Electronic Colloquium on Computational Complexity, Report No. 118 (2017)

Octahedral Tucker is PPA-Complete

Xiaotie Deng[1], Zhe Feng[2], and Rucha Kulkarni*[3]

[1]Shanghai Jiao Tong University, Shanghai, China
[2]Harvard University, Cambridge, MA, USA
[3]Indian Institute of Technology Bombay, Mumbai, India
[1]deng-xt@cs.sjtu.edu.cn
[2]zhe_feng@g.harvard.edu
[3]ruku937@gmail.com

Abstract

The Octahedral Tucker problem considers an n-dimensional hypergrid of side length two, centered at the origin, triangulated by connecting the origin to all outside vertices (applied recursively on each of the lower dimensional hypergrids passing through their origins at the corresponding reduced dimensions). Each vertex is assigned a color in $\pm i : i = 1, 2, ..., n$ computed using a polynomial size logic circuit which is also a component of the input. Given that antipodal vertices are assigned complementary colors, there is always an edge (equivalently, a 1-simplex of the triangulation) for which the two adjacent vertices are assigned complementary colors. The computational complexity to find one has been an outstanding challenging problem. In this paper, we resolve this problem by proving Octahedral Tucker to be PPA-complete.

*This work was done while visiting Shanghai Jiao Tong University, China

ISSN 1433-8092

1 Introduction

The Complexity class PPA (Polynomial Parity Argument on Graphs) was introduced by Christos Papadimitriou in his seminal paper[13], which is the class of all problems that can be reduced to the problem of finding another odd degree vertex in an undirected graph, where one odd degree vertex has been given. A parity argument affirms the existence of a second odd degree vertex in such a graph. Note that the graph considered has an exponential number of vertices, and the edges incident to a vertex can be computed with a polynomial time size circuit using the index of the vertex as the input. In comparison, the PPAD class is the analog of PPA on directed graphs where both the in-degree and the out-degree is no more than one for all vertices.

The study of these classes has motivated diverse interests, leading to increased modeling efforts for solving computational issues involved with interesting combinatorial content that go into the heart of combinatorial principles in a wide range of areas[2]. Both classes have within them a significant number of member problems, such as a group of equilibrium computation problems listed in [8] for PPAD, the integer factoring problem [10]. When it comes to complete problems, although there have been a significant number of problems known to be PPAD-complete, such as an array of problems listed to be PPAD-complete [11], few completeness results have been proven for PPA. Despite close resemblance in their definitions, there is an extreme disparity in our understanding of their complete problems until the last couple of years.

In defining both PPA and PPAD as well as some related complexity classes [13], Papadimitrou introduced several problems belonging to PPA. Later, the first PPA-complete result was proven by Grigni in [9], proving the non-orientable 3-D version of the SPERNER problem (finding a fully colored simplex in a properly colored and trian- gulated polygon) to be PPA-Complete. Friedl et al in [7] further strengthened this result, proving the PPA-completeness of a non-orientable locally 2-D version of SPERNER. After a decade, with no new results in between, Deng et al in [4] characterized the Möbius band as the source for the PPA-complete properties in discrete fixed points by a unified proof for several such problems they then proved PPA-complete. These included a version of TUCKER (finding an edge colored by two colors $\{-c, c\}$ in a triangulated polygon with a proper coloring anti-symmetric on the boundary) and SPERNER based on the mobius band, along with other versions of discrete fixed point problems. Subsequently, Aisenberg et al proved the first problem based in Euclidean space, the General 2-D Tucker, to be PPA-complete, in [1].

Under the context of combinatorial topology, Fan [5] originated the study of Octahedral Tucker in 1952, which is a high dimensional version of the TUCKER problem on a hypergrid with side-length of two. Despite of its ressemblance and strong ties to fixed point problems, their differences are not fully understood in the past 65 years. Since Aisenberg, et al., raised the challenge of computational complexity for the Octahedral Tucker problem in [1], the key question becomes computational: whether any other PPA-complete problem can be reduced in polynomial time to the Octahedral Tucker problem. The hardness proof is difficult especially because of its simple structure. For related high dimensional fixed point problems, the SPERNER problem can be shown PPAD-complete in the case where it is restricted to a constant side length high dimension case [3]. For the case of high dimensional non-orientable space, SPERNER can be shown to be PPA-complete for a slightly larger constant side length [4]. Yet the Octahedral Tucker challenge is almost impossible at a first look: its side length is the minimum possible two. A far more precise design in construction would be necessary to embed the proven structure of the generic PPA-complete problem for undirected graphs of cycles and paths into the smallest side length hypergrid, if possible at all. If done, its simplicity would open up new frontiers in the above discussed wide areas.

In this paper, we prove Octahedral Tucker PPA-complete. Our approach starts at a new version of the 2-D TUCKER (2-D General Octahedral TUCKER) problem. It pads together an exponential number of 2-D Octahedral Tucker instances to make a generalized 2-D TUCKER instance, which is proven PPA-hard. We reduce this new 2-D TUCKER instance to Octahedral Tucker in steps called wrappings, where we reduce the side lengths of the hypergrid, and simultaneously increase the number of dimensions. First, we need to make sure that, if the input 2-D Tucker instance has parameter n, i.e. can be specifed using n bits, the final dimension is bounded by a polynomial in n. Note that the initial 2-D TUCKER instance can have a side length exponential in n(Such an input can be easily described in several ways, for example, given a node presented by two coordinates of n bits each in the exponential size 2-D grid, we could have a polynomial size logic circuit computing the color assigned to the node). The wrapping process needs to reduce exponential side lengths to constant(2) side lengths in a polynomial number of steps. Thus, when done iteratively, each iteration must wrap a larger length as tightly as possible on higher dimensional smaller lengths(without wasting lengths as that will lead to increased number of iterations). This property must be maintained until the side length reduces to two, the smallest possible for the problem to be nontrivial.

In the first stage of the wrapping process, we reduce each side length of the hypergrid to multiple side lengths all of size 8. Here in each iteration we ensure that while the number of dimensions increases by one, the sum of side lengths decreases geometrically. Eventually, we end up with a polynomial number of dimensions with each side length exactly 8. Therefore, the sum of side lengths will become a polynomial in the input parameter n.

The second stage of wrapping process reduces a General Octahedral TUCKER instance of constant length 8 sides to an Octahedral Tucker instance, and has a much trickier requirement in maintaining TUCKER's structure than in the first stage of wrapping. However, as we already have a total side length polynomial in the input parameter n, we can afford to be wasteful in wrapping. Our main idea is to reduce the side length of each dimension to a quarter of the original at the beginning of this stage, while increasing the total number of dimensions by a factor of 4.

The detailed reductions are tricky and require careful design of gadgets to ensure antipodal symmetry on the boundary so that eventually the two stage process allows us to reduce each dimension side length to 2 and the number of dimensions to a polynomial in n, and therefore complete our wrapping process successfully.

In the other direction for proving higher dimension TUCKER is in PPA, Papadimitriou [13] proposed a key idea of dimension raising [6] which is also employed in Aisenberg, et al., for 2-D TUCKER [1]. The nice structure of Octahedral Tucker allows us to develop a proof which focuses on the use of this technique, included in appendix A for completeness.

To overcome the almost impossible task of a precise result, a series of technical tools were developed along the way toward the proof of Octahedral Tucker's PPA-completeness.

First, the proper problem to reduce to Octahedral Tucker had to be chosen, or defined and proved PPA-Complete. The main concern here was developing one on which the wrapping stages could be applied iteratively while maintaining a sustainable consequence of wraps. We thus defined the General Octahedral Tucker problem, and fixed it's 2-D case as our starting point. This problem, by itself, could be useful in other fixed point theory and complexity theory problems.

Even if all the first stage wrapping processes go through, there is still the inevitable task of reducing side lengths down to two, a task next only to impossible. All vertices except the origin are boundary nodes, and hence each has an antipodal partner. Coloring these vertices while following tucker lemma constraints, without adding extra solutions in the wrapping processes is challenging when the manipulation room is only an arm length of two.

Additionally, according to Octahedral triangulation, the origin is connected to every vertex in the hypergrid. Thus, for any color assigned to the origin, if some other vertex is also assigned the same color, then the antipodal partner of this other node has the opposite color assigned to it, and forms a complementary edge with the origin. Hence, with an exponential(3^n) number of vertices in the grid to be colored using polynomial($2n$) colors, if the origin cannot be adjacent to a complementary edge, it has to assigned a color different from that assigned to every vertex in the hypergrid(Allowing the origin to be adjacent to a complementary edge, and maintaining this edge in successive stages is even more difficult).

With every increased dimension, we are provided exactly one pair of complementary colors. Obvious ways to ensure retaining the original solution set would involve coloring all new vertices with the new colors(and their antipodal vertices with the opposite color). Thus, for a hypergrid in an n-dimensional space, we would require n colors simply to avoid extra solutions. But this leaves the origin to be colored using one of the same n colors, thus creating unwanted solutions. The idea of reusing one color in two new dimensions, without coloring their boundary with a new color, is key to the reduction, and we believe, could prove a useful tool to resolve outstanding graph theoretical, coloring, or fixed point problems in complexity theory.

The other proof proving Octahedral Tucker in PPA follows the key idea in [13], but is itself a precise execution. The technique exhibits a set of elegant choices of structures that move toward the final prettiness of a piece of theoretical work.

Presentation Structure: In Section 2, we introduce the class PPA, and the standard PPA-complete problem AEUL (another end of undirected lines) [4]. We then define the General TUCKER and Octahedral Tucker problems. In Section 3 we focus on the main task to prove Octahedral TUCKER PPA-hard. We conclude with remarks and discussion on the future potential of this work.

2 Preliminaries

2.1 The PPA Class

A complexity class is often defined to have a collection of complete problems to which other problems can be reduced within time and space appropriate (with lower complexity) for the class. The famous complexity classes NP includes all problems which could be solved in polynomial time on non-deterministic Turing machines. A NP-complete problem is one to which every problem in NP can be reduced to in deterministic polynomial time. Other new complexity classes are also identified by their computational machinery, and the corresponding complete problems as its members to which all other member problems are reduced to within relatively lower computational computational resources. PPA(Polynomial Parity Argument on graphs) is one such complexity class, introduced by Cristos Papadimitriou in 1994, in his seminal paper [13], the member problems of which always have a solution guaranteed by a proof structure of a graph. A fixed odd degree node is given in the graph, and any other odd degree node will be a solution to the problem.

For simplicity, we make use of a computationally equivalent version of PPA based on an exponential size graph consisting of nodes of maximum degree two, with a given node of degree one. We name it AEUL (Another End of Undirected Lines) with the follow specifications:

Definition 2.1 (AEUL [4])**.** *Suppose that an input circuit L_n of polynomial size in n takes an input u in the configuration space $C_n = \{0,1\}^n$ and returns an output $L_n(u)$ in the form $\langle v, w\rangle, \langle v\rangle,$ or $\langle\rangle$ where $v > w$ and $v, w \in C_n \setminus \{u\}$. $\mathbf{0}^n$ is a given configuration of one tuple, i.e., $|L_n(\mathbf{0}^n)| = 1$. The search problem is to find another configuration v, $v \neq \mathbf{0}^n$ such that $|L_n(v)| = 1$.*

2.2 Octahedral Tucker is in PPA

The Tucker problem and it's various versions, like the General Tucker, Octahedral Tucker and General Octahedral Tucker, arise from the Tucker's lemma, a classical statement in topology. In this section, we state the lemma, and provide all terms and definitions building up to the definition of Octahedral Tucker, and end with a theorem proving the membership of Octahedral Tucker in PPA.

Lemma 2.2. *(Tucker's Lemma): In every triangulation of an n-dimensional simplex, whose vertices are assigned colors from $\{\pm 1, \pm 2 ... \pm n\}$ by a coloring function that maintains antipodal symmetry, a complementary edge always exists*

Tucker's lemma directly leads to General Tucker, the computational problem of finding a complementary edge in an antipodally symmetric colored triangulated n-D simplex. For simplicity, we define n-D Tucker on a *"Hypergrid"* called $V_n = \{\mathbf{p} = (p_1, p_2, ..., p_n) \in \mathbb{Z}_n, \forall i, -\frac{N_i}{2} \leq p_i \leq \frac{N_i}{2}\}$,[1] where N_i is the length of ith dimension. $B_n = \{\mathbf{p}, \exists p_i = \{-N_i/2, N_i/2\}\}$ is the boundary of this hypergrid. Let $K_{\mathbf{p}} = \{\mathbf{q} : q_i \in \{p_i, p_i + 1\}\}$ be the unit hypergrid in V_n.

Definition 2.3 (n-D Tucker)**.** *The input of n-D Tucker is a pair (G, V_n) where V_n is an n dimensional triangulated hypergrid centered at $\boldsymbol{0}$ and G is a polynomial-time machine, which generates a coloring function $g : V_n \to \{\pm 1, \pm 2, ..., \pm n\}$ satisfying the antipodal boundary condition: $\forall \boldsymbol{p} \in B_n$, $g(-\boldsymbol{p}) = -g(\boldsymbol{p})$. The output of n-D Tucker is a complementary 1-simplex, i.e an edge $(\boldsymbol{p}, \boldsymbol{q})$ s.t. $g(\boldsymbol{p}) = -g(\boldsymbol{q})$.*

With the help of this definition of n-D Tucker, we can give the most important definition in this paper — n-D Octahedral Tucker:

[1]For simplicity, we assume N_i is even for each i.

Definition 2.4 (n-D Octahedral Tucker)**.** *n-D Octahedral Tucker ($n \geq 2$) is the special case of n-D Tucker (G, V_n) which satisfies*

- *The side length of each dimension is exactly 2, i.e. $\forall \boldsymbol{p} \in V_n, \forall i \in [n], p_i \in \{-1, 0, 1\}$.*
- *Standard Octahedral Tucker triangulation (**SOTT**), which is defined recursively in the following:*
  - *The SOTT in 2-D Octahedral Tucker is shown in Figure 1.*
  - *Any 2-D 2×2 facet (contains 9 vertices) in n-D Octahedral is triangulated by **SOTT** shown in Figure 1. For example, there are 15 2-D 2×2 facets in 3-D Octahedral Tucker which, shown in Figure 2.*

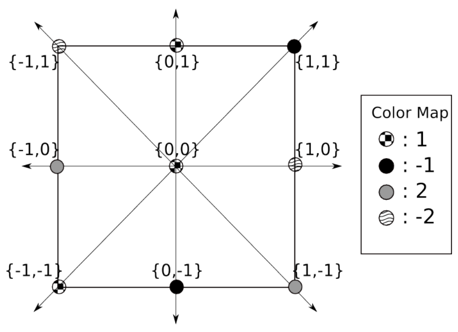

Figure 1: An instance of the 2-D Octahedral Tucker Problem

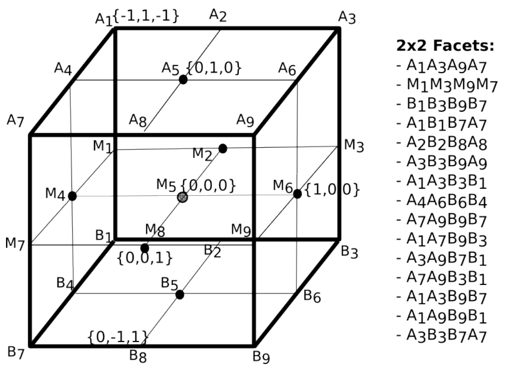

Figure 2: An instance of the 3-D Octahedral Tucker Problem. The 15 2×2 facets in the 3-D hypergrid, each triangulated by 2-D Octahedral Triangulation, are enumerated in the column to the right

Formally, we could give an equivalent definition for Standard Octahedral Tucker Triangulation in the following:

Definition 2.5 (Standard Octahedral Tucker Triangulation (**SOTT**))**.** *In a n-D Octahedral Tucker (G, V_n), we define a preference relation $\succsim$ s.t. $1 \succsim 0$, $-1 \succsim 0$, $1 \succsim 1$, $-1 \succsim -1$ and $0 \succsim 0$. However, there is no relation between -1 and 1. Further, we say $\boldsymbol{p} \succsim \boldsymbol{q}, \forall \boldsymbol{p}, \boldsymbol{q} \in V_n$*

iff $\forall i \in [n], p_i \gtrsim q_i$. *The Standard Octahedral Tucker Triangulation of* (G, V_n) *is defined as:* $\forall \boldsymbol{p}, \boldsymbol{q} \in V_n$, $\boldsymbol{p}$ *and* $\boldsymbol{q}$ *are linked iff* $\boldsymbol{p} \gtrsim \boldsymbol{q}$ *or* $\boldsymbol{q} \gtrsim \boldsymbol{p}$.

With the definition of n-D Octahedral Tucker, we know Octahedral Tucker is a special case of General Tucker. As General Tucker is in PPA[1, 13], Theorem 2.6 follows. For completeness, we present a simple and complete proof in Appendix A for showing n-D Tucker is in PPA.

Theorem 2.6 ([1, 13]). *Octahedral Tucker is in PPA*

2.3 2-D General Octahedral Tucker is PPA-hard

In addition, to prove the PPA-hardness of Octahedral Tucker, we need to define another special case of n-D Tucker *General Octahedral Tucker* as well as the corresponding triangulation, which is called *General Octahedral Tucker triangulation (**GOTT**)*.

Definition 2.7 (General Octahedral Tucker). *A General Octahedral Tucker is an* n*-D Tucker* (G, V_n) *with length of each dimension* $\{N_1, N_2, ..., N_n\}$ *where* $\forall i \in [n], N_i = 4k_i, k_i \geq 2$ *under the general Octahedral Tucker triangulation (**GOTT**), which is defined as: any octahedral hypergrid centered at vertices with odd numbers in both coordinate*

$$H_{\boldsymbol{p}} = \{\boldsymbol{q} : q_i \in \{p_i - 1, p_i, p_i + 1\} | p_i = 2m_i + 1, -k_i \leq m_i \leq k_i - 1\}$$

is triangulated by Standard Octahedral Tucker Triangulation.

To proceed further to prove Octahedral Tucker is PPA-hard, as a starting point of the reduction process, we first prove the PPA-hardness of 2*-D General Octahedral Tucker.*

Theorem 2.8. 2*-D* General Octahedral Tucker *is PPA-hard.*

Proof. The proof of this theorem follows directly from Aisenberg et al's proof[1] claiming 2-D Tucker is PPA-complete, with a number of modifications. Aisenberg et al's proof [1] reduces the AEUL problem to 2-D Tucker by encoding the entire AEUL graph on a 2-D Tucker instance of suitable size. The AEUL vertices are each mapped to 13×13 grids, with one entrance and one exit possible to each grid(denoted as the 'outgoing' and 'incoming' edge of the vertex). Edges of the AEUL graph are encoded by joining one of these 'edges' of each Tucker vertex to each other via 3-wide tubes, that have (-1)-colored vertices at the center, $(+2)$-colored edges on one side and (-2)-colored vertices on the other side of the tube. The rest of the Tucker vertices not mapped to any AEUL vertex/edge (called the 'environment') are colored $+1$. Thus, one end of every single degree vertex will have an open tube, with the center (-1)-colored vertex exposed to the environment, forming a complementary edge. That is, every Tucker solution corresponds to a solution of the AEUL problem that can be found, given the Tucker solution, in polynomial time, thus completing the reduction.

2-D Octahedral Tucker uses octahedral triangulation, and has size $4p \times 4q$ **cells**, for $p, q \in \mathbb{N}$. As Aisenberg et al's paper reduces AEUL to a Tucker instance of size $m \times m$ **lines**[2] where $m = 4 \times 13 \times |G|$, we modify their proof as follows:

[2]Lengths are measured in number of lines in [1], whereas in number of cells in our paper. Note that the number of lines = 1 + number of cells

1. Use a 4-wide tube with the inner two lines colored -1, instead of the original 3-wide tubes with only one center line
2. Map each AEUL vertex to a grid of size 20×20 cells (or 21×21 lines), instead of the original grid size of 13×13 lines

That is, we follow the entire reduction procedure by mapping an AEUL instance to a 2-D Octahedral instance of size $m \times m$ where $m = 4 \times 21 \times |G|$ lines. We map each AEUL vertex to a grid of size 20×20 cells, and edges to 4-wide tubes in the Tucker grid.

Our Tucker instance has General Octahedral triangulation. For an open end in a grid corresponding to a single degree AEUL vertex, having a 4-wide tube ensures at least one of the inner (-1)-colored vertices has an edge incident on an environment vertex. A 4-wide tube also ensures symmetric Tucker solutions generated for each AEUL solution, making analysis easy.

Each grid assigned to a AEUL vertex has a tube with a 'jump' in the center to ensure antipodal symmetry. As the width of the tube is now increased by 1, we need to add one more line in the base grid for the extra inner wire. When two tubes cross, the crossings are resolved by 'bending' the tubes slightly. The entire crossing is located in one grid. We need to add 4 cells each at the top and bottom of the center tube to ensure all crossings in the 2-D Octahedral Tucker instance too get resolved in one grid. This increases the grid side length to $8 + 4 + 8 = 20$ cells, or 21 lines.

This modified reduction scheme reduces AEUL to 2-D Octahedral Tucker, thus proving 2-D Octahedral Tucker PPA-Hard. □

3 Octahedral Tucker is PPA-hard

We prove that Octahedral Tucker is PPA-hard by reducing 2-D General Octahedral Tucker to Octahedral Tucker in polynomial time. The reduction algorithm folds each dimension of the 2-D Tucker instance into multiple dimensions of smaller lengths iteratively, until we reach a hypergrid whose length in all dimensions is two. Every step of the algorithm is a polynomial time reduction from one general Octahedral Tucker instance to another in a higher dimension. The final hypergrid, by definition is an instance of Octahedral Tucker, thus completing the reduction from 2-D General Tucker to Octahedral Tucker.

The algorithm is discussed in detail in the subsequent subsections.

3.1 Wrapping Stage 1: Reducing side lengths to 8

We divide this subsection into two parts: First, we introduce the Stairwise lemma: fold $(n-1)$D-Tucker to nD-Tucker and shrink the length of one dimension simultaneously. Second, we show how to use this lemma recursively to reduce 2D-Tucker to a higher dimension Tucker instance with length in each dimension 8.

Before we describe the Stairwise Lemma which is the main lemma in this subsection, we give the following lemma to restrict our discussion to a specific class of 2-D *General Octahedral Tucker*.

Lemma 3.1. *Any* $2D$ General Octahedral Tucker *instance* (G, V_2) *can be reduced to a* $2D$ General Octahedral Tucker *instance* (G', V_2') *whose lengthes in the two dimensions are* 2^m *and* 2^n*, for some* $m, n \in \mathbb{Z}$.

Proof. Denote the lengthes of two dimensions of (G, V_2) by N_1, N_2, where N_1 and N_2 are both multiple of 4. Let $m, n \in \mathbb{Z}^+$ s.t. $2^{m-1} < N_1 \leq 2^m$ and $2^{n-1} < N_2 \leq 2^n$. We list the construction below, illustrated in Figure 3.

I $\forall \mathbf{q} \in V_2'$ with $-N_1/2 \leq q_1 \leq N_1/2$ and $-N_2/2 \leq q_2 \leq N_2/2$, then $g'(\mathbf{q}) = g(\mathbf{p})$ where $p_1 = q_1$ and $p_2 = q_2$.

II $\forall \mathbf{q} \in V_2'$ with $-N_1/2 \leq q_1 \leq N_1/2$ and $N_2/2 < q_2 \leq 2^{n-1}$, then $g'(\mathbf{q}) = g(\mathbf{p})$ where $p_1 = q_1, p_2 = N_2/2$.

III $\forall \mathbf{q} \in V_2'$ with $N_1/2 < q_1 \leq 2^{m-1}$ and $-N_2/2 \leq q_2 \leq N_2/2$, then $g'(\mathbf{q}) = g(\mathbf{p})$ where $p_1 = N_1/2, p_2 = q_2$.

IV $\forall \mathbf{q} \in V_2'$ with $-N_1/2 \leq q_1 \leq N_1/2$ and $-2^{n-1} \leq q_2 < -N_2/2$, then $g'(\mathbf{q}) = g(\mathbf{p})$ where $p_1 = q_1, p_2 = -N_2/2$.

V $\forall \mathbf{q} \in V_2'$ with $-2^{m-1} \leq q_1 < -N_1/2$ and $-N_2/2 \leq q_2 \leq N_2/2$, then $g'(\mathbf{q}) = g(\mathbf{p})$ where $p_1 = -N_1/2, p_2 = q_2$.

VI For $\mathbf{q} \in V_2'$ with $-2^{m-1} \leq q_1 < N_1/2$ and $N_2/2 < q_2 \leq 2^{n-1}$, then $g'(\mathbf{q}) = g(\mathbf{p})$ where $p_1 = -N_1/2$ and $p_2 = N_2/2$.

VII For $\mathbf{q} \in V_2'$ with $N_1/2 < q_1 \leq 2^{m-1}$ and $N_2/2 < q_2 \leq 2^{n-1}$, then $g'(\mathbf{q}) = g(\mathbf{p})$ where $p_1 = N_1/2$ and $p_2 = N_2/2$.

VIII For $\mathbf{q} \in V_2'$ with $N_1/2 < q_1 \leq 2^{m-1}$ and $-2^{n-1} \leq q_2 < -N_2/2$, then $g'(\mathbf{q}) = g(\mathbf{p})$ where $p_1 = N_1/2$ and $p_2 = -N_2/2$.

IX For $\mathbf{q} \in V_2'$ with $-2^{m-1} \leq q_1 < N_1/2$ and $-2^{n-1} \leq q_2 < N_2/2$, then $g'(\mathbf{q}) = g(\mathbf{p})$ where $p_1 = -N_1/2$ and $p_2 = -N_2/2$.

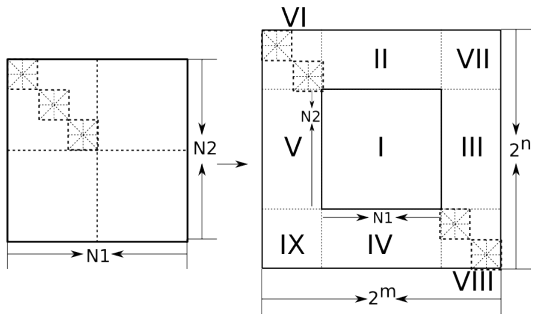

Figure 3: Converting a general 2-D Tucker instance into one whose sides have length $\{2^m, 2^n\}$, where $m, n \in \mathbb{N}$.

For the correctness of the reduction, first, there is a complementary edge existing in part I if and only if it is a complementary edge in original Tucker (G, V_2). Next, there is no complementary edge in parts VI, VII, VIII, IX. Third, if there is a complementary edge in part II, III, IV or V, it means there is a corresponding complementary edge located in the original Tucker (G, V_2) on the boundary and we can find it in polynomial time. We summarize in Table 1, the corresponding complementary edge $\{\mathbf{p}, \mathbf{q}\}$ in (G, V_2), for each complementary edge $\{\mathbf{p}', \mathbf{q}'\}$ in (G', V'_2) where $p'_1 \le q'_1$.

Given Complementary edge in (G', V'_2)		Corresponding Complementary edge in (G, V_2)	
$\mathbf{p}'$	$\mathbf{q}'$	$\mathbf{p}$	$\mathbf{q}$
$-\frac{N_1}{2} \le p'_1 \le \frac{N_1}{2}$, $-\frac{N_2}{2} \le p'_2 \le \frac{N_2}{2}$	$-\frac{N_1}{2} \le q'_1 \le \frac{N_1}{2}$, $-\frac{N_2}{2} \le q'_2 \le \frac{N_2}{2}$	$p_1 = p'_1$ $p_2 = p'_2$	$q_1 = q'_1$ $q_2 = q'_2$
$-\frac{N_1}{2} \le p'_1 \le \frac{N_1}{2}$, $\frac{N_2}{2} < p'_2 \le 2^{n-1}$	$-\frac{N_1}{2} \le q'_1 \le \frac{N_1}{2}$ $\frac{N_2}{2} < q'_2 \le 2^{n-1}$	$p_1 = p'_1$ $q_1 = \frac{N_1}{2}$	$q_1 = q'_1 = p'_1 + 1$ $q_2 = \frac{N_1}{2}$
$\frac{N_1}{2} < p'_1 \le 2^{m-1}$ $-\frac{N_2}{2} < p'_2 < \frac{N_2}{2}$	$\frac{N_1}{2} < q'_1 \le 2^{m-1}$ $-\frac{N_2}{2} < q'_2 < \frac{N_2}{2}$	$p_1 = \frac{N_1}{2}$ $p_2 = p'_2$	$q_1 = \frac{N_1}{2}$ $q_2 = q'_2$
$-\frac{N_1}{2} \le p'_1 \le \frac{N_1}{2}$, $-2^{n-1} \le p'_2 < -\frac{N_2}{2}$	$-\frac{N_1}{2} \le q'_1 \le \frac{N_1}{2}$ $-2^{n-1} \le q'_2 < -\frac{N_2}{2}$	$p_1 = p'_1$ $p_2 = -\frac{N_2}{2}$	$q_1 = q'_1 = p'_1 + 1$ $q_2 = -\frac{N_2}{2}$
$-2^{m-1} \le p'_1 < -\frac{N_1}{2}$ $-\frac{N_2}{2} < p'_2 < \frac{N_2}{2}$	$-2^{m-1} \le q'_1 < -\frac{N_1}{2}$ $-\frac{N_2}{2} < q'_2 < \frac{N_2}{2}$	$p_1 = -\frac{N_1}{2}$ $p_2 = p'_2$	$q_1 = -\frac{N_1}{2}$ $q_2 = q'_2$

Table 1: Finding the corresponding complementary edge $\{\mathbf{p}, \mathbf{q}\}$ in (G, V_2) given a complementary edge $\{\mathbf{p}', \mathbf{q}'\}$ in (G', V'_2).

As we can find a complementary edge in the original $2D-$Tucker (G, V_2) instance in polynomial time, given a complementary edge in (G', V'_2), the proof describes a valid reduction as asserted by the theorem. $\square$

Lemma 3.2 (Stairwise Lemma). *Suppose we have an $(n-1)$-D General Octahedral Tucker instance (G, V_{n-1}) with length of each dimension $\{N_1, N_2, ..., N_{n-1}\}$, where $N_{n-1} = 4k(k > 2)$, we can reduce it to n-D General Octahedral Tucker (G', V_n) with length of each dimension $\{N_1, N_2, ..., N'_{n-1}, N'_n\}$ where $N'_{n-1} = 8, N'_n = \frac{N_{n-1}}{2} = 2k$ under general Octahedral triangulation.*

Proof. The proof structure is divided to the following two stages.

1. we first show how to fold $(n-1)$D-Tucker to nD-Tucker,

2. then we prove the correctness of the reduction.

First part of the proof. The idea is to shrink $(n-1)$th dimension and append another new dimension (nth dimension) simultaneously. First, we show how to embed the $(n-1)$D-Tucker into V_n. We describe this process mathematically as follows:

Embedding

- $\forall \mathbf{p} \in V_{n-1}$ with $-2k \le p_{n-1} \le -2k+1$, embed it to $\mathbf{p}' = (p_1, ..., p_{n-2}, p_{n-1}+2k-4, 0)$ in V_n, i.e. $g'(\mathbf{p}') = g(\mathbf{p})$.
- $\forall \mathbf{p} \in V_{n-1}$ with $-2k+2 \le p_{n-1} \le -k$, embed it to $\mathbf{p}' = (p_1, ..., p_{n-2}, -2, p_{n-1}+2k-2)$ in V_n, i.e. $g'(\mathbf{p}') = g(\mathbf{p})$.
- $\forall \mathbf{p} \in V_{n-1}$ with $p_{n-1} = -k+1$, embed it to $\mathbf{p}' = (p_1, ..., p_{n-2}, -1, k-2)$ in V_n, i.e. $g'(\mathbf{p}') = g(\mathbf{p})$.
- $\forall \mathbf{p} \in V_{n-1}$ with $-k+2 \le p_{n-1} \le k-2$, embed it to $\mathbf{p}' = (p_1, ..., p_{n-2}, 0, -p_{n-1})$ in V_n, i.e. $g'(\mathbf{p}') = g(\mathbf{p})$.
- $\forall \mathbf{p} \in V_{n-1}$ with $p_{n-1} = k-1$, embed it to $\mathbf{p}' = (p_1, ..., p_{n-2}, 1, -k+2)$ in V_n, i.e. $g'(\mathbf{p}') = g(\mathbf{p})$.
- $\forall \mathbf{p} \in V_{n-1}$ with $k \le p_{n-1} \le 2k-2$, embed it to $\mathbf{p}' = (p_1, ..., p_{n-2}, 2, p_{n-1}-2k+2)$ in V_n, i.e. $g'(\mathbf{p}') = g(\mathbf{p})$.
- $\forall \mathbf{p} \in V_{n-1}$ with $2k-1 \le p_{n-1} \le 2k$, embed it to $\mathbf{p}' = (p_1, ..., p_{n-2}, p_{n-1}-2k+4, 0)$ in V_n, i.e. $g'(\mathbf{p}') = g(\mathbf{p})$.

Then, we show how to add new color to satisfy antipodal boundary in V_n.

Adding Colors Based on the embedding strategy above, $\forall$ $\mathbf{p}$ in V_{n-1}, there is a corresponding $\mathbf{p}'$ in V_n. We define the set of these corresponding vertices in V_n as S. Next we stipulate how to color vertices in $V_n \backslash S$. Based on the above discussion, we find that we divide V_n into two parts by S. One part is above S where we color all vertices with n, while the other part is below S where we color them with $-n$.

Formally, we define adding colors as follows:

- For any $\mathbf{q} \in V_n \backslash S$ which is above S,
  - $\forall \mathbf{q} \in V_n$ with $-4 \le q_{n-1} \le -3$ and $q_n > 0$, $g'(\mathbf{q}) = n$.
  - $\forall \mathbf{q} \in V_n$ with $-2 \le q_{n-1} \le 0$ and $q_n > k-2$, $g'(\mathbf{q}) = n$.
  - $\forall \mathbf{q} \in V_n$ with $q_{n-1} = 1$ and $q_n > -k+2$, $g'(\mathbf{q}) = n$.
  - $\forall \mathbf{q} \in V_n$ with $2 \le q_{n-1} \le 4$, $g'(\mathbf{q}) = n$.
- For any $\mathbf{q} \in V_n \backslash S$ which is below S,
  - $\forall \mathbf{q} \in V_n$ with $-4 \le q_{n-1} \le -2$ and $q_n < 0$, $g'(\mathbf{q}) = -n$.
  - $\forall \mathbf{q} \in V_n$ with $q_{n-1} = -1$ and $q_n < k-2$, $g'(\mathbf{q}) = -n$.
  - $\forall \mathbf{q} \in V_n$ with $0 \le q_{n-1} \le 2$ and $q_n < -k+2$, $g'(\mathbf{q}) = -n$.
  - $\forall \mathbf{q} \in V_n$ with $3 \le q_{n-1} \le 4$, $g'(\mathbf{q}) = -n$.

To illustrate, we show embedding and adding colors for the $n = 3$ case in Figure 4.

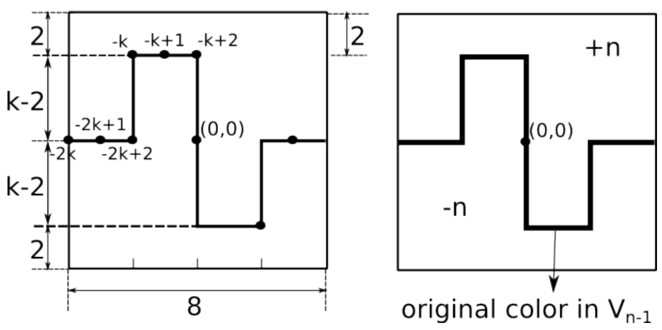

Figure 4: Reduce $2D$-Tucker with length of each dimension $\{N_1, N_2\}$ to $3D-$Tucker. (a): Embedding $2D$-Tucker with $N_2 = 4k$ $(k > 2)$ in $3D-$Tucker with $N_1' = N_1$, $N_2' = 8$ and $N_3' = 2k$. We just plot the boundary face of V_3 with $\mathbf{p}_1 = -N_1/2$. (b): Adding colors in V_3

Second part of the proof. We now show the correctness of this reduction. First, we can easily check that (G', V_n) satisfies the antipodal boundary condition, i.e, $\forall \mathbf{q} \in V_n$, $g'(-\mathbf{q}) = -g'(\mathbf{q})$. Second, we show there is no other complementary edge which is not in original $(n-1)$D-Tucker (G, V_{n-1}). This is because under **Octahedral** triangulation: (a) the edge between S and $V_n \backslash S$ is not a complementary edge, (b) we can't link two vertices colored by n and $-n$, (c) we don't link any new edge in original $(n-1)$D-Tucker.

- (a) is obviously correct since there are no $\pm n$ in S, then we turn to see (b) and (c).
- For (b), $\forall \mathbf{p}, \mathbf{q} \in V_n$ with $g'(\mathbf{p}) = n, g'(\mathbf{q}) = -n$, $|p_{n-1} - q_{n-1}| \geq 2$ or $|p_n - q_n| \geq 2$ according to our construction. Therefore, $\mathbf{p}, \mathbf{q}$ cannot both belong to a same unit hypergrid. Thus, we can't link $\mathbf{p}, \mathbf{q}$ together using general **Octahedral** triangulation.
- For (c), we show general **Octahedral** triangulation will guarantee this claim. We only need to prove the general Octahedral triangulation in V_n maintains the original general Octahedral triangulation in V_{n-1} for S. For any octahedral hypergrid (Definition 2.7) $H_\mathbf{p}$ in V_{n-1}, $H_\mathbf{p}$ is embedded in S which is denoted by $H'_\mathbf{p}$. Based on our construction, any $H'_\mathbf{p}$ is the boundary surface of a octahedral hypergrid in V_n. Following the definition of general Octahedral triangulation, $H'_\mathbf{p}$ is triangulated by standard Octahedral triangulation. Therefore, the triangulation of V_n for S is the original general Octahedral triangulation for V_{n-1}.

Claims (a), (b) and (c) show that there is no other new complementary edge added by our construction. Combined with the satisfication of antipodal boundary condition, we have thus proved the correctness of this reduction. □

Given Lemma 3.1 and Lemma 3.2, we have the following Theorem of PPA-completeness for constant side length Tucker.

Theorem 3.3. *For any $2D$ General Octahedral Tucker instance (G, V_2) where the length of two dimensions are N_1 and N_2 respectively, it can be reduced to a $O(\log N_1 + \log N_2)D$ General Octahedral Tucker with length in each dimension 8.*

$< \mathbf{k} >$	$< \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} >$
$< -4 >$	$< -1, -1, -1, -1 >$
$< -3 >$	$< -1, -1, -1, 0 >$ *and* $< -1, 0, -1, -1 >$
$< -2 >$	$< -1, -1, -1, 1 >$ *and* $< -1, 1, -1, -1 >$
$< -1 >$	$< -1, -1, 0, 1 >$ *and* $< -1, 1, 0, -1 >$
$< 0 >$	$< -1, -1, 1, 1 >$ *and* $< -1, 1, 1, -1 >$
$< 1 >$	$< -1, 0, 1, 1 >$ *and* $< -1, 1, 1, 0 >$
$< 2 >$	$< -1, 1, 1, 1 >$
$< 3 >$	$< 0, 1, 1, 1 >$

Table 2: Mapping coordinates of the input instance in the output instance. $< \mathbf{k} >$ denotes the vertex that has coordinate k in the dimension shrunk in the current iteration. $< \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} >$ denotes the corresponding coordinates of the differentiating dimensions in the output instance

Proof. Based on lemma 3.1, w.l.o.g. we assume length of two dimensions of 2*D General Octahedral Tucker* instance (G, V_2) are 2^m and 2^n. Using Stairwise Lemma recursively, we can reduce the 2*D General Octahedral Tucker* (G, V_2) to an $O(m+n)$D *General Octahedral Tucker* with length in each dimension 8 in polynomial time[3]. □

3.2 Wrapping Stage 2: Reducing Side Lengths from 8 to 2

To summarize, we have until now reduced a 2-D General Tucker instance into a Tucker instance having length 8 in a higher dimension(the dimension being in the order of the bits required to specify one vertex in the 2-D Tucker instance). To complete the reduction, we now describe the process to fold each side(of length 8) of this Tucker instance into four sides of a 4-D hypergrid. This process retains the solution set of the original 2-D Tucker instance, does not add new solutions to it, and maintains the antipodal symmetry property, making it a valid reduction from 2-D Tucker.

Lemma 3.4. *An n-D General Octahedral Tucker instance (G, V_n) with length of some dimension 8, can be reduced to an $(n+3)$-D Tucker instance (G, V_{n+3}), by folding the length 8 dimension into 4 dimensions of length 2 each.*

Proof. The proof is divided into two parts: first we describe the folding process of a dimension of length 8 into four dimensions of length 2 each, followed by the second part where we prove that this process is a valid reduction from an n-D Tucker instance to an $(n+3)$-D Tucker instance.

We fold a dimension of length 8 into four dimensions of length 2 each, by embedding the n-D Tucker instance into an $(n+3)$-D Tucker instance. Vertices in these Tucker instances have same coordinates in almost all dimensions. The difference is in the coordinate of one old dimension(which is being folded), which has length 8 in the input instance and length 2 in the new one, and the 3 additional coordinates of new dimensions which are not present in the input instance. We call these dimensions the **differentiating dimensions**.

[3]Actually, the order of dimensions doesn't influence the correctness of Stairwise Lemma.

Sr.No.	$< \mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3} >$	**Sr.No.**	$< \mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3} >$
1	$< -1, 0, -1 >$	5	$< 0, 0, -1 >$
2	$< -1, 0, 0 >$	6	$< 0, 1, -1 >$
3	$< -1, 1, -1 >$	7	$< 0, 1, 0 >$
4	$< -1, 1, 0 >$		

Table 3: Coordinate values of environment vertices in a 3-D cube

We embed the n-D graph in the $(n + 3)$-D graph. Table 2 specifies the coordinates in the differentiating dimensions of a vertex mapped from the old n-D instance into the new $(n + 3)$-D instance(Without loss of generality, we assume the coordinates of vertices of the dimension to be folded lie in $\{-4, -3, .., 4\}$). The mapping of vertices is also shown as black colored vertices in Figure 7.

To help visualise the fold process, we illustrate the folding of a side of length 6 into three sides of length 2 each, in Figure 5. The separating and environment vertices are marked in Figure 6.

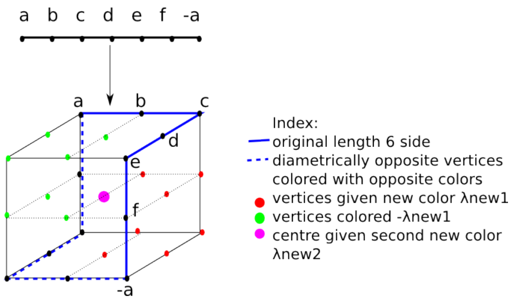
a b c d e f -a
a
b
c
d
e
f
-a
Index:
original length 6 side
diametrically opposite vertices colored with opposite colors
vertices given new color λnew1
vertices colored -λnew1
centre given second new color λnew2

Figure 5: Folding a dimension of length 6 into three dimensions of length 2 each.

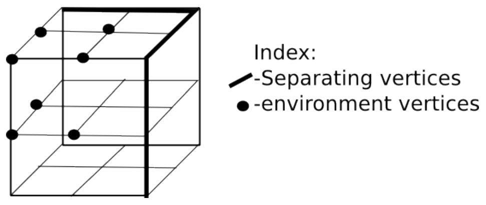

Figure 6: Separating and Environment vertex sets in a 3-D cube

We extend the idea of the coloring scheme of the length-6 reduction to the length-8 reduction, as shown in Figure 7. Intuitively, the 4 differentiating dimensions together form a 4-D hypergrid. Classifying all vertices of this hypergrid on the basis of their first coordinate value, we divide the hypergrid into 3 cubes $< -1, *, *, * >$, $< 0, *, *, * >$ and $< 1, *, *, * >$. The coloring scheme of the new instance divided in this way is as shown in Figure 7. Vertices mapped from the n-D Tucker instance are colored using the old colors. Vertices diametrically opposite to these vertices are given the opposite colors. For better exposition, the set of vertices of each cube is further divided into two groups of vertices, so called the 'separating' vertices and the 'environment vertices'. The coordinates

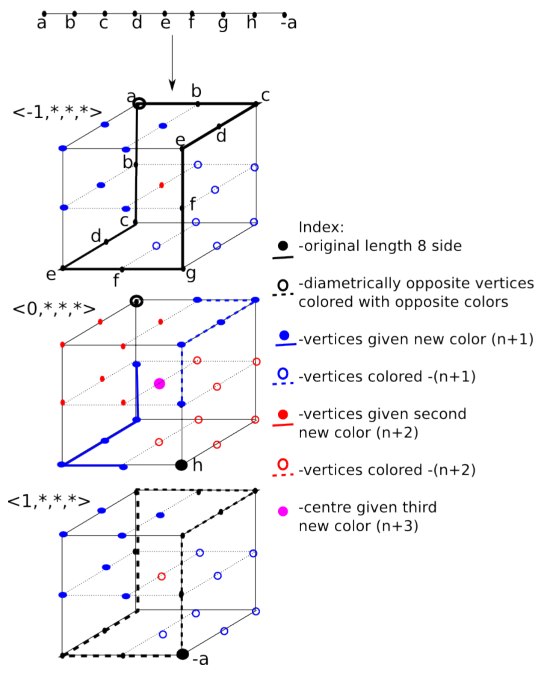
a b c d e f g h -a
<-1,*,*,*>
a
b
c
d
e
f
g
<0,*,*,*>
h
<1,*,*,*>
-a
Index:
-original length 8 side
-diametrically opposite vertices colored with opposite colors
-vertices given new color (n+1)
-vertices colored -(n+1)
-vertices given second new color (n+2)
-vertices colored -(n+2)
-centre given third new color (n+3)

Figure 7: Coloring scheme of new instance after folding one side of length 8 into 4 sides of length 2 each

Sr.No.	$< \mathbf{s_1, s_2, s_3} >$	**Sr.No.**	$< \mathbf{s_1, s_2, s_3} >$
1	$< -1, -1, -1 >$	4	$< -1, 0, 1 >$
2	$< -1, -1, 0 >$	5	$< -1, 1, 1 >$
3	$< -1, -1, 1 >$	6	$< 0, 1, 1 >$

Table 4: Coordinate values of separating vertices in a 3-D cube

of vertices belonging to each set, denoted by $< s_1, s_2, s_3 >$ and $< e_1, e_2, e_3 >$ respectively, are enumerated in Tables 4 and 3 respectively. Figure 7 marks these sets in a cube.

Formally, the coloring function of the new Tucker instance is as specified in the Table 5 also shown in Figure 7. The coloring functions of the $(n+3)$-D graph and the n-D graph are denoted by $g'(.)$ and $g(.)$ respectively. $< a, b, c, d >$ denotes the color of all vertices whose coordinates in the dimension shrunk and the new ones added are a, b, c, d respectively, $< k >$ denotes the color of the vertex in the original instance that has coordinate k in the dimension shrunk, and $< a, b, c, d >$ mapped to $< k >$ means $g'(a, b, c, d) = g(k)$, when the coordinates of all $g'(a, b, c, d)$ and $g(k)$ in all other dimensions are the same. In case of conflicting rules for some vertex, the rule higher in the Table is given higher priority.

This coloring function, by definition, is antipodally symmetric. If this coloring scheme now retains all complementary edges of the n-D Tucker instance, and does not add new ones, then it will be a valid reduction. We now proceed to the second part of the proof to argue the correctness of the reduction. We prove three assertions to do so:

1. Edges of the form (u, v), where u is a vertex mapped from the n-D Tucker instance, and v is not, are not complementary

Sr.No.	$<\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}>$	$<\mathbf{k}>$	**Sr.No.**	$<\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}>$	$<\mathbf{k}>$ or $\pm(\mathbf{n}+\mathbf{x})$, $\mathbf{x}\in\{\mathbf{1},\mathbf{2},\mathbf{3}\}$
1	$<-1,-1,-1,-1>$	<-4>	10	$<-1,e_1,e_2,e_3>$	(n+1)
2	$<-1,-1,-1,0>, <-1,0,-1,-1>$	<-3>	11	$<-1,-e_1,-e_2,-e_3>$	-(n+1)
3	$<-1,-1,-1,1>, <-1,1,-1,-1>$	<-2>	12	$<-1,0,0,0>$	(n+2)
4	$<-1,-1,0,1>, <-1,1,0,-1>$	<-1>	13	$<1,*,*,*>$	$-<-1,*,*,*>$
5	$<-1,-1,1,1>, <-1,1,1,-1>$	<0>	14	$<0,s_1,s_2,s_3>$	$(n+1)$
6	$<-1,0,1,1>, <-1,1,1,0>$	<1>	15	$<0,-s_1,-s_2,-s_3>$	$-(n+1)$
7	$<-1,1,1,1>$	<2>	16	$<0,e_1,e_2,e_3>$	$(n+2)$
8	$<0,1,1,1>$	<3>	17	$<0,-e_1,-e_2,-e_3>$	$-(n+2)$
9	$<1,1,1,1>$	<4>	18	$<0,0,0,0>$	$n+3$

Table 5: Coloring function of the Tucker instance reduced from a lower dimension instance.$<\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}>$ denotes the color $g'(a,b,c,d)$ assigned to vertices in the new instance that have coordinates a,b,c,d in the dimension shrunk and the 3 new dimensions added. $<\mathbf{k}>$ denotes the color $g(k)$ of the vertex having coordinate k in the dimension shrunk. A number $\pm(\mathbf{n}+\mathbf{x})$, $\mathbf{x}\in\{\mathbf{1},\mathbf{2},\mathbf{3}\}$, denotes the new color $\pm(n+x)$ assigned to the corresponding vertex in the new instance. $*$ denotes 'don't care', where the coordinate of the vertex in the respective dimension can be any of $\{-1,0,1\}$.

2. Vertices colored with new colors($(\pm(n+1), \pm(n+2), n+3)$) do not form complementary edges(i.e., no new solutions are formed)

3. The adjacencies of vertices embedded from the n-D Tucker instance are retained(i.e., all old solutions are retained and no new solutions connecting old vertices are formed)

The first statement is trivial, as all vertices mapped from the old Tucker instance are colored using some color from $\{\pm1,\pm2,..\pm n\}$, while the other vertices are colored using one of the new colors $\{\pm n+1,\pm n+2,n+3\}$. These edges, thus, can never be complementary. The second and third statements are proved separately in the lemmas 3.5 and 3.6, thus completing the reduction. □

Lemma 3.5. *While folding a dimension of length* 8 *of a general Tucker instance as described in lemma 3.4, vertices colored with new colors* $\{\pm(n+1),\pm(n+2),n+3\}$ *do not form complementary edges.*

Proof of Lemma 3.5. We make the following observations on the new instance from Figure 7:

1. No vertex belonging to the top cube is adjacent to any vertex in the bottom cube, as these have coordinate values -1 and 1 respectively in the first differentiating dimension, thus violate SOTT requirements.

2. Vertices belonging to the environment vertex set of the top cube are of the form $<-1,-1/0,0/1,-1/0> \setminus <-1,0,0,0>$. Vertices belonging to the separating vertex set of the middle cube are of the form $<\mathbf{0},-1,-\mathbf{1},*>$ or $<\mathbf{0},-\mathbf{1},*,1>$ or $<\mathbf{0},0,1,\mathbf{1}>$. Each of these vertices has some pair of dimensions(highlighted in bold in each case), which together has coordinate values that conflicts SOTT requirements

when compared with the values of the environment set vertices in the top cube. Thus, environment set vertices in the top cube(and similarly those in the bottom cube) cannot be adjacent to separating set vertices in the middle cube.

3. Similar to the reasoning in the second point, separating vertices in the top and bottom cube cannot be adjacent to environment vertices in the middle cube.

4. Environment Vertices and the set of their diametrically opposite vertices in the same cube facet have coordinate values 1 and -1, or vice versa, in some differentiating dimension, thus cannot be adjacent to each other.

To prove the lemma, we prove the correctness of the statement for vertices colored using distinct new colors separately. Vertices colored using $(n+1)$: These are the environment vertices in the top and bottom cube, and the vertices diametrically opposite to the separating set vertices in the middle cube. The observations made previously affirm that the top environment vertices cannot be adjacent to $-(n+1)$ colored vertices in the middle cube(point 2), or those in the bottom cube(point 1). Also, $-(n+1)$ colored vertices in the top cube cannot be adjacent to these vertices(point 4). Thus, $n+1$ colored vertices in the top cube are not adjacent to any $-(n+1)$ colored vertex. Similarly, we prove the lemma for $n+1$ colored vertices in the middle and bottom cubes, and for the $n+2$ colored vertices in the middle cube. The $n+2$ colored vertex in the top cube has the form $< -1, 0, 0, 0 >$. Thus by SOTT conditions, apart from vertices in the top cube, it is only adjacent to the center vertex in the middle cube, hence not adjacent to any $-(n+2)$ colored vertex. Similarly we prove the lemma for the $-(n+2)$ colored vertex in the bottom cube. Also, no vertex in the hypergrid has color $-(n+3)$, thus invalidating the existence of a $\pm(n+3)$ complementary edge. Thus, no complementary edges exist with adjacent vertices of colors $\pm(n+1)$, $\pm(n+2)$ or $\pm(n+3)$ □

Lemma 3.6. *While folding a dimension of length* 8 *of a general Tucker instance as described in lemma 3.4, the adjacencies of vertices embedded from the n-D Tucker instance are retained.*

Proof of Lemma 3.6. To prove this lemma, we need to prove all edges that existed in the n-D instance exist in the $(n+3)$-D instance, and any edge that did not exist still doesnt. We prove these two parts separately. First, assume two vertices u and v in the n-D instance are adjacent to each other. Then, by the definition of general Octahedral triangulation, all coordinates of these vertices differ at most by 1, i.e.

$$|u_i - v_i| \leq 1, \ \forall i \in [n] \tag{1}$$

where $u_i(v_i)$ is the coordinate of $u(v)$ in the i^{th} dimension.

After applying the fold described in lemma 3.4, all coordinates of these vertices remain the same, except for those of the last dimension which is now shrunk. As the last coordinate differed by atmost one, the new coordinates of this dimension, and the three new dimensions added, also differ by atmost one, as can be verified by the list of new coordinates specified in the proof of lemma 3.4(For instance, if the coordinate of u was -3 in the n-D graph, it's coordinates in the changed and new dimensions are $(-1, -1, -1, 0)$(and $< -1, 0, -1, -1 >$). The coordinate of v in the n-D graph can only be one of $\{-4, -3, -2\}$ as u and v are adjacent. In these cases, the new coordinates are $(-1, -1, -1, -1)$, $(-1, -1, -1, 0)$ or $(-1, -1, -1, 1)$(and $< -1, 0, -1, -1 >$ or $< -1, 1, -1, -1 >$) respectively, each of which is adjacent to u in the $(n + 3)$-D graph.

On the other hand, suppose u and v are not adjacent in the n-D instance. Then, the difference in at least one coordinate is greater than 1. If this coordinate was not shrunk by the lemma 3.4, then it remains the same in the $(n + 3)$-D instance too, implying u and v cannot be adjacent in the new graph too. If this coordinate was shrunk, then as can be seen from the list mapping differentiating coordinates in the two instances, they are still not adjacent. This can be verified by observing that any two sets of new coordinates, that are mapped from old coordinates differing by atleast 2, have a greater coordinate in one dimension in the first set and a smaller one in another dimension, or there is at least one dimension with coordinate 1 in one set and -1 in the other. Either of these cases contradicts general Octahderal Tucker conditions, implying these sets of coordinates of the vertices u and v are not adjacent in the $(n + 3)$-D instance too. $\square$

Combing all lemmas in this section, we prove Octahedral Tucker is PPA-hard. Moreover, we show the PPA-completeness of Octahedral Tucker based Section 2.2 and Section 3.

4 Remarks and Discussion

In this paper, we prove Octahedral TUCKER is PPA-complete, which has been open for a long time. Several other problems like Kneser, Ham-Sandwich, Necklace-Splitting and SMITH show promise to be complete for PPA. Specifically, the Kneser problem from graph theory, if proved PPA-complete, would be to the best of our knowledge the first graph theoretical PPA-complete problem. Matousek in [12] has reduced Kneser to Octahedral Tucker. We believe resolving the computational complexity of Octahedral Tucker will provide insight for resolving that of Kneser too. The SMITH problem is a classical mystery that has managed to remain unresolved for decades. We think resolving Kneser's complexity will provide pathways towards solving this. The ultimate goal of resolving the computational complexity of specific problems would be to find a relation between PPA and PPAD. As of now, we only know $PPAD \subseteq PPA$. We would like to resolve if the inclusion is strict.

5 Acknowledgements

We would like to thank Prof. Sundar Vishwanathan for introducing this problem to us and his valuable contribution during the early stages of this project, Simon's Institute for the Theory of Computing, from where we learnt the work [1] of James Aisenberg, Maria Luisa Bonet, and Sam Buss. Research results reported in this work are partially supported by the National Natural Science Foundation of China (Grant No. 61632017).

References

[1] James Aisenberg, Maria Luisa Bonet, and Sam Buss. 2-D Tucker is PPA-Complete. *Technical Report ECCC-TR15-163, Electronic Colloquium on Computational Complexity*, 2015.

[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] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the Complexity of Computing Two-player Nash Equilibria. *J. ACM*, 56(3):14:1–14:57, May 2009.

[4] Xiaotie Deng, Jack R. Edmonds, Zhe Feng, Zhengyang Liu, Qi Qi, and Zeying Xu. Understanding PPA-Completeness. In Ran Raz, editor, *31st Conference on Computational Complexity (CCC 2016)*, volume 50 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:25, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[5] Ky Fan. A Generalization of Tucker's Combinatorial Lemma with Topological Applications. *Annals of Mathematics*, 56(3):431–437, 1952.

[6] Robert M Freund and Michael J Todd. A constructive proof of Tucker's combinatorial lemma. *Journal of Combinatorial Theory, Series A*, 30(3):321 – 325, 1981.

[7] Katalin Friedl, Gábor Ivanyos, Miklos Santha, and Yves F. Verhoeven. *Locally 2-Dimensional Sperner Problems Complete for the Polynomial Parity Argument Classes*, pages 380–391. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[8] Paul W. Goldberg. A Survey of PPAD-Completeness for Computing Nash Equilibria. *CoRR*, abs/1103.2709, 2011.

[9] Michelangelo Grigni. A Sperner Lemma Complete for PPA. *Inf. Process. Lett.*, 77(5-6):255–259, March 2001.

[10] Emil Jerábek. Integer factoring and modular square roots. *CoRR*, abs/1207.5220, 2012.

[11] Shiva Kintali. List of PPAD-complete problems. *https://en.wikipedia.org/wiki/List_of_PPAD-complete_problems*.

[12] Jiří Matoušek. A Combinatorial Proof of Kneser's Conjecture. *Combinatorica*, 24(1):163–170, 2004.

[13] Christos H. Papadimitriou. On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence. *J. Comput. Syst. Sci.*, 48(3):498–532, June 1994.

Appendix

A Complete Proof for Octahedral Tucker is in PPA.

Proof of Theorem 2.6. Octahedral Tucker is a special case of the General Octahedral Tucker problem, which itself is a special case of General Tucker. As General Tucker is in PPA[13, 1], the theorem follows.

For completeness of presentation, we have added a proof that n-D Octahedral Tucker belongs to PPA here. The proof is based on Papadimitriou's proof [13] who refers to an idea started by Todd and Freund [6].

To prove Octahedral Tucker is in PPA, we reduce Octahedral Tucker to AEUL. We first introduce the concept of 'admissible simplices', which are sets of vertices in the Octahedral Tucker hypergrid, and create a sequence of these simplices where every admissible simplex can have at most two neighbours. Additionally, the singleton set containing only the origin in it, is an admissible simplex and has degree one. This sequence of simplices, with the singleton set, forms the input to the AEUL graph. As required for membership in PPA, we then describe polynomial time algorithms that find neighbours of any vertex(admissible simplex) in the AEUL graph, completing the reduction.

We now introduce the notation used in the proof:

Let $\boldsymbol{Z_n} := \{1, 2, \cdots, n\}$ are indices given to n dimensions and $\boldsymbol{Q} \subseteq \{\pm 1, \pm 2, \cdots, \pm n\}$ is a set of axes in these dimensions. Q is called a **d-orthant index set** if $|Q| = d$ and $\forall i \in Z_n,\ |\{i, -i\} \cap Q| \leq 1$. $\boldsymbol{T_Q} := \{\mathbf{t} : t_i \in \{-1, 0, 1\} \forall i \in Q, Q \ is\ a\ d-orthant\ index\ set\}$ is a hypergrid of $2^{|Q|}$ vertices containing the origin $0^{|Q|}$ in $|Q|$ dimension space, where the number of choice for the values in each dimension is 2. We denote $\boldsymbol{T} = T_{Z_n}$, the n-dimension hypergrid of length 2 that has the vertex set $\{-1, 0, 1\}^n$.

We call T_Q a **Q-orthant** if $\forall t \in T_Q\ t_i \in \begin{cases} \{0,1\} & \text{if } i \in Q \\ \{0,-1\} & \text{if } -i \in Q \\ \{0\} & \text{else} \end{cases}$.

Note that the Octahedral triangulation of T_{Z_n} induces a (possibly lower dimensional) triangulation on every $Q - orthant\ T_Q$.

Note that T_Q and T_{Z_n} are different in that the boundary vertices of T_{Z_n} are all non-zero in every coordinate but only half of boundaries of T_Q have that property. We should name the boundaries of T_Q coincident to T_{Z_n} its external boundaries, and the rest its internal boundaries.

Let $\mathbf{g}$ be the coloring function on T which is anti-symmetric on the boundary of T: $\forall \mathbf{x} \in T \backslash \{\mathbf{0^n}\}\ \mathbf{g}(\mathbf{x}) = -\mathbf{g}(-\mathbf{x})$. Without of loss of generality, we assume the color of the origin $\mathbf{0}^n$ to be 1: $g(\mathbf{0}^n) = 1$.

To define the nodes for the graph in the AEUL structure, we now introduce the concept of admissible simplices of the triangulation on the Octahedral grid T.

Definition A.1 (Admissibility). *A $(d-1)$-simplex $S = \{v^1, v^2, \cdots, v^d\}$ in the Octahedral triangulation of $T = T_{Z_n}$ is admissible, if the following conditions hold:*

- *$\boldsymbol{0}^n \in S$;*
- *$Q(S) = \{g(v^i) : i = 1, 2, \cdots, d\}$: Colors of S;*

- $S \subseteq T_{Q(S)}$*: S located in space index by its colors* $T_{Q(S)}$.

Note an admissible $(d-1)$-simplex S contains d vertices, each of which is assigned a different color. The set of non zero coordinates of any vertex in S is a subset of the set colors assigned to all vertices of S i.e., $\exists u \in S : t_c(u) \neq 0 \implies \exists v \in S : \; c = g(v)$. Further, there is a possibility that there is a d-simplex S' in $T_{Q(S)}$ such that it contains a vertex z of color $j = g(z)$ but $j \notin T_{Q(S)}$. Then S' is admissible in $T_{Q(S)\cup\{j\}}$ but not admissible in $T_{Q(S)}$.

Also, each admissible $(d-1)$-simplex S can be a face for up to two d-simplices in $T_{Q(S)}$, or is a face for one d-simplex of $T_{Q(S)}$ and S is a boundary face on $T_{Q(S)}$.

To define the neighbors of an admissible simplex, we take its dimension d to identify its orthant T_Q. Then the next steps become uniquely determined. Each admissible $(d-1)$-simplex S is either an interior face in the triangulation of $T_{Q(S)}$ or a face on the boundary of $T_{Q(S)}$.

In the former case, it is contained as a boundary of one admissible d-simplex S_1 of T_Q. Dependent on the color of the vertex in $\{v\} = S_1 \backslash S$, it divides into three cases. First, if $g(v) \notin Q(S)$, then S_1 is an admissible d-simplex in $T_Q(S_1)$ which becomes the neighbour of S in the AEUL graph (a case dimension rising). Second, if $g(v)$ is not 1 and $g(v) \in Q(S)$, then the other $(d-1)$-simplex of S_1 with exactly the same color as S is also an admissible $(d-1)$-simplex in $T_{Q(S)}$ and becomes the neighbour node of S. Third, $g(v) = 1$. In this case, we S has a nil edge as a new vertex of color 1 is on the boundary which has an antipodal image -1, forming a complementary edge with the origin.

In the latter case, the non-zero coordinates of vertices of S reduces by one, say the coordinate c_j. We mark the vertex of S of color c_j as v^{c_j}. Then $S\backslash\{v^{c_j}\}$ will be an admissible $(d-2)$-simplex, which is a lower dimension neighbour of S, unless $c_j = 1$. If $c_j = 1$, $-S$ will be an admissible $(d-1)$-simplex on T_{-Q} which will be made the neighbour of S. The two admissible $(d-1)$-simplices are in two antipodal orthant of T_{Z_n}: one $Q(S)$-orthant and another $Q(-S)$-orthant which is a $(-Q(S))$-orthant.

In the above discussion, all go through except the case where the new node is ± 1. In this case, we have an 1-simplex of complementary edge which is what we want. We mark this creates a nil edge and hence the admissible $(d-1)$-simplex leading to this complementary edge is an AEUL node of degree one.

With the above clarification, we complete the construction of the nodes and edges of the AEUL graph, with the origin as the given degree one node, and every node has degree no more than two.

$\square$

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il