a gadget around u, as depicted in Fig. 2.

For example, the graph G_3 is shown in Fig. 3.

The graph G_n is rich enough: one can embed any graph with n vertices as a subgraph of G_n on an $3n^2 \times 6n$ grid. However, we would like to work with graphs in which every vertex has at most one in-neighbour and at most one out-neighbour. Hence we define the class C_n to be the set of all subgraphs of G_n that have this property, and in which the vertex $(0, 0)$ has in-degree 0 and out-degree 1 (so it is a leaf). Now we can define the problem RLEAFD.

Definition (RLEAFD)**.** The search problem RLEAFD is defined as follows. The input is

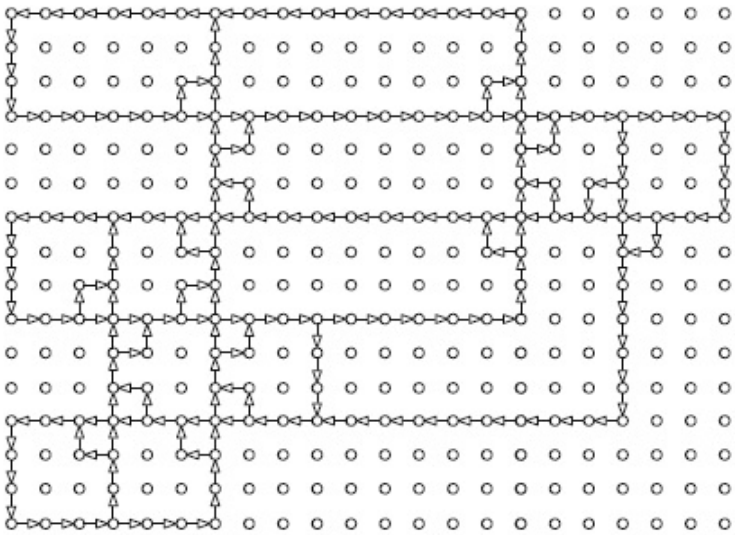


Fig. 3. The planar grid graph G_3

a pair $(K, 0^k)$, where K is a polynomial time Turing machine generating a graph in C_{2^k}, and the output is a leaf other than $(0, 0)$.

Theorem 1. *The problem RLEAFD is PPAD-complete.*

Proof. It is clear that RLEAFD is the same as LEAFD, with a restricted set of input. Hence RLEAFD is reducible to LEAFD, so it is contained in PPAD. To prove the completeness of RLEAFD, we need to show that LEAFD is reducible to RLEAFD.

Let $(M, 0^k)$ be an input instance of LEAFD, and G be the directed graph generated by M. Note that G has 2^k vertices. Let $V(G) = \{0, 1, \ldots, 2^k - 1\}$. We build the graph $C(G)$ as follows. First, embed the graph G as a subgraph of G_{2^k} in a $3 \cdot 2^{2k} \times 6 \cdot 2^k$ grid by mapping vertex i to grid point $(0, 6i)$, adding a grid path between $(0, 6i)$ and $(0, 6j)$ for each edge ij of G (as we did in the definition of G_n), and putting a gadget (see Fig. 2) at each crossing. Next, for each crossing at a grid point u, remove the edges incident to u. For example, if G has vertices $\{0, 1, 2\}$ and edges $\{(0, 2), (2, 1)\}$, then the graph $C(G)$ is shown in Fig. 4.

It is easy to see (but messy to write down) that a Turing machine generating $C(G)$ can be built given a Turing machine generating G. Observe that if G is a valid input for LEAFD, then $C(G) \in C_{2^k}$, i.e. $C(G)$ is a valid input for RLEAFD. The crucial observation is, $C(G)$ 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.

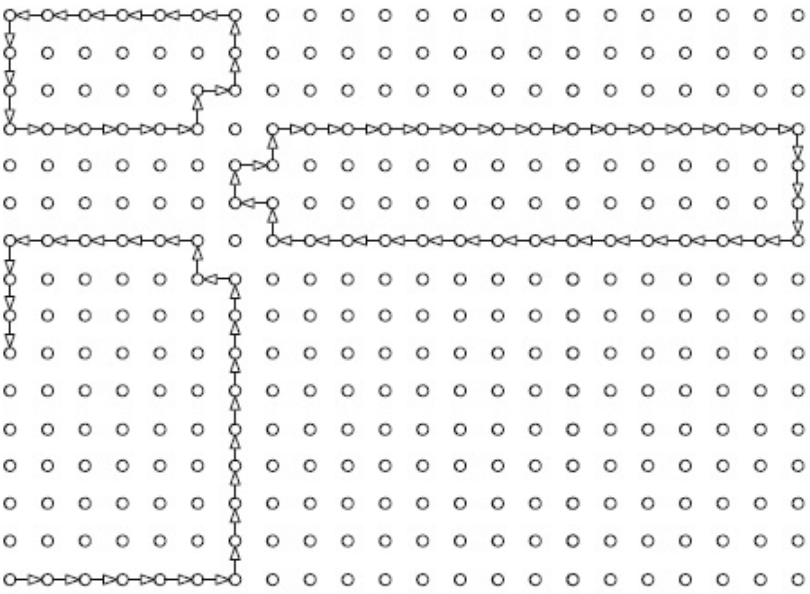


Fig. 4. Graph $C(G) \in C_3$

In other words, vertex i is a leaf in G if and only if grid point $(0, 6i)$ is a leaf in $C(G)$, and any leaf of $C(G)$ is of the form $(0, 6j)$ for some $0 \leq j \leq n-1$. Therefore, given a leaf of $C(G)$, we can locate a leaf of G easily, and this proves that this is indeed a reduction. □

4 2D-SPERNER is PPAD-complete

We prove our main result in this section.

Theorem 2. *The problem 2D-SPERNER is PPAD-complete.*

Proof. First we show that RLEAFD is reducible to 2D-SPERNER, which shows that LEAFD is reducible to 2D-SPERNER. Let $(K, 0^k)$ be an input instance of RLEAFD, and $G \in C_{2^k}$ be the directed graph K generates, which is embedded in a $3 \cdot 2^{2k} \times 6 \cdot 2^k$ grid (for example, G could be the graph in Fig. 4). From G we build a Sperner 3-colouring of $T_{2^{2k+5}}$ (the standard $2^{2k+5} \times 2^{2k+5}$ triangulation) such that a multicoloured triangle gives a leaf of G. We will use the three colours white, black and gray.

To understand the reduction better, it helps to look at an example while reading the details below. Let G_2 be the grid embedding of the complete graph on 2 vertices, which is shown in Fig. 6 (up). The graph $G = C(G_2) \in C_2$ is shown in Fig. 6 (bottom). The

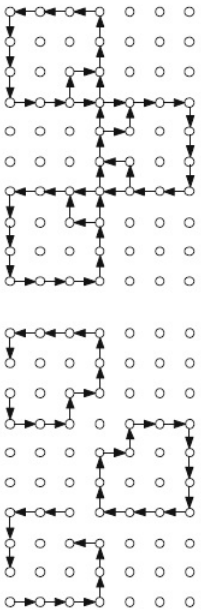


Fig. 6. Graph G_2 and $G \in C_2$

obtained Sperner 3-colouring of T_{128} is shown in Fig. 7, in which for clarity, not all vertices and edges of the triangulation are shown.

Assign coordinates to the points of $T = T_{2^{2k+5}}$ so that the lower-left point is $(0, 0)$, the rightmost points is $(2^{2k+5}, 0)$, and the upmost point is $(0, 2^{2k+5})$. First, we colour all points of T white. We map vertex (u_1, u_2) of G into point $(3u_1 + 3, 3u_2 + 3)$ of T (to get some space between the components). Thus every edge in G corresponds to a path of length 3 in T. Since graph G has $\Delta^-(G) = \Delta^+(G) = 1$, it is a collection of (directed) paths and (directed) cycles.

For every path in G, which corresponds to a (longer) path in T, we colour the points on the path black, but we keep the two endpoints white. When we walk along the path, we colour the points that are immediately to our left gray, and colour the points that are immediately at our right white.

For every cycle in G, which corresponds to a (longer) cycle in T, we colour the points on the cycle black. When we walk along the cycle, we colour the points that are immediately to our left gray, and colour the points that are immediately at our right white.

Next we do some small modifications. First, we do not want the leaf $(0, 0)$ of G correspond to a multicoloured triangle (since we are seeking another leaf), so we just "imagine" that the path in T starting from $(3, 3)$ (corresponding to the $(0, 0)$ leaf of G), actually starts from $(0, 0)$ (which corresponds to nothing in G), then goes to $(0, 3)$, and

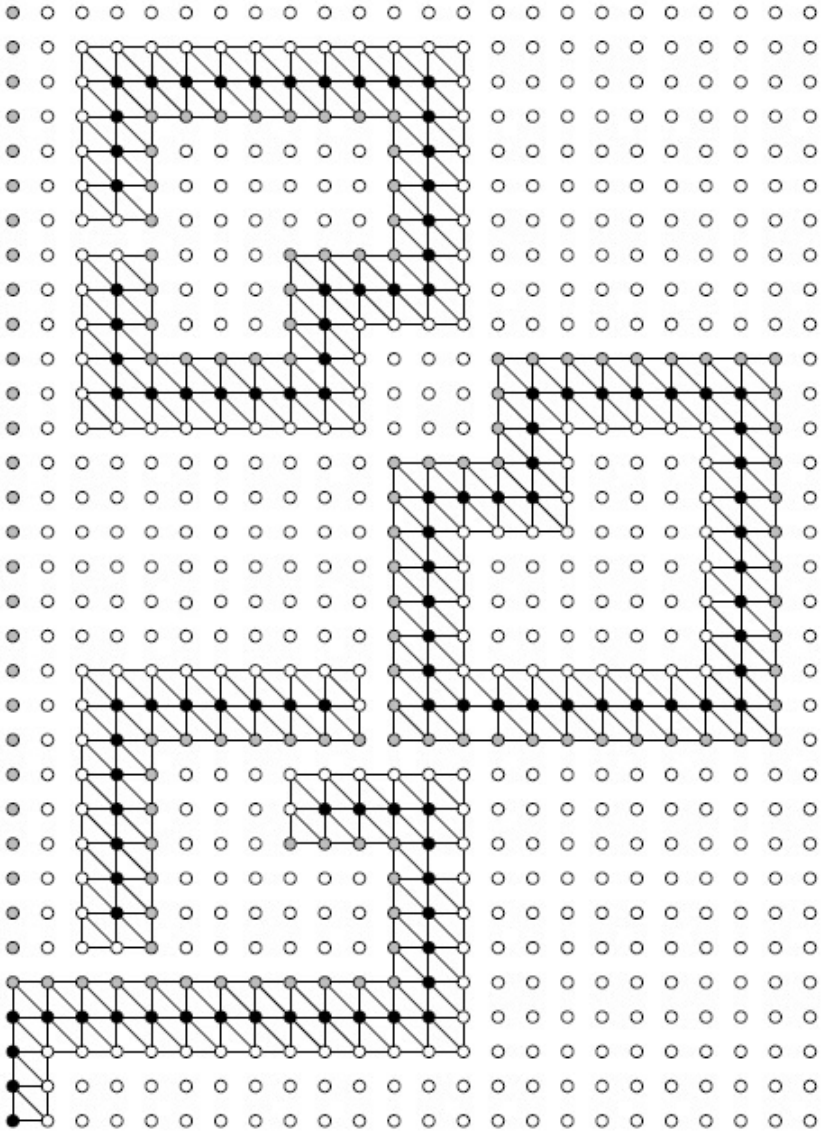


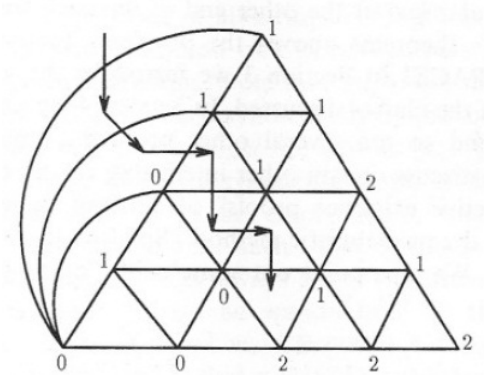
Fig. 7. F: black – 0, gray – 1, white – 2

then goes to $(3,3)$ (look at the lower left part of Fig. 7), and colour the point around this path as described above. Also we colour the vertex $(0,0)$ black, although it is an endpoint of a path. This way we avoid a multicoloured triangle around the point $(3,3)$. Second, we want for the colouring to be a Sperner colouring. The point $(0,0)$ is black, and $(2^{2k+5},0)$ is white. So we colour $(0,2^{2k+5})$ gray, and colour all uncoloured points on the left segment of the triangle gray. This way we get a Sperner colouring of T.

Now, notice that any multicoloured triangle in this 3-colouring corresponds to an endpoint of a path, and so gives a leaf of G.

Conversely, we show that 2D-SPERNER is reducible to LEAFD. The proof is due to Papadimitrou (On the complexity of the parity argument and other inefficient proofs of existence, JCSS, 1994) and the figure below is copied from that article. Consider a Sperner colouring using colours $0,1,2$ of a standard triangulation. We augment the triangulation by adding an edge from the main triangle's vertex coloured 0 to all vertices on the $(0,1)$-segment, except for the vertex to which it is already adjacent (see the figure below). Now, build a directed graph G by putting a vertex in each face (including the outer face) of the resulting inner-triangulated planar graph. Put an edge from vertex u to vertex v in G if, when going from the face of u to the face of v, you should pass through an edge with endpoints coloured 0 and 1, such that the 1 is on your left, and the 0 is on your right (in the figure below, some of the edges are shown).

It is easy to see that every vertex in G has in-degree and out-degree at most 1, and if a vertex is a leaf, then its corresponding face is either the outer face, or a multicoloured triangle. Moreover, the vertex of the outer face is indeed a leaf (since there is no (0,1)-edge on the other two segments of the original triangulation), so G has another leaf, i.e. there exists a multicoloured triangle. But none of the artificial triangles we built on the (0,1)-segment is multicoloured, because the colour 2 does not appear in this segment. Hence any other leaf gives a multicoloured triangle in the original triangulation, and this



completes the proof. □

5 Other PPAD-complete problems

In recent years, the search version of many mathematical theorems, whose proofs are based on some type of parity argument were proved to be PPAD-complete. In this section, the material of which is taken from the homepage of Shiva Kintali[1], we mention some theorems whose corresponding search versions are PPAD-complete (see his homepage for references and more examples).

- The 2-dimensional Brouwer's fixed point theorem
- The 2-dimensional Sperner's lemma (this article)
- Nash's theorem for 2-player games
- Scarf's lemma
- The 2-dimensional Tucker's lemma
- The Borsuk-Ulam's theorem

[1] `http://www.cc.gatech.edu/~kintali/ppad.html`