2-D Tucker is PPA complete

James Aisenberg[a,1], Maria Luisa Bonet[b,2], Sam Buss[c,1,3]

[a] *Dept. of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112, USA*
[b] *Computer Science Department, Universidad Politécnica de Cataluña, Barcelona, Spain*
[c] *Dept. of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112, USA*

Abstract

The 2-D Tucker search problem is shown to be PPA-hard under many-one reductions; therefore it is complete for PPA. The same holds for k-D Tucker for all $k \geq 2$. This corrects a claim in the literature that the Tucker search problem is in PPAD.

Keywords: Tucker lemma, NP search problems, Parity principle, PPA, TFNP

1. Introduction

PPA and PPAD are classes of total NP search problems introduced by Papadimitriou [24]. The class PPA consists of the search problems reducible to the parity principle for undirected graphs, whereas the class PPAD consists of those reducible to the parity principle for directed graphs. The class PPAD has many complete problems from diverse areas of mathematics: Brouwer's theorem and Sperner's lemma in topology [24], Nash equilibria in game theory [9, 6, 8], and others. As discussed by [24, 10], several natural problems are known to be in PPA but not known to be in PPAD. One example is the Smith theorem about Hamiltonian cycles in cubic graphs [25]. Another is the integer factoring problem [3, 20]. However, few natural problems have been shown to be PPA-complete. By definition, the canonical problem LEAF is PPA-complete. For natural topological problems, it has been shown that Sperner's lemma and Tucker's lemma on two-dimensional non-orientable manifolds can be PPA-complete [19, 18, 10]. In addition, Deng et al. [10] show they are PPA-complete in the Möbius band, in two-dimensional projective space, and in the Klein bottle. In contrast, Chen and Deng [7] showed that the two-dimensional Sperner lemma gives a PPAD-complete problem.

In this paper we show that the 2-D TUCKER search problem is PPA-complete. This is the usual Tucker search problem in Euclidean space as defined by Pa-

Email addresses: `jaisenberg@gmail.com` (James Aisenberg), `bonet@cs.upc.edu` (Maria Luisa Bonet), `sbuss@ucsd.edu` (Sam Buss)

[1]Supported in part by NSF grants DMS-1101228 and CCF-1213151.
[2]Supported in part by grant TIN2013-48031-C4-1.
[3]Supported in part by Simons Foundation awards 306202 and 578919.

padimitriou [24]. That paper used an argument by Freund and Todd [17] (a similar argument is given by [22]) to show that TUCKER is in PPA, It also claimed that directionality techniques of Freund [15, 16] can put TUCKER into PPAD. This last part was incorrect; the failure of that argument is discussed more in Section 3. However, the argument in [24] that TUCKER is in PPA is correct; likewise, the proofs in [24] that SPERNER and BROUWER are PPAD-complete are also correct. We stress that the erroneous claim about TUCKER was only a small piece of [24]; our correction does not detract from the significance of that seminal paper.

The 3-D TUCKER search problem was shown in [24] to be hard for PPAD. Subsequently, it was shown that 2-D TUCKER is PPAD-hard [23]. This was extended by [12] to show that k-D TUCKER is PPAD-hard for all fixed $k \geq 2$. We improve these constructions to establish the following:

Theorem 1. 2-D TUCKER *is* PPA-*complete under many-one reductions. The same holds for* k-D TUCKER *for all* $k \geq 2$.

It follows that 2-D TUCKER is in PPAD if and only if PPAD = PPA. In the Type II (oracle) setting, it is known that PPAD $\neq$ PPA [2]. However, it is open whether these classes are equal in the non-relativized (Type I) setting.

We write BORSUK-ULAM for the search problem associated with the Borsuk–Ulam theorem. Since BORSUK-ULAM and TUCKER are many-one reducible to each other[22, 24], another consequence of Theorem 1 is:

Corollary 2. BORSUK-ULAM *is* PPA-*complete.*

We conclude the introduction with a couple of problems that were open when the first version of this paper was circulated in October 2015 [ECCC Report TR15-163]. As discussed below they have subsequently been answered.

The search problems NECKLACE SPLITTING and DISCRETE HAM SANDWICH are known to be many-one reducible to TUCKER [22, 24]. From this, we know they are in PPA; however that leaves open whether they are in PPAD:

Open Question 3. *Is* NECKLACE SPLITTING *in* PPAD*, or* PPA-*complete? Is* DISCRETE HAM SANDWICH *in* PPAD*, or* PPA-*complete? Are they* PPAD-*hard?*

A recent preprint of Filos-Ratsikas and Goldberg [14], building on [13], has answered this (no-longer-)open question by showing both NECKLACE SPLITTING and DISCRETE HAM SANDWICH are PPA-complete. The latter paper uses our Theorem 1 as a tool.

The octahedral Tucker lemma is a special case of the Tucker lemma in which the dimension k varies and the triangulation is the first barycentric subdivision of the k-dimensional hypercube. Thus, the size of the triangulation cannot be increased without also increasing the dimension (and the number of available labels). For the precise statement of the octahedral Tucker lemma, see [21, 26] or [1]. As a special case of TUCKER, the OCTAHEDRAL TUCKER search problem is known from [24] to be in PPA. This left open the following (also asked by [23]):

Open Question 4. *Is* OCTAHEDRAL TUCKER PPA*-complete? Is it in* PPAD*? Is it* PPAD*-hard?*

A recent preprint of Deng, Feng and Kulkarni [11] has resolved this (no-longer-)open question by showing OCTAHEDRAL TUCKER is PPA-complete; their constructions use in part methods from the proof of Theorem 1.

Finally, as already mentioned, it is still open whether problems such as integer factoring, or Smith's theorem on cubic graphs give PPA-complete TFNP search problems. Papadimitriou [24], Grigni [19], and Chen and Deng [7] mention the Smith problem as a candidate for a PPA-complete problem that does not have a Turing machine explicitly encoded in its input.

1.1. Definitions

We briefly review the search problems discussed in this paper. We first state the general form of Tucker's lemma, and then give the "rectangular" 2-D version that we will actually work with. For more information about Tucker's lemma and triangulations, see [22]. Let $B^k \subset \mathbb{R}^k$ be the closed k-dimensional unit ball, and S^{k-1} be its boundary. A triangulation T of B^k is antipodally symmetric if it is antipodally symmetric on the boundary — that is, if each simplex $\sigma \in T \cap S^{k-1}$ has the property that $-\sigma \in T$, where the negation of a simplex is the negation of each of its vertices. The set $V(T)$ of vertices of T is the set of 0-simplices in T.

Theorem 5 (Tucker's lemma). *Let T be an antipodally symmetric triangulation of B^k, and let $\lambda : V(T) \to \{\pm 1, \dots, \pm k\}$ be a function with the property that $\lambda(-v) = -\lambda(v)$ for all $v \in S^{k-1}$. Then there exists a 1-simplex $\{v_1, v_2\}$ in T with $\lambda(v_1) = -\lambda(v_2)$.*

To simplify our constructions, we will work with a rectangular 2-D version of Tucker's lemma, following Pálvölgyi [23]. For m a natural number, define $[m] = \{1, \dots, m\}$.

Definition 6. Let $m \geq 2$. An *instance of the* 2-D TUCKER *search problem* is a function $\lambda : [m] \times [m] \to \{\pm 1, \pm 2\}$ with the property that for $1 \leq i, j \leq m$, $\lambda(i, 1) = -\lambda(m-i+1, m)$ and $\lambda(1, j) = -\lambda(m, m-j+1)$. A *solution* to such an instance of 2-D TUCKER is a pair of vertices (x_1, y_1), (x_2, y_2) with $|x_1 - x_2| \leq 1$ and $|y_1 - y_2| \leq 1$ such that $\lambda(x_1, y_1) = -\lambda(x_2, y_2)$. A solution (x_1, y_1), (x_2, y_2) is called a *complementary pair.*

Two points $(i, 1)$ and $(m-i+1, m)$ are called *antipodal.* Likewise, $(1, j)$ and $(m, m-j+1)$ are *antipodal.*

The $m \times m$ rectangular grid can be triangulated by the addition of diagonals, so it is clear that the existence of a solution to the 2-D TUCKER search problem is guaranteed by Tucker's lemma.

Definition 7. An *instance of the* LEAF *search problem* is an undirected graph G where each node has degree at most 2, and there is a given ("standard") leaf ℓ with degree 1. A *solution* to LEAF is any other node of G with degree 1.

The class PPA is the set of total NP search problems reducible to Leaf under polynomial time many-one reductions [24]. In this paper, we envision 2-D Tucker and Leaf as Type II search problems in the sense of [2]. This means that instances of the search problems are exponentially big and are given by oracles: For 2-D Tucker, the oracle specifies the values of the function λ. For Leaf, the oracle specifies the neighbors of any given node.

The Type II formulation uses an oracle to specify the input; this is sometimes called a "black-box" model. In contrast, the Type I formulation of [2] uses a "white-box" model; e.g., the exponentially large input is specified with an explicitly given polynomial size circuit. In the Type II setting, it is known that PPAD is a proper subset of PPA [2]. We conjecture the same holds in the Type I setting, but this is open: For instance, if P = NP, then PPAD and PPA both collapse to P. Our results, including Theorems 1 and 8, hold in the Type II setting; consequently they also hold for the Type I formulations.

2. Reduction from Leaf

We now show that 2-D Tucker is PPA-hard. Since 2-D Tucker is in PPA, this suffices to establish Theorem 1.

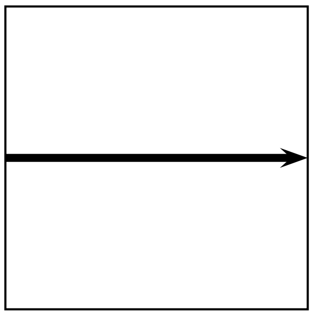
Theorem 8. 2-D Tucker *is* PPA*-hard under many-one reductions.*

Proof. We give a reduction from Leaf. Let G be an instance of Leaf. We will describe λ, a labelling of the $m \times m$ grid with labels $\{\pm 1, \pm 2\}$. We will take $m = 4 \cdot 13 \cdot |G|$, where $|G|$ is the number of nodes in G. Our task is to define the values of $\lambda(i,j)$ for (i,j) a point on the $m \times m$ rectangular grid. The domain of λ will be referred to as *the grid*, and points (i,j) on the grid will be called *grid nodes.*

The reduction is similar to constructions of Papadimitriou [24] and especially Pálvölgyi [23]. The vast majority of the grid will be labelled with 1's (this is called the "environment"). The remainder of the grid will be filled with "wires": a wire consists of a strip of grid nodes of width three; the central "conductor" has label -1 and "insulators" on either side have labels ± 2. Wires are always directional. When travelling in the forward direction, the insulator on the left always has label 2, and the insulator on the right always has label -2.

We generally avoid exposing the conductor to the environment, as this would create complementary pairs between the conductor (-1) and the environment (1). We will route the wire in such a way that regions corresponding to solutions of G are the only wires exposed to the environment.

The grid is partitioned into 13×13 squares called *tiles.* A tile on the boundary is called a *boundary tile.* Two boundary tiles are *antipodal* if one of them contains some grid nodes antipodal to some grid nodes in the other. Specifically, this happens when the right column (resp., top row) of nodes in one tile are antipodal to the left column (resp., bottom row) of nodes in the other tile. In this case, since λ must be antipodal, the λ values of the nodes in the right


(a) Schematic representation

	1	2	3	4	5	6	7	8	9	10	11	12	13	
1														1
2														2
3														3
4														4
5														5
6	2	2	2	2	2	2	2	2	2	2	2	2	2	6
7	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	7
8	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	8
9														9
10														10
11														11
12														12
13														13
	1	2	3	4	5	6	7	8	9	10	11	12	13	

(b) Realization

Figure 1: A horizontal wire. (a) shows the schematic representation. (b) shows its realization with values of the labelling λ. The center of the wire has labels -1; the insulator labels 2 are on the left-hand side of the wire as it is traversed in its forward direction. The blank space represents grid nodes with label values of 1.

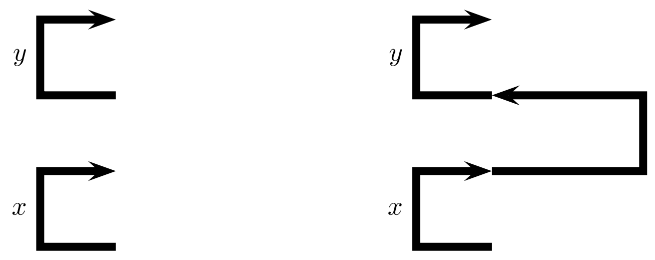
column (resp., top row) of the first tile are the negations of the λ values of the nodes in the left column (resp., bottow row) in reverse order.

The schematic representation and its realization on the grid of a horizontal wire are shown in Figure 1. In figures representing the grid, 1's are indicated with blank space. The tile for the horizontal wire in the opposite direction can be obtained from the tile in Figure 1 by rotating 180°, or alternatively by reflecting about the horizontal axis. The tiles for the vertical wires can be obtained by rotating the horizontal ones 90°. Our tiles will typically have the conductor meet the edge of the tile at row 7 or column 7.

Notice that two wires can be in adjacent tiles without creating a complementary pair as long as they either are parallel or are joined head to tail. However, wires joined head to head or tail to tail do create complementary pairs, because the insulator labelled 2 is adjacent to the insulator labelled -2.

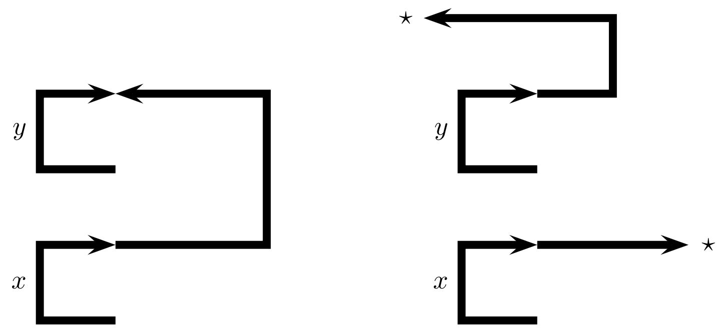
Recall that one node of G is given as the *standard* leaf ℓ, a degree 1 node. All other nodes $x, y, \ldots$ of G have degree ≤ 2; those of degree 1 are solutions to G as an instance of Leaf. Each node of G other than ℓ is assigned a region in the grid with two exposed edges: the *inbound edge* and the *outbound edge*, as pictured in Figure 2(a). The idea for our construction is that, when x has degree 2, the two exposed edges of x are wired to the edges of the two neighbors of x. If x has degree 0, its inbound and outbound edges are connected to each other. If x has only one neighbor, then one edge of x is exposed to the environment, creating a complementary pair. This is the only way that a complementary pair is formed; thus any complementary pair for λ corresponds to a solution to the instance G of Leaf.

Sometimes we are able to attach an outbound edge of a node x to an inbound



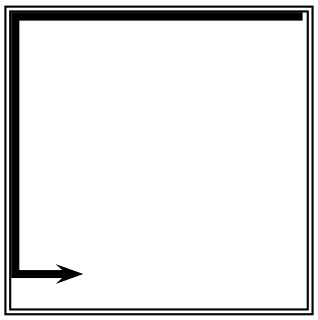
(a) Two nodes x and y (b) Connection of outbound to inbound

Figure 2: Two nodes and their connection. (a) Each node of G is assigned a region in the grid with an inbound edge and an outbound edge. (b) The schematic representation of connecting the outbound edge of x to the inbound edge of y.



(a) Incorrect connection (b) Correct connection

Figure 3: The outbound edge of x is connected to the outbound edge of y. When the boundary is crossed, the wire direction is reversed. The two locations in (b) marked with $\star$ are antipodal on the boundary.



(a) Schematic representation

	1	2	3	4	5	6	7	8	9	10	
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1
2	-1	2	2	2	2	2	2	2	2	2	2
3	-1	2								1	3
4	-1	2								1	4
5	-1	2								1	5
6	-1	2								1	6
7	-1	2	2							1	7
8	-1	-1	-1							1	8
9	-2	-2	-2							1	9
10	1	1	1	1	1	1	1	1	1	1	10
	1	2	3	4	5	6	7	8	9	10	

(b) Realization (not to scale)

Figure 4: The boundary with no crossings. The figure shows a 10×10 grid, but in actuality it is an $m \times m$ grid. For this reason, we say that it is "not to scale".

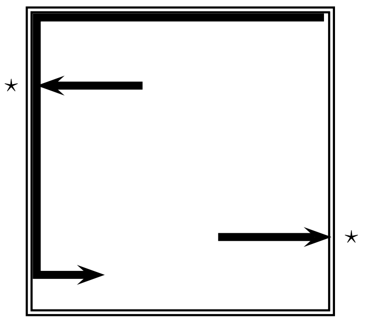
edge of a neighboring node y. This is pictured schematically in Figure 2(a). However, since G is undirected, we will sometimes need to connect an outbound edge of x to an outbound edge of y. As shown in Figure 3(a), this creates unwanted complementary pairs. We thus use instead the construction shown in Figure 3(b). The outbound edge of x is routed "across the boundary", where it reverses direction (we shall see in Figure 5 how the reversal works), and then continues on to meet the outbound edge of y. A similar construction works to join an inbound edge of x to an inbound edge of y.

The rest of the proof shows how to apply the ideas behind the schematic representations shown in Figures 2(b) and 3(b) to define the labelling λ. For this, we must describe how the boundary is labelled, how a wire can cross the boundary and reverse direction, how two wires can cross each other in the grid, and the global strategy for routing wires.

First, we consider how to label the boundary of the grid, while preserving the antipodal property of λ. The underlying construction is shown in Figure 4; however it will need modification for wires that cross the boundary (as in Figures 3(b), 5 and 7). The boundary is represented by a double line in the figures. As shown in Figure 4, the outbound edge for the standard leaf ℓ emerges out the lower-left corner of the grid. The standard leaf, being of degree 1 in G, has only an outbound edge and no inbound edge. For simplicity, Figure 4(b) is shown scaled down to be 10×10 instead of its actual size of $m \times m$.

Let's describe the details of how a wire crosses the boundary and reverses direction. For this, refer first to Figures 3(b) and 5. There is a wire pointing to the right exiting the right boundary, and a wire pointing to the left exiting the left boundary. Recall that blank space indicates label values 1; thus, by examination, the antipodal property of λ holds on the boundary.

Figure 5 is "not to scale", and shows only label values needed for the wire crossing the boundary. The wire exiting to the left in Figure 5 is shown again inside its 13×13 tile in Figure 6. Note that it jogs downward two rows. This is

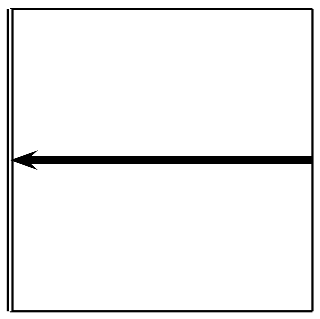


(a) Schematic representation

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1
2	-1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	-1	2																		1	3
4	-1	2																		1	4
5	-1	2																		1	5
6	-1	2																		1	6
7	2	2			-2	-2	-2	-2	-2											1	7
$\star$ 8					-2	-1	-1	-1	-1											1	8
9	-2	-2	-2	-2	-2	-1	2	2	2											1	9
10	-1	-1	-1	-1	-1	-1	2													1	10
11	-1	2	2	2	2	2	2													1	11
12	-1	2												2	2	2	2	2	2	2	12
13	-1	2												-1	-1	-1	-1	-1	-1	-1	13 $\star$
14	-1	2												-2	-2	-2	-2	-2	-2	-2	14
15	-1	2																		1	15
16	-1	2																		1	16
17	-1	2	2	2	2															1	17
18	-1	-1	-1	-1	-1															1	18
19	-2	-2	-2	-2	-2															1	19
20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	20
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	

(b) Realization (not to scale)

Figure 5: A wire crossing the boundary for joining two outbound edges. The realization (b) is "not to scale", and shown as 20×20. In actuality it is $m \times m$, and row 8 on the left is a row number i, and row 13 on the right is the antipodal point on row $m+1-i$. The wire jogs down two rows as it reaches the left boundary so as to make the antipodal property hold.



(a) Schematic representation

	1	2	3	4	5	6	7	8	9	10	11	12	13	
1	-1	2												1
2	-1	2												2
3	-1	2												3
4	-1	2												4
5	-1	2												5
6	2	2				-2	-2	-2	-2	-2	-2	-2	-2	6
7						-2	-1	-1	-1	-1	-1	-1	-1	7
8	-2	-2	-2	-2	-2	-2	-1	2	2	2	2	2	2	8
9	-1	-1	-1	-1	-1	-1	-1	2						9
10	-1	2	2	2	2	2	2	2						10
11	-1	2												11
12	-1	2												12
13	-1	2												13
	1	2	3	4	5	6	7	8	9	10	11	12	13	

(b) Realization

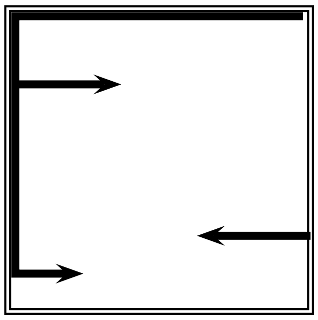
Figure 6: A boundary crossing tile

to maintain the convention that the conductor of a wire, which is labelled -1, is in the middle row of its tile. The $\star$'s in Figure 5(b) mark the middle rows of antipodal tiles, thus antipodal boundary points of the grid. The left exiting wire, exiting from the antipodal tile, has label value 1 (not -1) on the middle row in the leftmost column. Figure 6 shows how this is implemented inside a 13×13 tile. The right column of Figure 6 has -1 in its middle position, so as to correctly match up with the continuation of the wire into the adjacent tile.

A similar construction allows wires to cross the boundary in the opposite direction. This is shown in Figures 7 and 8.

Since we are routing wires in a two-dimensional grid, wires will need to "cross each other". For this, following [5, 7, 23, 4], we use the "avoided crossing" construction shown in Figure 9. We also need to let wires turn at right angles; this is very simple and shown in Figure 10.

We will now describe the global layout of the grid. Fix a total order $<$ on the nodes of G, with the standard leaf ℓ as the least element. The nodes are arranged vertically in the lower-left quadrant of the grid according to the total order. The grid will need to accomodate routing wires between inbound and/or outbound edges of vertices. These wires are routed along "lanes" in a manner that will allow a simple description of the labelling (the λ values) of the grid nodes. Each inbound and outbound edge of each node has a horizontal lane that extends to the right boundary. At the tile antipodal to where this lane reaches the right boundary, a second horizontal lane continues in the upper half of the grid; this second horizontal lane allows a wire to wrap around from right-to-left, and at the same time reverse its direction. Each inbound and outbound edge also has a vertical lane that extends from the top boundary to the bottom boundary. Unlike the horizontal lanes, the vertical lanes do not allow wires to wrap around. The layout of the grid for a graph with three nodes is shown in

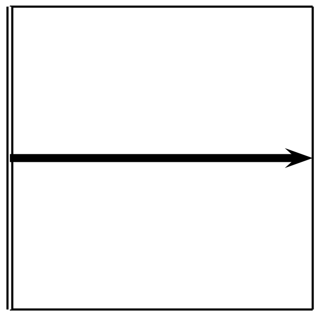


(a) Schematic representation

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1
2	-1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	-1	2																		1	3
4	-1	2																		1	4
5	-1	2	2	2	2	2	2	2	2											1	5
6	-1	-1	-1	-1	-1	-1	-1	-1	2											1	6
7	-2	-2	-2	-2	-2	-2	-2	-1	2	2	2									1	7
⋆ 8							-2	-1	-1	-1	-1									1	8
9	2	2					-2	-2	-2	-2	-2									1	9
10	-1	2																		1	10
11	-1	2																		1	11
12	-1	2												-2	-2	-2	-2	-2	-2	-2	12
13	-1	2												-1	-1	-1	-1	-1	-1	-1	13 ⋆
14	-1	2												2	2	2	2	2	2	2	14
15	-1	2																		1	15
16	-1	2																		1	16
17	-1	2	2	2	2															1	17
18	-1	-1	-1	-1	-1															1	18
19	-2	-2	-2	-2	-2															1	19
20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	20
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	

(b) Realization (not to scale)

Figure 7: A wire crossing the boundary for joining two inbound edges.

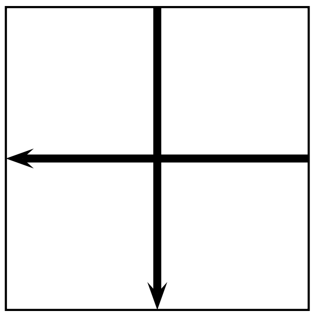


(a) Schematic representation

	1	2	3	4	5	6	7	8	9	10	11	12	13	
1	-1	2												1
2	-1	2												2
3	-1	2												3
4	-1	2	2	2	2	2	2	2						4
5	-1	-1	-1	-1	-1	-1	-1	2						5
6	-2	-2	-2	-2	-2	-2	-1	2	2	2	2	2	2	6
7						-2	-1	-1	-1	-1	-1	-1	-1	7
8	2	2				-2	-2	-2	-2	-2	-2	-2	-2	8
9	-1	2												9
10	-1	2												10
11	-1	2												11
12	-1	2												12
13	-1	2												13
	1	2	3	4	5	6	7	8	9	10	11	12	13	

(b) Realization

Figure 8: A boundary crossing tile

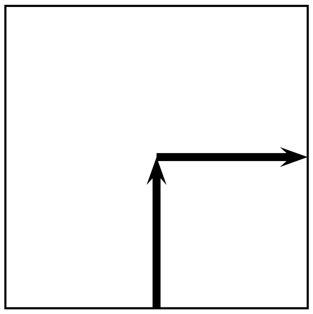


(a) Schematic representation

	1	2	3	4	5	6	7	8	9	10	11	12	13	
1						-2	-1	2						1
2		-2	-2	-2	-2	-2	-1	2						2
3		-2	-1	-1	-1	-1	-1	2						3
4		-2	-1	2	2	2	2	2						4
5		-2	-1	2										5
6	-2	-2	-1	2						-2	-2	-2	-2	6
7	-1	-1	-1	2						-2	-1	-1	-1	7
8	2	2	2	2						-2	-1	2	2	8
9										-2	-1	2		9
10						-2	-2	-2	-2	-2	-1	2		10
11						-2	-1	-1	-1	-1	-1	2		11
12						-2	-1	2	2	2	2	2		12
13						-2	-1	2						13
	1	2	3	4	5	6	7	8	9	10	11	12	13	

(b) Realization

Figure 9: An avoided crossing. This effectively allows wires to cross each other.


(a) Schematic representation

	1	2	3	4	5	6	7	8	9	10	11	12	13	
1														1
2														2
3														3
4														4
5														5
6						2	2	2	2	2	2	2	2	6
7						2	-1	-1	-1	-1	-1	-1	-1	7
8						2	-1	-2	-2	-2	-2	-2	-2	8
9						2	-1	-2						9
10						2	-1	-2						10
11						2	-1	-2						11
12						2	-1	-2						12
13						2	-1	-2						13
	1	2	3	4	5	6	7	8	9	10	11	12	13	

Figure 10: A right angle

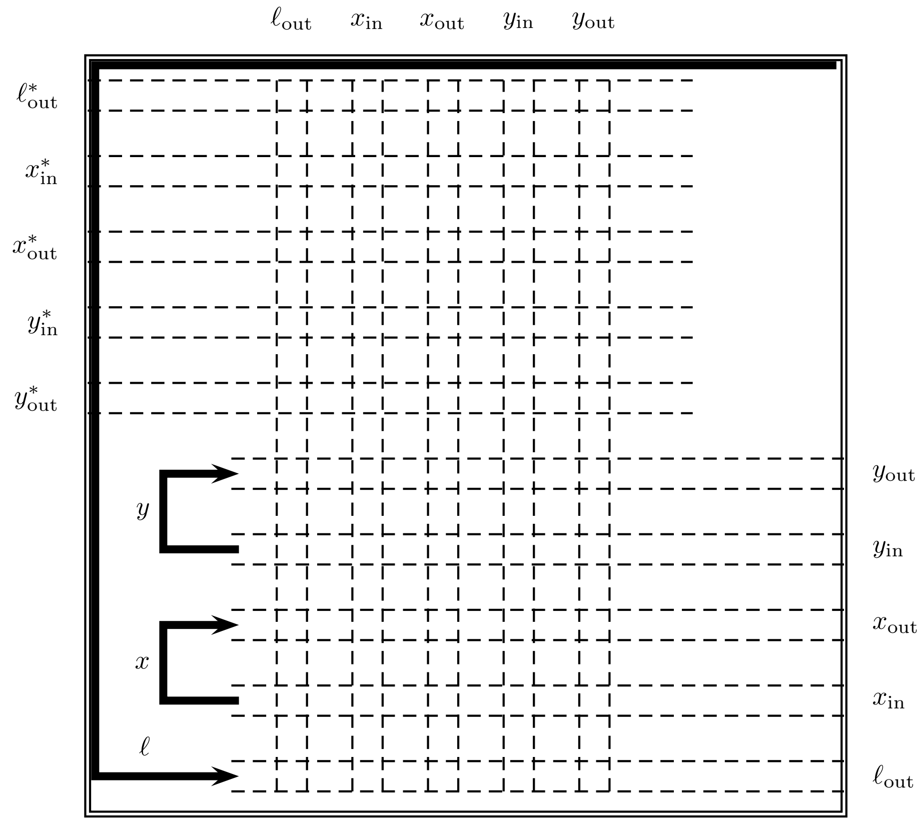


Figure 11: Global layout of the grid.

Figure 11. For example, consider the inbound edge of vertex x in that figure. There is a horizontal lane, indicated with x_{in}, extending from x's inbound edge to the right boundary. It continues, via an antipodal transition, to the second horizontal lane x^*_{in}. The inbound edge of x has in addition a vertical lane, also indicated as x_{in} in the figure.

Each node of G has four horizontal lanes and two vertical lanes; since the lanes have width 13 and since $m = 4 \cdot 13 \cdot |G|$, the grid has sufficient space to hold the construction.

We will now describe how nodes are connected together. When x and y are neighbors in G, we will connect one edge of x in the grid with one edge of y in the grid. For this, we select either the outbound or inbound edge of x and either the outbound or inbound edge of y. This works even though G is undirected.

1. If x is a node in G with two neighbors y and z, with $y < z$, then the outbound edge of x connects to y and the inbound edge of x connects to z.
2. If x is a node with no neighbors, then the outbound edge of x connects to the inbound edge of x.
3. If x is the standard leaf ℓ, then the outbound edge of x connects to its one neighbor y. In this case, x has no inbound edge.
4. If x is a node that is not the standard leaf with only one neighbor y, then the outbound edge of x connects to y, and the inbound edge of x is exposed to the environment. This will create a complementary pair at x's inbound edge as desired.

Suppose that x and y are neighbors in G, with $x < y$. We will describe how x and y are connected together:

1. If x's outbound edge connects to y's inbound edge, then we add a wire that takes the following route: x's outbound edge, x's horizontal outbound lane, x's vertical outbound lane, y's horizontal inbound lane, and finally y's inbound edge.
2. If y's outbound edge connects to x's inbound edge, then we add a wire that takes the following route: y's outbound edge, y's horizontal outbound lane, x's vertical inbound lane, x's horizontal inbound lane, and finally x's inbound edge.
3. If x's and y's outbound edges connect together, then half of the route is as follows: start at x's outbound edge, continue along x's horizontal outbound edge to the boundary. The other half of the route is as follows: start at y's outbound edge, continue along y's horizontal outbound lane to x's vertical outbound lane. Follow x's vertical outbound lane up to x's reflected outbound horizontal lane. Continue along x's reflected outbound horizontal lane to the boundary.
4. If x's and y's inbound edges are connected together, then one path originates from the boundary at x's horizontal inbound lane into x's inbound edge. The other path originates at the antipodal boundary point, travels along x's reflected horizontal inbound path to x's vertical inbound lane, down to y's horizontal inbound lane, and into y's inbound edge.

If x is a node of G with no neighbors, then we must connect the outbound edge of x to the inbound edge of x. This is done by the following route: x's outbound edge to x's horizontal outbound lane, to x's vertical outbound lane, to x's horizontal inbound lane, to x's inbound edge.

The paths formed by the above procedure can cross each other: if so, we use the avoided crossing construction. By inspection, at most two paths can intersect a given tile, and if so, they meet at right angles.

Claim 9. *The only complementary pairs in the grid that are formed by the above construction are at the inbound edge of a node $x \neq \ell$ of degree 1 in G.*

Claim 9 is obvious by inspection of the construction. It follows that there is a polynomial time method to find a degree 1 node $x \neq \ell$ in G, given the location of a complementary pair for λ in the grid.

Claim 10. *It is possible to decide in polynomial time which tile to place at a given position in the grid using only constantly many oracle queries to G.*

Claim 10 follows from the fact that a given tile can lie in at most two "lanes". To illustrate this, consider the following example. Consider a tile that is at the intersection of x's horizontal inbound lane, and y's vertical outbound lane. We query G about x's neighbors which, say, are $u_1 < u_2$. Thus the inbound edge of x connects to u_2. We then query G about u_2's neighbors in order to decide if x connects to u_2 at u_2's inbound or outbound edge. With this information, we can decide if the route taken on x's horizontal inbound lane passes through the tile, does not pass through this tile, or turns at a right angle at the tile. We will similarly query G about y's two neighbors, say $v_1 < v_2$, and then query G about v_1's neighbors. This is enough to determine what happens in the vertical lane. With all this information, we can decide how to assign λ values for this tile, namely as a blank tile, a horizontal wire, a vertical wire, a right angle, or an avoided crossing. This is accomplished with only queries to only four nodes G.

This completes the proof of Theorem 8 and hence Theorem 1.

3. Tucker, Leaf, and LeafD

This section gives a quick sketch of the reduction from Tucker to Leaf. The constructions are due to Freund [15, 16], Freund and Todd [17], Matoušek [22], and Papadimitriou [24]. However, it seems useful to repeat the arguments here to illustrate the reduction from Tucker to Leaf, as well as to point out why it does not give a reduction to the directed analogue LeafD of Leaf.[4] This illustrates the failure in the earlier argument that Tucker is

[4]LeafD is one of the canonical complete problems used for the definition of PPAD [24]. An instance of LeafD is a directed graph with the following properties: each vertex has in-degree at most one and out-degree at most one, and there is a distinguished vertex (usually vertex 0) with in-degree zero and out-degree one. A solution to the instance of LeafD is a vertex which has total degree (in-degree plus out-degree) equal to one.

PPAD-complete. As we shall see, the reduction gives a graph G in which many of the edges can be coherently directed, but edges which connect antipodal simplices cannot be coherently directed.

Let a triangulation T of the unit ball in the L^1-norm and a labelling λ satisfy the hypotheses of the Tucker lemma in dimension 2. Further suppose (for sake of a contradiction) that there are no complementary 1-simplices in T. A 1-simplex in T is just an edge in T. Without loss of generality, refining T if necessary, we may assume that the triangulation contains the origin, and no 1-simplex in T has endpoints in distinct quadrants. A simplex is defined to be *happy* if it is a 1-simplex and certain labels are present on its vertices, according to what region the simplex lies in, as given by the following table:

Midpoint of 1-simplex lies in	Required labels (λ values)
Positive x-axis	1 and one of $1, 2, -2$
Negative x-axis	-1 and one of $-1, 2, -2$
Positive y-axis	2 and one of $1, -1, 2$
Negative y-axis	-2 and one of $1, -1, -2$
First quadrant (interior)	1 and 2
Second quadrant (interior)	-1 and 2
Third quadrant (interior)	-1 and -2
Fourth quadrant (interior)	1 and -2

By our assumptions on T, each 1-simplex has its interior lying in exactly one region. Without loss of generality, the origin has label 1, so the 1-simplex that lies in the positive x-axis with one endpoint at the origin is happy. This 1-simplex is called the *initial* 1-simplex.

A graph G is defined on the happy 1-simplices of T. The initial 1-simplex is a node of degree 1. All other happy 1-simplices will have degree 2 in G (this holds because of the assumption that there are no complementary 1-simplices). The graph G is undirected; nonetheless, many (but not all) of its edges can be coherently directed. The different types of directed edges between happy 1-simplices are as shown in Figures 12 and 13: the curved arrows connect happy 1-simplicies; the arrows indicate the directions. For example, an edge in G connecting two happy 1-simplices in the first quadrant is directed so that the vertex with label 1 is on the left, and the vertex with label 2 is on the right. Two adjacent happy 1-simplices that lie on an axis have their edge directed away from the origin (e.g., rightward on the positive x-axis, leftward on the negative x-axis, etc.).

There are additional undirected edges between antipodal happy 1-simplices which lie on the boundary of the ball. These are as follows:

1. If σ is a happy 1-simplex, and both vertices of σ are on the boundary, then σ has $-\sigma$ as a neighbor in G. Note $-\sigma$ is happy, since σ is. An example is illustrated in Figure 13 with a dashed curve.
2. If σ is happy, σ lies in a 1-dimensional region (the x- or y-axis), one of σ's vertices v is on the boundary of the ball, and v has the required label

which by itself is sufficient to make σ happy, then σ has neighbor τ, the unique (and happy) 1-simplex lying in a 1-dimensional region that has $-v$ as a vertex. This case applies when the vertex v is at the boundary point on the positive x (respectively, negative x, positive y or negative y) axis, and v's label is 1 (respectively, -1, 2 or -2).

Under the assumption that there are no complementary 1-simplices, a straightforward case analysis shows that the initial 1-simplex has degree 1 in G, and all other nodes have degree 2.

Since the construction of G from T is constructive, and the presence of edges in G only depends locally on T, the above gives a many-one polynomial time reduction from from TUCKER to LEAF in the case of two dimensions. Higher dimensions work analogously, but require considering k-simplices also for $k > 1$.

The edges in G that are not on the boundary of the ball can be coherently oriented as illustrated in Figure 12. This can be easily checked in the two dimensional case, and the general case is carried out by Freund [15, 16]. However, the undirected edges connecting antipodal simplices cannot be directed coherently. For example, the dashed curve of Figure 12 cannot be directed without creating a 1-simplex with two incoming edges in G. It was exactly this ability to "reverse directions" by connecting antipodal simplices that was exploited in the proof of Theorem 1.

Acknowledgements. We thank Christos Papadimitriou, Dömötör Pálvölgyi, and Xiaotie Deng for comments on early drafts of this paper, and the referee for an insightful reading and helpful comments.

References

[1] James Aisenberg, Maria Luisa Bonet, Sam Buss, Adrian Crăciun, and Gabriel Istrate. Short proofs of the Kneser-Lovász coloring principle. *Information and Computation*, 261(2):296–301, 2018.

[2] Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *Journal of Computer and System Sciences*, 57(1):3–19, 1998.

[3] Josh Buresh-Oppenheim. On the TFNP complexity of factoring. unpublished manuscript, www.cs.toronto.edu/∼bureshop/factor.pdf, 2006.

[4] Samuel R. Buss. Polynomial-size Frege and resolution proofs of *st*-connectivity and Hex tautologies. *Theoretical Computer Science*, 1-3:35–52, 2006.

[5] Xi Chen and Xiaotie Deng. On the complexity of 2D discrete fixed point problem (extended abstract). In *Proc. Intl. Colloquium on Automata, Languages, and Programming (ICALP), Part I*, Lecture Notes in Computer Science 4051, pages 489–500. Springer, 2006.

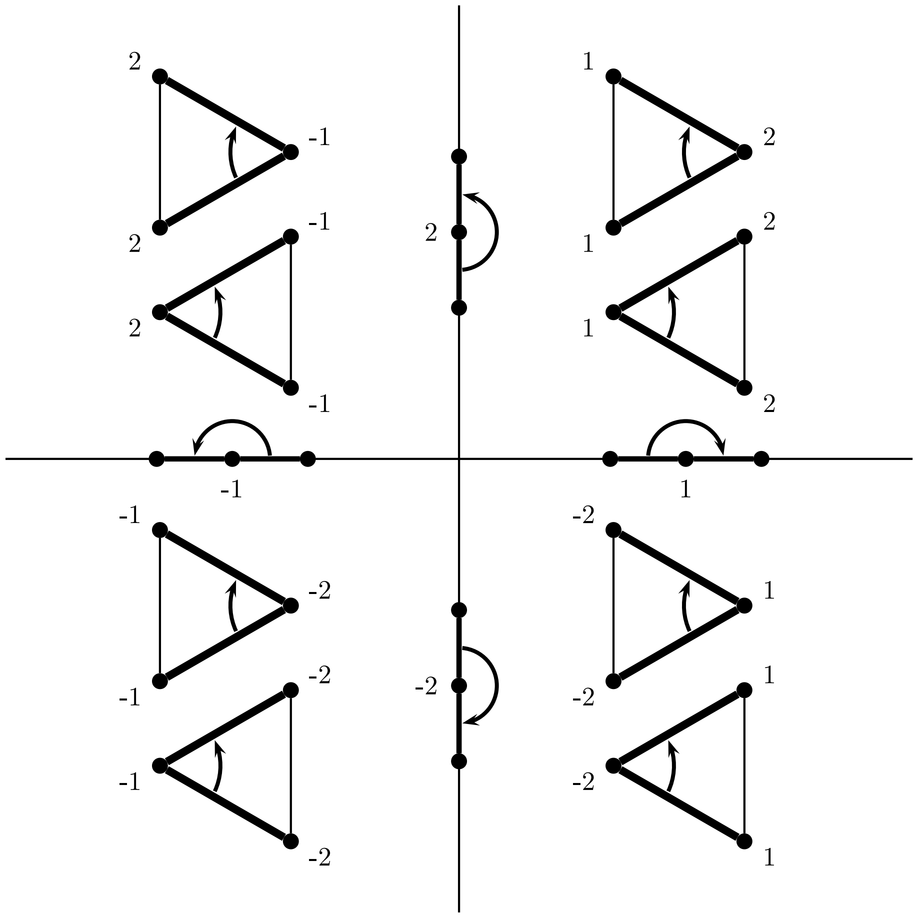


Figure 12: Happy 1-simplices, and their directed neighbors. 1-simplices which are happy are drawn with thick lines. Happy 1-simplices in quadrant I, II, III or IV (respectively) have their vertices labelled with a 1 and 2, with a −1 and 2, with a −1 and −2, or with a 1 and −2 (respectively). A 1-simplex in the positive x-axis, the positive y-axis, the negative x-axis, or the negative y-axis (respectively) have at least one vertex labelled 1, 2, −1, or −2 (respectively). The directed edges between happy vertices are shown by the curved arrows.

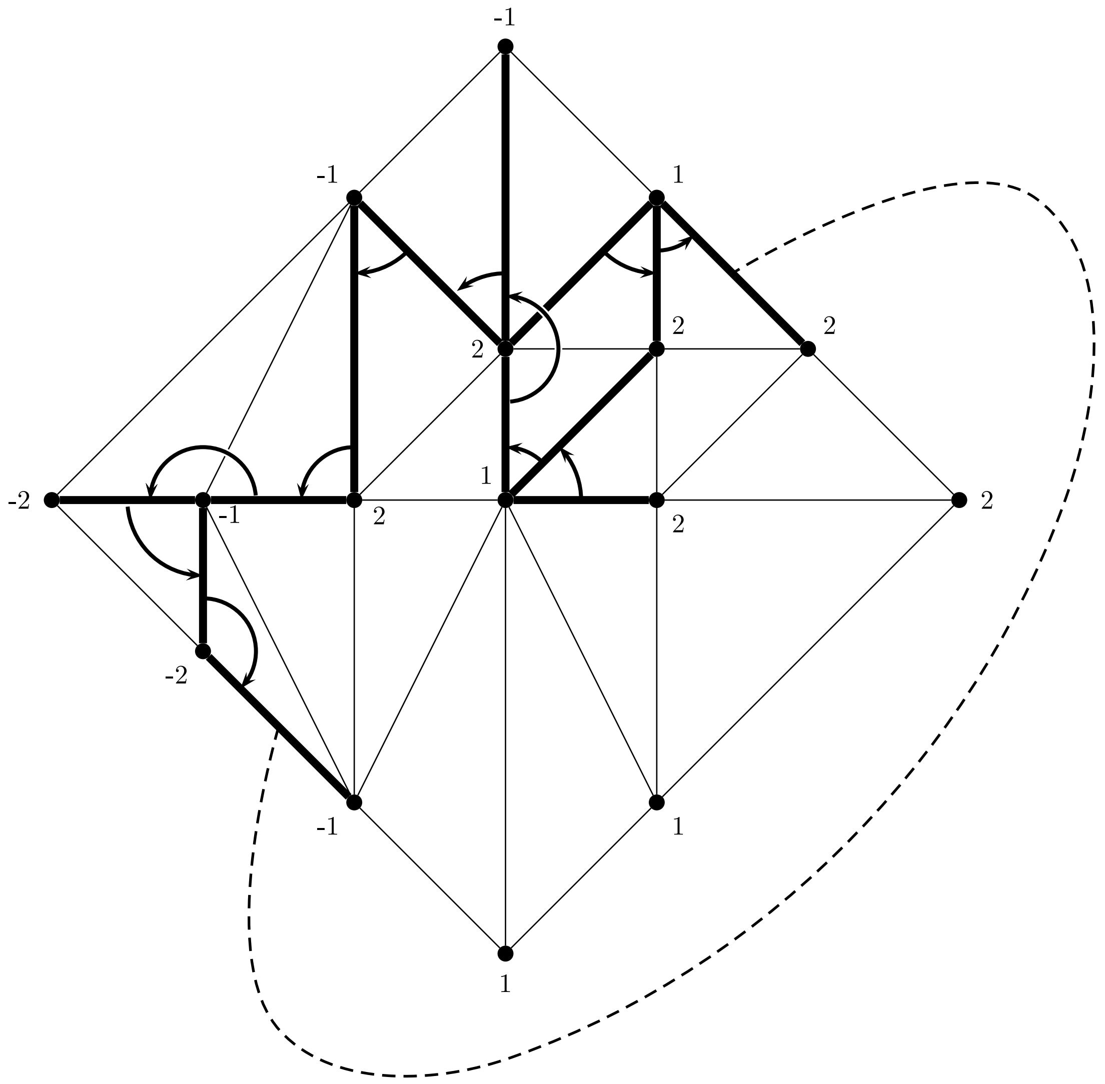


Figure 13: An example of an instance of TUCKER, and the graph G. Happy 1-simplices are indicated with thick lines. Arrows indicate the edges in G that can be directed. The dashed curve indicates an edge in G that is not given a direction; it connects a pair of antipodal happy 1-simplices.

[6] Xi Chen and Xiaotie Deng. Settling the complexity of two-player Nash equilibrium. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 261–272, 2006.

[7] Xi Chen and Xiaotie Deng. On the complexity of 2D discrete fixed point problem. *Theoretical Computer Science*, 410(44):4448–4456, 2009.

[8] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibrium. *Journal of the ACM*, 56(3):Article 14, 2009.

[9] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing (STOC'06)*, pages 71–78, 2006.

[10] Xiaotie Deng, Jack Edmonds, Zhe Feng, Zhengyang Liu, Qi Qi, and Zeying Xu. Understanding PPA-completeness. In *Proc. 31st Conference on Computational Complexity (CCC)*, pages 23:1–23:25, 2016.

[11] Xiaotie Deng, Zhe Feng, and Rucha Kulkarni. Octahedral Tucker is PPA-complete. Technical Report ECCC-TR15-120, Electronic Colloquium on Computational Complexity, July 2017.

[12] Xiaotie Deng, Qi Qi, and Jie Zhang. Direction preserving zero point computing and applications (extended abstract). In *Internet and Network Economics, 5th International Workshop (WINE)*, Lecture Notes in Computer Science 5929. Springer, 2009.

[13] Aris Filo-Ratsikas and Paul W. Goldberg. Consensus-halving is PPA-complete. In *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC'18)*, pages 51–64, 2012.

[14] Aris Filo-Ratsikas and Paul W. Goldberg. The complexity of splitting necklaces and bisecting ham sandwiches. Preprint, arXiv:1805.12559, May 2018.

[15] Robert M. Freund. Variable dimension complexes, part I: Basic theory. *Mathematics of Operations Research*, 9(4):479–497, 1984.

[16] Robert M. Freund. Variable dimension complexes, part II: A unified approach to some combinatorial lemmas in topology. *Mathematics of Operations Research*, 9(4):498–509, 1984.

[17] Robert M. Freund and Michael J. Todd. A constructive proof of Tucker's combinatorial lemma. *Journal of Combinotorial Theory, Series A*, 30:321–325, 1981.

[18] Katalin Friedl, Gábor Ivanyos, Miklos Santha, and Yves F. Verhoeven. Locally 2-dimensional Sperner problems complete for the Polynomial Parity Argument classes. In *Algorithms and Complexity, 6th Italian Conference (CIAC)*, Lecture Notes in Computer Science 3998, pages 380–391. Springer, 2006.

[19] Michelangelo Grigni. A Sperner lemma complete for PPA. *Information Processing Letters*, 77(5-6):255–259, 2001.

[20] Emil Jeřábek. Integer factoring and modular square roots. *Journal of Computer and System Sciences*, 82(2):380–394, 2016.

[21] Jiří Matoušek. A combinatorial proof of Kneser's conjecture. *Combinatorica*, 24(1):163–170, 2004.

[22] Jiří Matoušek. *Using the Borsuk-Ulam Theorem: Lectures on Topological Methods in Combinatorics and Geometry.* Springer, second edition, 2008.

[23] Dömötör Pálvölgyi. 2D-Tucker is PPAD-complete. In *Internet and Network Economics, 5th International Workshop (WINE)*, Lecture Notes in Computer Science 5929, pages 569–574. Springer, 2009.

[24] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

[25] Andrew G. Thomason. Hamiltonian cycles and uniquely edge colorable graphs. *Annals of Discrete Mathematics*, 3:259–268, 1978.

[26] Günter M. Ziegler. Generalized Kneser coloring theorems with combinatorial proofs. *Inventiones Mathematicae*, 147(3):671–691, 2002.