Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 118 (2017)

Octahedral Tucker is PPA-Complete

Xiaotie Deng* Zhe Feng† Rucha Kulkarni‡

Abstract

Octahedral Tucker (a special case of the celebrated Tucker problem) is the natural computational problem based on Octahedral Tucker's lemma, a classical statement from algebraic topology. Like many fixed point results, this problem has been central to proving several important results from diverse fields of theoretical computer science. [20, 19, 14, 10]. Resolving the complexity of the natural computational problem associated with it was an important open question, also raised in [16, 2]. PPA (Polynomial Partity Argument on Graphs) is a complexity class defined in [17]. Computational versions of Tucker's lemma and the Borsuk-Ulam theorem were conjectured to be ideal candidates for PPA − Complete problems, as many problems thought to belong in PPA were reduced from these. While there has been some progress in finding PPA − Complete problems [9, 3, 2, 8], including 2-D Tucker[1], the 2-dimensional version based on Tucker's lemma, the n-dimensional Octahedral Tucker problem has evaded resolution.

In this paper, we resolve this decade old open question by proving n-dimensional Octahedral Tucker PPA − Complete. Our reductions also contribute two stand-alone folding techniques, *Fold* and *Wrap*, which are novel as far as we know and could be of broader interest. Additionally, during the reduction process, we define a new PPA − Complete problem General Octahedral Tucker, another computational problem based on Tucker's lemma that generalizes Octahedral Tucker, adding to the growing list of PPA − Complete problems.

*School of EECS, Peking University, Beijing, China. Email: xiaotie@pku.edu.cn.

†John A. Paulson School of Engineering and Applied Sciences, Harvard University, 33 Oxford Street, Cambridge, MA 02138, USA. Email: zhe_feng@g.harvard.edu. This work was partly done when the author was an undergraduate student in Shanghai Jiao Tong University, China.

‡Department of Computer Science, University of Illinois at Urbana-Champaign, 201 N Goodwin Avenue, Urbana, IL 61801. Email: ruchark2@illinois.edu. This work was done while visiting Shanghai Jiao Tong University, China.

[1]**In the rest of the whole paper, for simplifying presentation, n-D means n-dimensional.**

ISSN 1433-8092

1 Introduction

Octahedral Tucker's lemma, discovered in [20], is the following combinatorial statement:

Lemma 1 (Octahedral Tucker's Lemma:)**.** *If an n-dimensional hyper grid of length 2 in all dimensions is octahedrally triangulated (this is the first barycentric subdivision in [14] and we formally define it in Definition 5), and every vertex is assigned a color from the set $\{\pm 1, \pm 2.., \pm(n-1), \pm n\}$ such that diametrically opposite vertices on the boundary of the hyper grid get assigned complementary colors (i.e colors that have the same magnitude and opposite sign), then there always exists an edge (1-simplex) such that vertices adjacent to the edge have complementary colors.*

Octahedral Tucker is the natural computation problem arising from the lemma: Given an n-dimensional hyper grid satisfying the above constraints in the form of a polynomial time algorithm that gives the color assigned to any vertex, is there a polynomial time algorithm to find an edge with complementary colors on adjacent vertices. The formal definition of Octahedral Tucker is given later in Definition 2.

The Octahedral Tucker's lemma has been used to prove several results, the most widely used of which are the Borsuk-Ulam theorem from algebraic topology and the set covering Lusternik-Schnirelmann antipodal point theorems [20]. The computational Octahedral Tucker problem has been used to prove several results in discrete geometry like the Ham-Sandwich theorem (proves existence of a hyperplane that simultaneously bisects d point sets in $\mathbb{R}^d$), and in combinatorics like the Kneser-Lovasz theorem (proves chromatic number of Kneser graphs)[14], by describing reductions from Octahedral Tucker to computational problems corresponding to the theorems. These theorems in turn were central to proving numerous results in the respective areas. In more applied areas, fair division methods in algorithmic game theory use Octahedral Tucker as the core theorem to prove several results, for instance the necklace-splitting techniques [19].

Resolving the complexity of Octahedral Tucker will lead towards resolving that of all these important problems in several areas of theoretical computer science. The problem was raised as an open question by Pálvölgyi [16] and Aisenberg et al. [2] (and also referred to in the survey paper by Loera et al. [13]). In this paper, we resolve the question by proving Octahedral Tucker Complete for the complexity class PPA.

1.1 Related Work

PPA (Polynomial Parity Argument on Graphs) is a complexity class defined by Papadimitrou in [17]. In [17], the semantic complexity class TFNP (all Total search problems in NP) was divided into several syntactic classes, based on the nature of the proof of existence of a solution. PPA is one of these classes, and it contains all problems whose proof of existence of solution is the following combinatorial lemma: *Every graph has an even number of odd degree nodes.* Tucker and various other fixed point results, like the Sperner, Brouwer, Kakutani and Borsuk-Ulam theorems were proved to belong to PPA [17], thus making them candidates for possible PPA − Complete problems.

The Sperner, Brouwer and Kakutani theorems proved complete for PPAD [17] (Polynomial Parity Argument for Directed Graphs), a subclass of PPA. These led to completeness results for other theorems from other areas (whose proofs reduced the problem to any of these fixed point results). The most celebrated of these is perhaps the Nash Equilibrium theorem in Game Theory [7, 6]. With the completeness result establishing probable hardness of computing equilibria in general games, further investigations were then pursued in other directions, for example to find subclasses of games or other equilibrium notions that might be easy to compute [1, 15, 5], analyze

the smoothed complexity of computing equilibria [6], analyze their cryptographic complexity [4] and finding a PTAS for general Nash [18]. This has now led to a rich theory in equilibrium computation.

The Tucker and Borsuk-Ulam theorems however evaded resolution for several years. Other completeness results for PPA were found, for example Grigni's proof for the non-orientable 3-dimensional version of Sperner [12], and the locally 2-dimensional version of the same by Friedl et al. [11]. Deng et al. [8] characterized the Möbius band as a defining property of PPA − Complete discrete fixed point problems, by providing a unified PPA − Complete proof for several problems, including versions of Tucker and Sperner, defined on the möbius band[2]. Algebraic results were recently added to the collection, with Chevalley's theorem and the Combinatorial Nullstellensatz set of results proven PPA − Complete by Belovs et al. [3].

Originally defined for the octahedrally triangulated n-dimensional ball, Tucker's lemma holds true for any *antipodally symmetric* triangulation (i.e. triangulation in which every vertex on the simplex boundary has a diametrically opposite vertex). Different versions of Octahedral Tucker were defined, by changing the nature of triangulation and/or the dimension of the space considered, with the coloring constraints on the boundary retained. The computational complexity of these versions was investigated. These attempts were succesful, with Pálvölgyi [16] first proving PPAD-Hardness of 2-D Tucker, the 2-dimensional Tucker version of exponential side lengths and grid based triangulation. Aisenberg et al. [2] then proved PPA-Completeness of the same problem[3]. As a corollary, this leads to the Borsuk-Ulam theorem being PPA − Complete. Recently, Filos-Ratsikas and Goldberg [9] proved the Consensus Halving problem PPA − Complete, by reducing 2-D Tucker to Consensus Halving.

While these results have already increased the scope of PPA, resolving the complexity of the original candidate problem Octahedral Tucker was an important gap to cover. Resolving this question still leaves several more interesting avenues left to explore, for instance the complexity of: The SMITH problem, of finding a second hamiltonian path in an odd degree graph, The Kneser problem, of finding a monochromatic edge in a Kneser graph (given parameters n and k, vertices are k-element subsets of a set of n-integers, with edges between disjoint sets) colored using $n-2k+1$ colors (which is one less than the chromatic number of these graphs), the Necklace Splitting problem and the HamSandwich problem, all of which belong in PPA. The Necklace Splitting problem has been partially resolved as PPAD − Hard, as a corollary of the PPA − Complete proof of Consensus Halving [9]. Of the rest, Octahedral Tucker has been used in a constructive proof of existence of a monochromatic edge in a Kneser graph [14]. Thus resolving the complexity of Kneser could be the first direct step to pursue.

1.2 Technical Contribution

To prove Octahedral Tucker PPA − Hard, we define a new problem, General Octahedral Tucker. As figure 1 shows, Octahedral Tucker is a special case of General Octahedral Tucker. 2-dimensional (2-D) General Octahedral Tucker is a special case of 2-D Tucker. Theorem 9 proves 2-D General Octahedral Tucker is PPA − Hard, following the same technique of [2]. We then reduce 2-D General Octahedral Tucker to Octahedral Tucker. The reduction maps vertices of 2-D General Octahedral Tucker defined on a grid of size $2^n \times 2^m$ to vertices of $O(m+n)$-dimensional Octahedral Tucker, such that solutions to Octahedral Tucker can be mapped to solutions of 2-D General Octahedral Tucker in time polynomial in m and n.

The main challenges in the reduction were:

[2]Until this point, PPA seemed to capture the complexity only of non orientable structures

[3]This was the first PPA − Complete problem based in Euclidean space

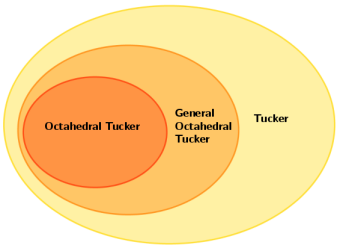


Figure 1: The new PPA – Complete problem General Octahedral Tucker, and its relation with Tucker and Octahedral Tucker.

1. **Reducing exponential side lengths to constant in polynomial steps**: The main goal of the reduction is to reduce the exponential side lengths of 2-D General Octahedral Tucker to length 2 sides, at the cost of increase in the number of dimensions. The challenge was to design a process that reduced side lengths exponentially, by allowing only a polynomial increase in the number of dimensions.
2. **Center Vertex**: Octahedral triangulation is such that the center vertex of the hyper grid is connected to *every other vertex*. In our reduction, we maintain the adjacencies between the vertices of General Octahedral Tucker (no new edges are formed between their mapped vertices in Octahedral Tucker, no edge gets deleted). As every vertex in 2-D General Octahedral Tucker is adjacent to at most 8 vertices, the center vertex cannot be mapped to some vertex of the General Octahedral Tucker grid. Further, if this vertex is assigned a color that has also been assigned to some vertex in 2-D General Octahedral Tucker, then the center vertex, along with the vertex diametrically opposite to the mapped vertex in Octahedral Tucker form a complementary edge. Thus, in all coloring schemes, the center vertex must be assigned a color distinct from that of every other vertex in the grid.
3. **Interior vertices of 2-D General Octahedral Tucker**: In 2-D General Octahedral Tucker, only vertices on the boundary of the grid satisfy the *valid property* (every vertex has a diametrically opposite vertex of complementary color). An exponential number of vertices in the interior of the grid do not. Once an interior vertex of 2-D General Octahedral Tucker is mapped to some vertex on the boundary of Octahedral Tucker, its diametrically opposite vertex in the Octahedral Tucker hyper grid automatically gets colored using the complementary color. This vertex should not be adjacent to any vertex mapped from the 2-D General Octahedral Tucker, to ensure no false solution (new complementary edge) get involved.

The reduction from 2-D General Octahedral Tucker to Octahedral Tucker involves two main folding techniques, which focus to overcome separate challenges: (1) *Fold*: where we recursively reduce an $(n-1)$-dimensional General Octahedral Tucker instance to an n-dimensional General Octahedral Tucker instance, by halving the length of one dimension and adding one constant length 8 side in a new dimension. (2) *Wrap*: where we iteratively reduce every length 8 dimension into 4 mutually orthogonal length 2 dimensions.

The *Fold* step targets the first challenge of ensuring a polynomial time reduction. By reducing every side length to half (and adding another side of only constant length in the process), we ensure every side length is a constant in a polynomial number of steps. Another aim here was to keep the interior vertices of 2-D General Octahedral Tucker in the interior of the new instance too, to avoid adding diametrically opposite vertices of complementary color. Thus, as figure 3 shows, we fold a side twice in a *snake-like* fashion, add one layer of extra vertices between the folds to insulate original grid vertices from becoming adjacent to each other, and pad the outer layers to keep the

vertices in the interior of the new instance. This results in length 8 in the new dimension added. We describe the lemma and other details (coloring function and adjacencies of the new instance) formally in section 2.

At the end of the *Fold* process, we have a General Octahedral Tucker instance of constant length 8 in every dimension. While applying the *Fold* process, the length in every dimension is more than 2, hence there is no *center vertex* to consider. A *Wrap* step reduces one side of length 8 into 4 mutually orthogonal sides of length 2. Intuitively, a length 8 dimension is folded along orthogonal sides of a length 2 4-dimensional hyper grid. After applying *Wrap* even once, dimensions of length 2 are introduced in the structure, hence there is a center vertex (intuitively the vertex at the center of the hyper grid along which the length 8 side was wrapped) to color. With each length 8 reduction, a new center vertex gets introduced, which is connected to every vertex, including all the previous center vertices. Also, reducing side lengths to two necessarily involves mapping every vertex to some boundary vertex. Both the *coloring center vertices* and *isolating complementarily colored vertices of interior* 2-*D* General Octahedral Tucker *vertices* problems are taken care of simultaneously in every *Wrap* step. Adding three new dimensions adds 3 new complementary color pairs. Two pairs of these colors are used to insulate the original vertices from each other and from their diametrically opposite vertices, and one color of the new pair is used to color the center vertex now formed. One color is left unused. Details of the reduction, with comments on why other possibly simpler ideas do not work, are specified in section 3.

The *Wrap* and *Fold* are stand-alone folding techniques independent of the constraints imposed by the Tucker lemma. These could be of independent interest in applications where problems have geometric interpretations, are based in different dimensional spaces and need to be related to each other.

Presentation: In the rest of the paper we describe the high level idea as clearly as possible. Complete proofs to all new ideas are attached as Appendices. The next section starts with the definitions of the complexity class PPA and Octahedral Tucker, with a proof of its membership in PPA. We also define General Octahedral Tucker, required for the hardness reduction proof. Section 3 then discusses the reduction from 2-D General Octahedral Tucker to Octahedral Tucker. We conclude by remarks on applying our concepts to other problems in future.

2 Preliminaries

2.1 The PPA Class

PPA (Polynomial Parity Argument on graphs) is a complexity class introduced by Papadimitriou in his seminal paper [17]. It is the set of all problems in $NP \cap coNP$ that are guaranteed to have a solution, whose proof of existence is the following combinatorial statement:*Every graph has an even number of odd degree nodes* That is, all problems that can be reduced to the problem of finding *another* odd degree node in a graph (the first given as input), belong in PPA. An equivalent version of PPA, more commonly used, assumes the degree of every node in the graph to be at most 2.

Every problem in PPA thus has a natural path following argument associated with it, the description of which comprises the proof of membership of the problem in PPA. Octahedral Tucker is one of these, defined in the proceeding section.

2.2 Octahedral Tucker: Definition and membership in PPA

Originally defined for octahedral triangulation of an n-dimensional Ball of diameter 2, Tucker's lemma holds true for *antipodally symmetric* triangulations (for every edge $\{x, y\}$ on the boundary

of the triangulation, the negation $\{-x, -y\}$ is also an edge of the triangulation) of n-Ball of any diameter too.

Formally, *antipodally symmetric* triangulation T of the closed n-Dimensional ball $B^n \subset \mathbb{R}^n$ is the triangulation such that if each simplex $\sigma \in T \cap S^{n-1}$, then $-\sigma \in T$, where the negation of a simplex is the negation of each of its vertices and S^{n-1} is the boundary of B^n. We now define the general lemma (which is required for the intermediate General Octahedral Tucker definition) and all notation required for further discussion.

Lemma 2. *(Tucker's Lemma [20]): Let T be an antipodally symmetric triangulation of B^n, and let g be a mapping (coloring function) from the vertices of T to $\{\pm 1, \pm 2 ... \pm n\}$ which satisfies the valid property: if vertex v is on the boundary, then $g(-v) = -g(v)$, then there exists a 1-simplex (edge) $\{v_1, v_2\}$ in T with $g(v_1) = -g(v_2)$, which is called complementary edge.*

Tucker's lemma directly leads to the computational Tucker problem: Given an antipodally symmetric triangulated n-dimensional Ball, and a valid coloring function on its vertices, find a complementary edge in the Ball. Equivalently, the lemma can be stated for any simplex on which an antipodal symmetric triangulation can be realized. For simplicity, we define Tucker on a *hyper grid*.

Definition 3 (Hyper Grid)**.** *$V_n = \{\boldsymbol{p} = (p_1, p_2, ..., p_{n-1}, p_n) \in \mathbb{Z}^n, \forall i, -\frac{N_i}{2} \leq p_i \leq \frac{N_i}{2}\}$ is a hyper grid, [4] where N_i is the length of ith dimension. The boundary of the hyper grid is*

$$\text{Boundary}(V_n) = \left\{\boldsymbol{p} \middle| \exists i,\ s.t.\ p_i \in \{-N_i/2, N_i/2\}\right\}$$

Let $K_{\boldsymbol{p}} = \{\boldsymbol{q} : q_i \in \{p_i, p_i + 1\}\}$ be the unit hyper grid in V_n associated with the vertex $p \in V_n$.

From now on, we focus on Tucker and its various versions (Octahedral Tucker, General Octahedral Tucker) defined on V_n and corresponding triangulation of V_n (denoted T_n). As a start point, n-D Tucker can be formally defined as:

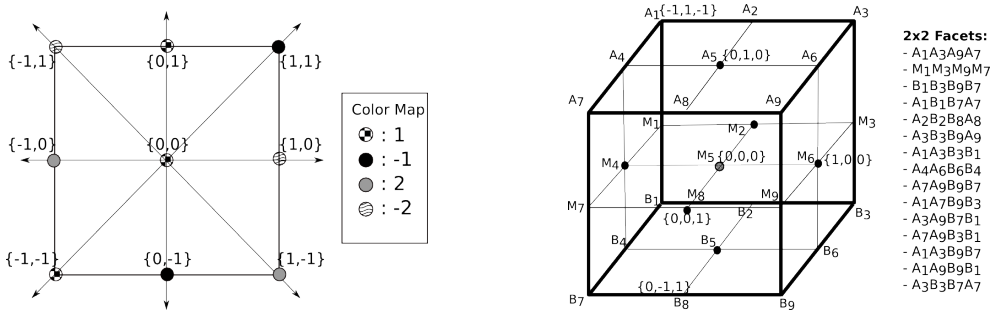
Definition 4 (n-D Tucker)**.** *The input of n-D Tucker is a pair (G, T_n), where T_n is an antipodally symmetric triangulation of n-dimensional hyper grid V_n centered at $\boldsymbol{0}$ and G is a polynomial-time machine, which generates a valid coloring function (defined in Lemma 2) $g : V_n \to \{\pm 1, \pm 2, ..., \pm n\}$, i.e. $\forall \boldsymbol{p} \in \text{Boundary}(V_n)$, $g(-\boldsymbol{p}) = -g(\boldsymbol{p})$. The output of n-D Tucker is a complementary 1-simplex, i.e an edge $(\boldsymbol{p}, \boldsymbol{q})$ s.t. $g(\boldsymbol{p}) = -g(\boldsymbol{q})$.*

Octahedral Tucker is a special case of Tucker, defined on a hyper grid of length 2 in all dimensions. The triangulation of the hyper cube is the standard first barycentric subdivision, but we define the same in a form useful for the General Octahedral Tucker definition.

Definition 5 (Standard Octahedral Tucker Triangulation (**SOTT**))**.** *In an n-D hyper grid $V_n = \left\{\boldsymbol{p} \in \mathbb{Z}^n \middle| \forall i \in [n], p_i \in \{-1, 0, 1\}\right\}$ 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 $\boldsymbol{p} \succsim \boldsymbol{q}, \boldsymbol{p}, \boldsymbol{q} \in V_n$ iff $\forall i \in [n], p_i \succsim q_i$. The Standard Octahedral Tucker Triangulation of V_n is defined as: $\forall \boldsymbol{p}, \boldsymbol{q} \in V_n$, $\boldsymbol{p}$ and $\boldsymbol{q}$ are linked iff $\boldsymbol{p} \succsim \boldsymbol{q}$ or $\boldsymbol{q} \succsim \boldsymbol{p}$.*

On a high level, **SOTT** is a recursive triangulation, where we start with a 2-D (3^2 sized) grid, and triangulate it as shown in figure 2. For a higher dimensional n-D grid (of 3^n size), for all choices of 3^{n-1} vertices, using the preference relation (defined for n-coordinates) we verify they form a valid $(n-1)$-D instance, and if they do, apply the triangulation of the $(n-1)$-D grid between them. The preference relation allows local determination of the triangulation in polynomial time (additionally, without need of visualizing the structure).

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



(a) 2-D Octahedral Tucker **(b)** 3-D Octahedral Tucker

Figure 2: Instances of 2-D and 3-D Octahedral Tucker. For the 3-D instance, instead of drawing all edges, the 15 2×2 facets in the 3-D hyper grid, each triangulated by 2-D Octahedral Triangulation, are enumerated in the column to the right.

Definition 6 (n-D Octahedral Tucker)**.** *n-D* Octahedral Tucker *($n \geq 2$) is the special case of n-D* Tucker *(G, T_n) defined on V_n that satisfies*

1. *The side length of each dimension is exactly 2, i.e. $\forall \boldsymbol{p} \in V_n, \forall i \in [n], p_i \in \{-1, 0, 1\}$*
2. *T_n is the Standard Octahedral Tucker triangulation (**SOTT**) of V_n. Sample instances of 2-D and 3-D* Octahedral Tucker *are shown in Figure 2.*

Theorem 7 ([2, 17])**.** Octahedral Tucker *is in* PPA

The proof of this theorem closely resembles Aisenberg et al. [2] and Papadimitriou [17] of membership of General Tucker in PPA, and for the completeness of presentation we include it in appendix A. Proving Hardness was a more challenging task, and involves new ideas presented in the next section.

2.3 General Octahedral Tucker

Towards proving the PPA-hardness of Octahedral Tucker, we define another special case of n-D Tucker: General Octahedral Tucker and the corresponding triangulation called the *General Octahedral Tucker triangulation (**GOTT**).*

Definition 8 (General Octahedral Tucker)**.** General Octahedral Tucker *is the special case of n-D Tucker (G, T_n) defined on V_n which satisfies*

1. *V_n has lengths $\{N_1, N_2, ..., N_{n-1}, N_n\}$ in the respective dimensions, such that $\forall i \in [n], N_i = 2k_i, k_i \in \mathbb{N}^+$;*
2. *T_n is the General Octahedral Tucker triangulation (**GOTT**) of V_n, where every length-2 hyper grid $H_{\boldsymbol{p}} = \{\boldsymbol{q} : q_i \in \{p_i - 1, p_i, p_i + 1\}\}$, termed as* Octahedral Hypergrid*, centered at vertex $\boldsymbol{p}$ such that for any $i \in [n]$*

$$p_i = \begin{cases} 2m_i + 1, m_i \in \mathbb{N} \text{ and } -\frac{k_i}{2} \leq m_i \leq \frac{k_i}{2} - 1 & \text{if } N_i \geq 4 \\ 0 & \text{if } N_i = 2 \end{cases}$$

*is triangulated by Standard Octahedral Tucker Triangulation **SOTT**. Note, when $N_i = 2$ for all $i \in [n]$,* General Octahedral Tucker *is* Octahedral Tucker*.*

Theorem 9. 2-*D* General Octahedral Tucker *is* PPA − Hard*.*

The proof follows the same technique of [2] with minor modifications. A sketch of the proof is provided in Appendix B. Figure 1 uncovers the relationship among General Octahedral Tucker, Octahedral Tucker and Tucker. With the necessary tools in hand, we can now turn to the main result of the paper:

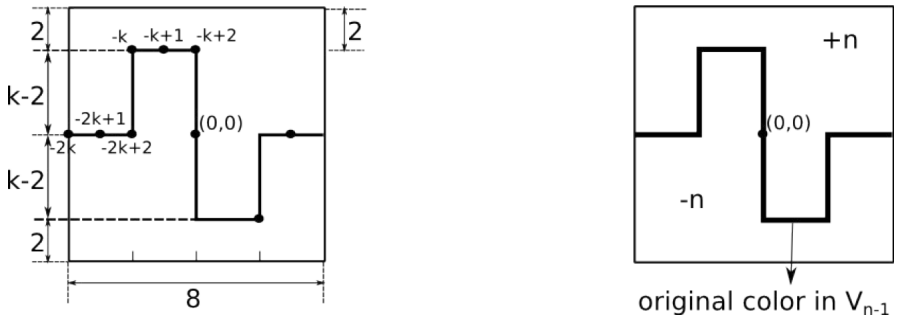
3 Octahedral Tucker is PPA – Hard

We prove Octahedral Tucker is PPA – Hard by reducing 2-D Tucker with size $2^m \times 2^n$ to $O(m+n)$-D Octahedral Tucker[5] in polynomial time, in two stages: the *Fold* and *Wrap*.

3.1 Reduction Stage 1: Reducing side lengths to 8

We divide this subsection into two parts: First, we introduce the *Fold* lemma: fold $(n-1)$-D Tucker into n-D Tucker by halving the length in one dimension and adding an extra dimension of constant length 8. We then show how to use this lemma recursively to reduce 2-D Tucker to a higher dimension Tucker instance with length in each dimension 8.

Lemma 10 (Fold Lemma)**.** *Given an* $(n-1)$*-D* General Octahedral Tucker *instance* (G, T_{n-1}) *on* V_{n-1} *with length of each dimension* $\{N_1, N_2, ..., N_{n-2}, N_{n-1}\}$*, where* $N_{n-1} = 4k_{n-1}$*, for some* $k_{n-1} \geq 4$*, we can reduce it to* n*-D* General Octahedral Tucker (G', T_n) *on* V_n*, where* V_n *has lengths of each dimension* $\{N_1, N_2, ..., N_{n-2}, N'_{n-1}, N'_n\}$*, where* $N'_{n-1} = \frac{N_{n-1}}{2} = 2k_{n-1}, N'_n = 8$*.*



(a) Folding and embedding. **(b)** Adding colors.

Figure 3: Reduce $(n-1)$-D General Octahedral Tucker on V_{n-1} with length of each dimension $\{N_1, N_2, \cdots, N_{n-2}, N_{n-1}\}$ to n-D General Octahedral Tucker. This is a 2-D projection of V_n along with $(n-1)$th dimension and nth dimension. (a): Embedding $(n-1)$-D General Octahedral Tucker on V_{n-1} with $N_{n-1} = 4k$ $(k > 2)$ in n-D General Octahedral Tucker with $N'_i = N_i (1 \leq i \leq n-2)$, $N'_{n-1} = 2k$ and $N'_n = 8$. (b): Adding additional colors $\pm n$ in T_n.

The thousand words worth Figure 3 and Figure 4 show the folding technique. Speaking at a high level, we fold one dimension in a *snake-like* fashion shown in Figure 3 and keep other dimensions unchanged. This *snake-like* embedding of the $(n-1)$-D instance (G, T_{n-1}) in the n-D instance (G', T_n), results in adding one extra dimension with constant length 8. To illustrate how this works, we show the toy example of reducing 2-D Tucker (with 16×16 size) to 3-D Tucker (with $16 \times 8 \times 8$ size) in Figure 4 (For better visibility, we do not show all edges of the triangulation). Ideally, to keep the width of the new dimension small, we want to pack the vertices as close as possible. We also want the size of the dimension being folded to reduce as much as possible. Folding in half allows keeping the extra dimension's length a constant 8, providing the required exponentially fast reduction.

The proof of the Fold lemma is stated in Appendix D. To apply the *Fold* lemma to a 2-D General Octahedral Tucker instance recursively, we require the lengths of all new dimensions generated in the intermediate steps to be even. We state the following lemma that allows us to do so by restricting attention to a specific class of 2-D General Octahedral Tucker:

[5] w.l.o.g., we refer to this as n-D Octahedral Tucker hence forth

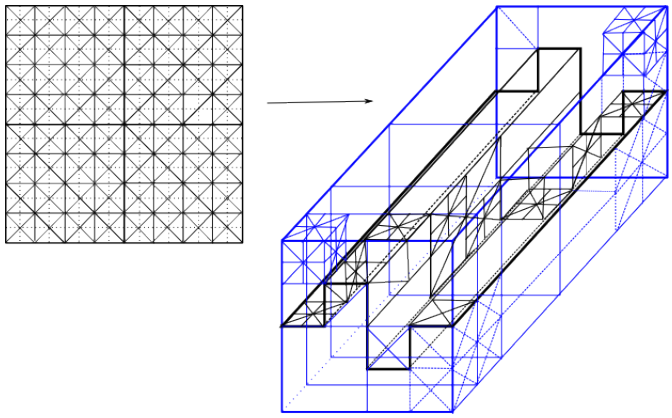


Figure 4: Fold 2-D General Octahedral Tucker (both dash lines and solid lines are triangulation) on 16×16 grid into 3-D General Octahedral Tucker on $16 \times 8 \times 8$ 3-D hyper grid.The black edges in the 3-D instance are mapped from the 2-D one. One can verify that the black triangulation, which is the original 2-D triangulation, and the blue one, which is that of the 3-D instance, coincide, thus retaining all adjacencies of original vertices

Lemma 11. *Any 2-D* General Octahedral Tucker *instance* (G, T_2) *on* V_2 *can be reduced to a 2-D* General Octahedral Tucker *instance* (G', T_2') *on* V_2' *whose lengths in the two dimensions are* 2^m *and* 2^n*, for some* $m, n \in \mathbb{Z}$*.*

The complete proof of Lemma 11 is presented in Appendix C, and the following theorem now immediately follows:

Theorem 12. *Any 2-D* General Octahedral Tucker *instance* (G, T_2) *defined on* V_2*, where the lengths of the two dimensions of* V_2 *are* N_1 *and* N_2 *respectively, can be reduced to an* $O(\log N_1 + \log N_2)$*-D* General Octahedral Tucker *on* $V_{O(\log N_1 + \log N_2)}$ *with length in each dimension* 8*.*

Proof. Based on lemma 11, w.l.o.g. we assume length of two dimensions of 2-D General Octahedral Tucker instance (G, T_2) are 2^m and 2^n. Using *Fold* Lemma recursively, we can reduce the 2-D General Octahedral Tucker (G, T_2) to an $O(m+n)$-D General Octahedral Tucker with length in each dimension 8 in polynomial time[6]. □

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

Theorems 9 and 12 summarize our work until now. We now present the final piece completing the puzzle, the Wrap lemma:

Lemma 13. *(Wrap Lemma) An n-D* General Octahedral Tucker *instance* (G, T_n) *on* V_n *with length of each dimension* $\{N_1, N_2, ..., N_{n-1}, 8\}$*, can be reduced to an* $(n+3)$*-D* General Octahedral Tucker *instance* (G, T_{n+3}) *on* V_{n+3}*, where* V_{n+3} *has lengths* $\{N_1, N_2, ...N_{n-1}, 2,2,2,2\}$ *in the respective dimensions.*

Proof. We first describe the wrapping process, then argue its correctness.

In essence, the Wrap process embeds the n-D General Octahedral Tucker instance into an $(n+3)$-D General Octahedral Tucker instance, by folding a length 8 side along 4 orthogonal sides of a length-2 4-D hyper grid. To formally specify the embedding, note that vertices in these Tucker instances have the same coordinates in almost all dimensions. The difference is in the coordinate of one dimension (which is being folded), which has length 8 in the input instance and

[6]Actually, the order of dimensions doesn't influence the correctness of *Fold* Lemma.

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*. Table 2 in appendix F 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 $\{-4,-3,..,4\}$). It is easy to verify that this is a bijective map. Figure 5b is a visual representation of the map.

To complete the wrap process, the coloring function of the new instance remains to be specified. To gain insight into the designing of the coloring function, and help visualise the wrap process, we first illustrate a similar wrapping of a side of length 6 into three sides of length 2 each, in Figure 5a. A 4-to-2 length reduction can also be done similarly: we have one new vertex added, the origin, which gets assigned one of the new complementary color pair, now available due to the added dimension. Wrapping a side of length 8 is an extension of this wrap process into one more dimension.

While extending the reduction idea from 4-to-2 to 6-to-2 is natural, extending to 8-to-2 requires careful assignment of the new colors. Intuitively, the 4 differentiating dimensions together can be thought to form a 4-D hyper grid. Classifying all vertices of this hyper grid on the basis of their first coordinate value, we divide the hyper grid into 3 cubes $\langle -1,*,*,*\rangle$, $\langle 0,*,*,*\rangle$ and $\langle 1,*,*,*\rangle$. A valid coloring function that assigns colors without violating Tucker lemma conditions is shown in Figure 5b.[7] Here, vertices mapped from the old n-D General Octahedral Tucker instance are assigned the same colors in the new instance. Those diametrically opposite to these vertices are assigned their complementary colors. The new vertices of the $(n+3)$-D instance are colored using the 3 new color pairs available, as shown in Figure 5b. The coloring function is formally specified in Table 5 of appendix F.

This coloring function, by definition, is valid. To prove correctness, we need to prove this coloring scheme retains all complementary edges of the n-D Tucker instance, and does not add new ones. We prove three assertions to do so:

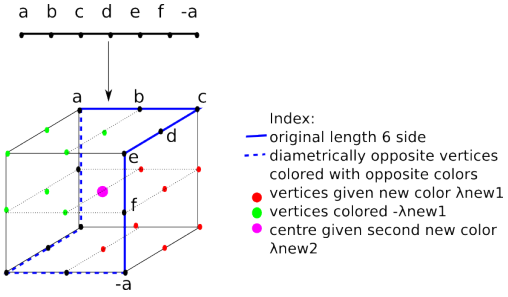
1. Edges of the form (u,v), where u is a vertex mapped from the n-D General Octahedral Tucker instance, and v is not, are not complementary
2. Vertices colored with new colors $\{\pm(n+1),\pm(n+2),n+3\}$ do not form complementary edges (i.e., no new solutions are formed)[8]
3. The adjacencies of vertices embedded from the n-D General Octahedral Tucker instance are retained (i.e., all old solutions are retained and no new extra solutions 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 can never be complementary. The second and third statements are proved separately as Lemma 14 and Lemma 15, thus completing the reduction. □

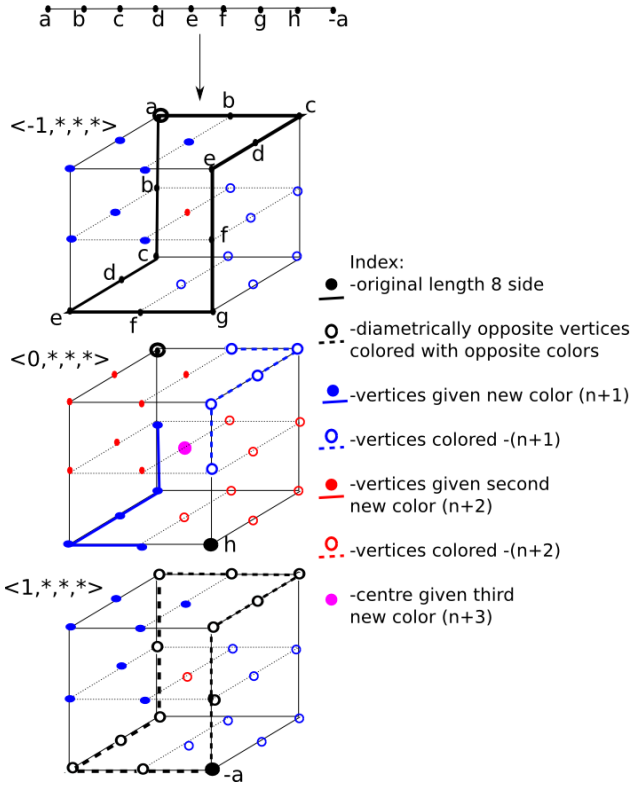
Lemma 14. *While wrapping a dimension of length* 8 *of a* General Octahedral Tucker *instance as described in lemma 13, vertices colored with new colors* $\{\pm(n+1),\pm(n+2),n+3\}$ *do not form complementary edges.*

[7] To avoid working in 4-D, its natural to think of the following alternate ideas: (1) Reduce side lengths from 8 to 4, then to 2, both in 2-D. (2) Reduce side lengths from 8 to 6, then to 2, which keeps the reduction in 3-D space. However, these ideas are not achievable using the current folding techniques: *Fold* lemma allows folding only until length 8, and Wrap lemma requires a separating boundary between the new vertices added. Wrapping a length 8 side along a side-4 square adds all new vertices in the interior of the new structure.

[8] Note that $-(n+3)$ color is not used in the reduction anywhere. There may exist another wrap scheme that can fully use the colors and reduce to lower-dimensional Octahedral Tucker. However, "tightness" is not very crucial to this work and we leave it to future work.



(a) Folding a dimension of length 6 into three dimensions of length 2 each.



(b) Coloring scheme of new instance after folding one side of length 8 into 4 sides of length 2 each

Figure 5: Wrapping constant length sides into length 2 hyper grids

Lemma 15. *While wrapping a dimension of length* 8 *of a* General Octahedral Tucker *instance as described in lemma 13, the adjacencies of vertices embedded from the* n*-D Tucker instance are retained.*

Combined with the above two lemmas (the detailed proof are shown in appendix E), the Wrap lemma naturally leads to the following reduction:

Lemma 16. *An* n*-D* General Octahedral Tucker *instance* (G, T_n) *defined on* V_n *that has length* 8 *in all dimensions, can be reduced to an* $O(n)$*-D* Octahedral Tucker *instance.*

Proof. The Wrap lemma, while reducing one dimension, does not constrain the lengths of other dimensions. Each length 8 dimension can thus be independently wrapped by applying Wrap lemma, resulting in a General Octahedral Tucker of higher dimension. Finally, the process ends in a General Octahedral Tucker instance of length 2 in all dimensions, which by definition is an Octahedral Tucker instance. □

The picture is now complete. Theorems 9 and 12, and lemma 16 together prove n-D Octahedral Tucker PPA – Hard. Along with the membership proof of theorem 7, this establishes:

Theorem 17. n*-D* Octahedral Tucker *is* PPA – Complete.

4 Remarks and Discussion

In this paper, we resolve a decade old open problem by proving Octahedral Tucker PPA – Complete. The statement of the lemma from which it arises is more than 70 years old, thus the problem has

numerous applications. Additionally, as an interesting side note, Theorem 12 also proves higher dimensional General Octahedral Tucker (under our defined *General Octahedral Tucker Triangulation*) of constant lengths $\mathsf{PPA} - \mathsf{Hard}$.

In further work, the next problems to analyze are the Ham Sandwich and Kneser Lovasz theorems. These problems are proved using Octahedral Tucker [13, 14], and this result, along with that of [2], places them in PPA. We further ask if they are complete for the class too. There are several other interesting problems in PPA, notably the Integer factoring problem, and the classical SMITH problem (finding a second hamiltonian circuit in a 3-regular graph). We would like to know if answers to the previous questions help resolve these. The final goal of resolving the computational complexity of these specific problems would be to find a relation between PPA and PPAD.[9]

References

[1] Bharat Adsul, Jugal Garg, Ruta Mehta, and Milind Sohoni. 2011. Rank-1 Bimatrix Games: A Homeomorphism and a Polynomial Time Algorithm. *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing (STOC'11)* (2011), 195–204.

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

[3] Aleksandrs Belovs, Gábor Ivanyos, Youming Qiao, Miklos Santha, and Siyi Yang. 2017. On the Polynomial Parity Argument Complexity of the Combinatorial Nullstellensatz. *32nd Computational Complexity Conference (CCC 2017)* 79 (2017), 30:1–30:24.

[4] N. Bitansky, O. Paneth, and A. Rosen. 2015. On the Cryptographic Hardness of Finding a Nash Equilibrium. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS'15)*. 1480–1498.

[5] Xi Chen, Decheng Dai, Ye Du, and Shang-Hua Teng. 2009. Settling the Complexity of Arrow-Debreu Equilibria in Markets with Additively Separable Utilities. *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS '09)* (2009), 273–282.

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

[7] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. 2009. The Complexity of Computing a Nash Equilibrium. *SIAM J. Comput.* 39, 1 (2009), 195–259.

[8] Xiaotie Deng, Jack R. Edmonds, Zhe Feng, Zhengyang Liu, Qi Qi, and Zeying Xu. 2016. Understanding PPA-Completeness. *31st Conference on Computational Complexity (CCC 2016)* 50 (2016), 23:1–23:25.

[9] Aris Filos-Ratsikas and Paul W. Goldberg. To appear. Consensus Halving is PPA-Complete. *Proceedings of the Fiftieth Annual ACM Symposium on Theory of Computing (STOC'18)* (To appear).

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

[9] As of now, we only know the trivial relation $\mathsf{PPAD} \subseteq \mathsf{PPA}$.

[11] Katalin Friedl, Gábor Ivanyos, Miklos Santha, and Yves F. Verhoeven. 2006. Locally 2-Dimensional Sperner Problems Complete for the Polynomial Parity Argument Classes. *Algorithms and Complexity: 6th Italian Conference, CIAC 2006, Rome, Italy, May 29-31, 2006. Proceedings* (2006), 380–391.

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

[13] Jesús A. De Loera, Xavier Goaoc, Frédéric Meunier, and Nabil H. Mustafa. 2017. The discrete yet ubiquitous theorems of Carathéodory, Helly, Sperner, Tucker, and Tverberg. *CoRR* abs/1706.05975 (2017).

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

[15] Ruta Mehta. 2014. Constant Rank Bimatrix Games Are PPAD-hard. *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing (STOC'14)* (2014), 545–554.

[16] Dömötör Pálvölgyi. 2009. 2D-TUCKER Is PPAD-Complete. *Internet and Network Economics: 5th International Workshop, WINE 2009, Rome, Italy, December 14-18, 2009. Proceedings* (2009), 569–574.

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

[18] Aviad Rubinstein. 2017. Settling the Complexity of Computing Approximate Two-player Nash Equilibria. *SIGecom Exchanges.* 15, 2 (Feb. 2017), 45–49.

[19] Forest W. Simmons and Francis Edward Su. 2003. Consensus-halving via theorems of Borsuk-Ulam and Tucker. *Mathematical Social Sciences* 45, 1 (2003), 15 – 25.

[20] Albert W. Tucker. 1945. *Some topological properties of disk and sphere.*

A Octahedral Tucker is in PPA: A Complete Proof

Proof of Theorem 7. Octahedral Tucker is a special case of the General Octahedral Tucker problem, which itself is a special case of Tucker. As Tucker is in PPA [17, 2], the theorem follows. Nevertheless, for completeness, we present the proof here. The proof is based on Papadimitriou's proof [17], which in turn refers to ideas of Todd and Freund [10].

AEUL is a natural $\mathsf{PPA} - \mathsf{Complete}$ computation problem based on the definition of PPA, defined as follows:

Definition 18 (AEUL [8])**.** *Suppose an input circuit L_n of size polynomial in n accepts inputs u from 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 $v, w \in C_n \setminus \{u\}$. A pair $u, v \in C_n$ is called valid, if $v \in L_n(u) \Leftrightarrow u \in L_n(v)$. $\mathbf{0}^n$ is an input known to have $|L_n(\mathbf{0}^n)| = 1$. The search problem is to find another configuration v, $v \neq \mathbf{0}^n$ such that $|L_n(v)| = 1$, or find an invalid pair.*

To visualize, AEUL defines a graph of maximum degree at most two, with one single degree vertex (0^n) specified. It then asks to compute another odd degree vertex.

To prove its membership 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 hyper grid, 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 $\boldsymbol{Z_n} := \{1, 2, \cdots, n\}$ be indices given to n dimensions and $Z \subseteq Z_n$. We define $\boldsymbol{T_Z} := \{\mathbf{t} \in \mathbb{R}^{|Z|} : t_i \in \{-1, 0, 1\} \forall i \in Z\}$ as the hyper grid of $3^{|Z|}$ vertices containing the origin $0^{|Q|}$ in $|Z|$-dimensional space, where the number of choices for possible lengths in each dimension is 2. We denote $\boldsymbol{T} = T_{Z_n}$, the n-dimension hyper grid of length 2 that has the vertex set $\{-1, 0, 1\}^n$. To generalize the definition of T_Z to orthants of $\mathbb{R}^n$, we define $\boldsymbol{Q} \subseteq \{\pm 1, \pm 2, \cdots, \pm n\}$, a subset of the standard set of axes of $\mathbb{R}^n$. Q is called a **d-orthant index set** if $|Q| = d$ and $\forall i \in Z_n,\ |\{i, -i\} \cap Q| \leq 1$.

We define T_Q, termed **Q-orthant**, as the set of all vertices $t \in \mathbb{R}^n$ such that:

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

Note that the Octahedral triangulation of T_{Z_n} induces a (possibly lower dimensional) triangulation on every $Q - orthant\ T_Q$. Also, T_Q and T_{Z_n} have one difference: no boundary face of T_{Z_n} contains the origin, but half of the boundary faces of T_Q do. We denote the boundaries of T_Q coincident to T_{Z_n} its *external boundaries*, and the rest its *internal boundaries*.

Finally, let $\mathbf{g}$ be the given coloring function on T. We know that g is valid: $\forall \mathbf{x} \in T \backslash \{\mathbf{0^n}\}\ \mathbf{g}(\mathbf{x}) = -\mathbf{g}(-\mathbf{x})$. Without of loss of generality, we assume the color of the origin $\mathbf{0}^n$ to be 1: $g(\mathbf{0}^n) = 1$.

We now describe the reduction of Octahedral Tucker to AEUL: To define the nodes for the graph in the AEUL structure, we introduce the concept of admissible simplices of the Octahedral triangulated grid T.

Definition 19 (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 is located in the space indexed by its colors $T_{Q(S)}$.*

Note an admissible $(d-1)$-simplex S contains d vertices, at least $(d-1)$ of which are assigned different colors. 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. □

B Proof of Theorem 9: 2-D General Octahedral Tucker is PPA − Hard

Proof Sketch. [2] 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 blocks of vertices in 2-D Tucker, with one entrance and one exit possible in each block (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 AEUL 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 remaining 2-D Tucker vertices (called the 'environment') are not mapped to any AEUL vertex/edge, and 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 2-D Tucker solution corresponds to a solution of the AEUL problem, and given the 2-D Tucker solution, the solution for AEUL can be found in polynomial time.

We modify the proof of [2], to reduce General Octahedral Tucker to AEUL. The reason why the exact proof does not apply here is the size of grids of the Tucker instances: 2-D General Octahedral Tucker has **GOTT**, and has size $4p \times 4q$ **cells**, for $p, q \in \mathbb{N}$. [2] reduces AEUL to a 2-D Tucker instance of size $m \times m$ **lines**[10] where $m = 4 \times 13 \times |G|$. That is, the size of the 2-D Tucker grid is odd, and General Octahedral Tucker is defined only on instances with even lengths in all dimensions[11]. We modify their proof as follows:

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 block of size 20×20 cells (or 21×21 lines), instead of the original block size of 13×13 lines

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

[11] This is because of Octahedral Triangulation's asymmetry at odd and even coordinate vertices. This asymmetry is explained more in Appendix C

That is, we follow the entire reduction procedure by mapping an AEUL instance to a 2-D General Octahedral Tucker instance of size $m \times m$ where $m = 4 \times 21 \times |G|$ lines. We map each AEUL vertex to a block of size 20×20 cells, and edges to 4-wide tubes connecting the corresponding blocks in the General Octahedral Tucker grid. One cell in our reduction is illustrated in figure 6.

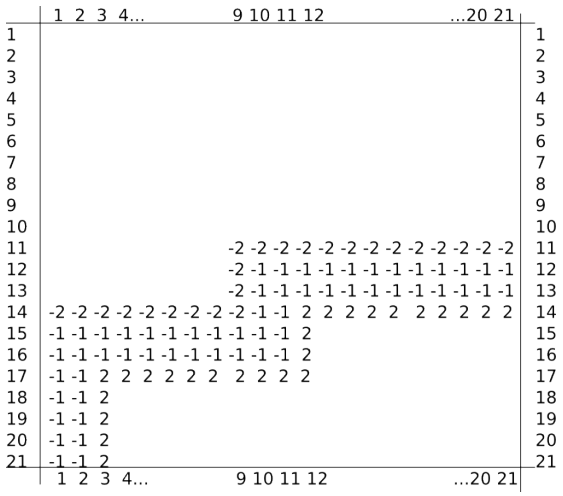


Figure 6: Reducing 2-D General Octahedral Tucker to AEUL: the block assigned to every AEUL vertex in General Octahedral Tucker. This is a modified version of the block used in the reduction of [2]. The difference is the size, and is illustrated to show the proof remains the same when applied to these blocks.

Our General Octahedral Tucker instance has **GOTT**. For an open end in a block 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 General Octahedral Tucker solutions generated for each AEUL solution, making analysis easy.

Each block 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 block. We need to add 4 cells each at the top and bottom of the center tube to ensure all crossings in General Octahedral Tucker too get resolved in one block. This increases the grid side length to $8 + 4 + 8 = 20$ cells, or 21 lines.

This modified reduction scheme reduces AEUL to 2-D General Octahedral Tucker, thus proving 2-D General Octahedral Tucker $\mathsf{PPA} - \mathsf{Hard}$. □

C Proof of Lemma 11

General Octahedral Tucker instances on which Fold lemma is applied are required to have lengths in all dimensions powers of 2. Thus the input General Octahedral Tucker instance, the starting step of the reduction process, should satisfy this condition. In this section, we prove: every 2-D General Octahedral Tucker instance can be converted into another instance where lengths of both dimensions are powers of 2.

Proof. Denote the lengths of the two dimensions of (G, T_2) on V_2 by N_1, N_2, where N_1 and N_2 are

both multiples of 4. Let $m, n \in \mathbb{Z}^+$ s.t. $2^{m-1} < N_1 \leq 2^m$ and $2^{n-1} < N_2 \leq 2^n$. The reduction simply pads extra vertices along all four boundaries to increase the lengths to the nearest powers of two. Along every boundary, a layer of vertices, repeating the color assignment of that boundary, are added. Then extra vertices are filled on all corners to make the new structure rectangular. These vertices are assigned the same color of the nearest corner vertex of V_2. Formally, we list the construction below. The reduction is also illustrated in Figure 7. The new hyper grid V_2' is divided into 9 parts, and the mapping of vertices from V_2 into V_2', along with the color assignment of vertices in each part are specified separately:

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 $p_2 = -N_2/2$.

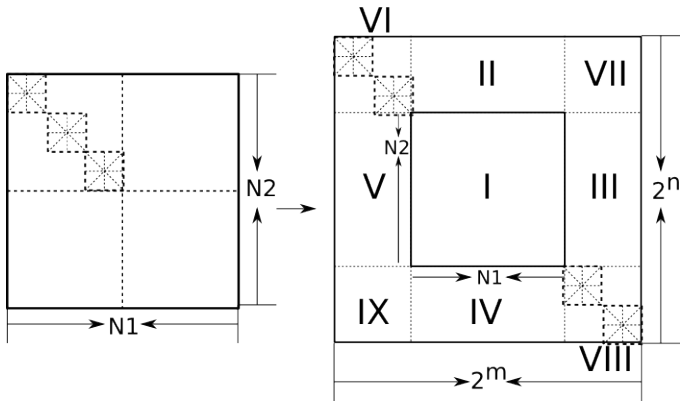


Figure 7: Converting a 2-D General Octahedral Tucker instance into one whose sides have length $\{2^m, 2^n\}$, where $m, n \in \mathbb{N}$.

For the correctness of the reduction, first, observe that there is a complementary edge existing in part I if and only if it is a complementary edge in original General Octahedral Tucker (G, T_2) defined on 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 General Octahedral Tucker (G, T_2), on the boundary of V_2 and we can find it in polynomial time. We summarize in Table 1, the corresponding complementary

edge $\{\mathbf{p}, \mathbf{q}\}$ in (G, T_2) of V_2, for each complementary edge $\{\mathbf{p}', \mathbf{q}'\}$ in (G', T_2') on V_2' where $p_1' \leq q_1'$.

Given Complementary edge in (G', T_2') defined on V_2		Corresponding Complementary edge in (G, T_2) defined on V_2	
$\mathbf{p}'$	$\mathbf{q}'$	$\mathbf{p}$	$\mathbf{q}$
$-\frac{N_1}{2} \leq p_1' \leq \frac{N_1}{2}$, $-\frac{N_2}{2} \leq p_2' \leq \frac{N_2}{2}$	$-\frac{N_1}{2} \leq q_1' \leq \frac{N_1}{2}$, $-\frac{N_2}{2} \leq q_2' \leq \frac{N_2}{2}$	$p_1 = p_1'$ $p_2 = p_2'$	$q_1 = q_1'$ $q_2 = q_2'$
$-\frac{N_1}{2} \leq p_1' \leq \frac{N_1}{2}$, $\frac{N_2}{2} < p_2' \leq 2^{n-1}$	$-\frac{N_1}{2} \leq q_1' \leq \frac{N_1}{2}$ $\frac{N_2}{2} < q_2' \leq 2^{n-1}$	$p_1 = p_1'$ $q_1 = \frac{N_1}{2}$	$q_1 = q_1' = p_1' + 1$ $q_2 = \frac{N_1}{2}$
$\frac{N_1}{2} < p_1' \leq 2^{m-1}$ $-\frac{N_2}{2} < p_2' < \frac{N_2}{2}$	$\frac{N_1}{2} < q_1' \leq 2^{m-1}$ $-\frac{N_2}{2} < q_2' < \frac{N_2}{2}$	$p_1 = \frac{N_1}{2}$ $p_2 = p_2'$	$q_1 = \frac{N_1}{2}$ $q_2 = q_2'$
$-\frac{N_1}{2} \leq p_1' \leq \frac{N_1}{2}$, $-2^{n-1} \leq p_2' < -\frac{N_2}{2}$	$-\frac{N_1}{2} \leq q_1' \leq \frac{N_1}{2}$ $-2^{n-1} \leq q_2' < -\frac{N_2}{2}$	$p_1 = p_1'$ $p_2 = -\frac{N_2}{2}$	$q_1 = q_1' = p_1' + 1$ $q_2 = -\frac{N_2}{2}$
$-2^{m-1} \leq p_1' < -\frac{N_1}{2}$ $-\frac{N_2}{2} < p_2' < \frac{N_2}{2}$	$-2^{m-1} \leq q_1' < -\frac{N_1}{2}$ $-\frac{N_2}{2} < q_2' < \frac{N_2}{2}$	$p_1 = -\frac{N_1}{2}$ $p_2 = p_2'$	$q_1 = -\frac{N_1}{2}$ $q_2 = q_2'$

Table 1: Finding the corresponding complementary edge $\{\mathbf{p}, \mathbf{q}\}$ in (G, T_2) defined on V_2, given a complementary edge $\{\mathbf{p}', \mathbf{q}'\}$ in (G', T_2') defined on V_2'.

As we can find a complementary edge in the original 2-D General Octahedral Tucker (G, T_2) defined on V_2 in polynomial time, given a complementary edge in (G', T_2') defined on V_2', the proof describes a valid reduction as asserted by the theorem. □

D Proof of Lemma 10 and Theorem 12

On a high level, we fold a $(n-1)$-D General Octahedral Tucker to n-D General Octahedral Tucker by reducing the half length of one dimension and add one extra dimension of constant length 8. With the help of **GOTT** and folding technique, we could show the correctness of this folding scheme: (i) we maintain the original triangulation of $(n-1)$-D General Octahedral Tucker when we embed it in a n-D General Octahedral Tucker instance. (ii) In the new n-D General Octahedral Tucker, we don't import any new complementary other than the original complementary edge in $(n-1)$-D General Octahedral Tucker.

Proof of Lemma 10. The proof structure is divided to the following two stages.

1. We first show how to fold $(n-1)$-D General Octahedral Tucker to n-D General Octahedral Tucker,
2. Then we prove the correctness of the reduction.

First part of the proof. The goal is to shrink $(n-1)$th dimension into half at the cost of appending another new dimension (nth dimension) of small length (turns out, 8 is the smallest possible length) simultaneously. The reduction embeds the input hyper grid V_{n-1} into V_n, such that the triangulations of both instances T_{n-1} and T_n agree on the embedded vertices. Figure 4

illustrates this for an embedding of V_2 into V_3. Intuitively, as figure 3 shows, we fold V_{n-1} along one side twice in a snake-like fashion, add one layer of extra vertices between the folds to insulate original grid vertices from becoming adjacent to each other, and pad the outer layers to keep the vertices in the interior of the new instance. This results in length 8 in the new dimension added. We now formalize the embedding of the hyper grid V_{n-1} of $(n-1)$-D General Octahedral Tucker into V_n, while maintaining all adjacencies between vertices of V_{n-1} as per T_{n-1}, and its coloring function:

Embedding

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

Now we show how to color the vertices of V_n such that the defined coloring function on V_n is valid. The coloring function on V_n, because of the added dimension, has one extra pair of colors $\{\pm n\}$ available.

Adding Colors Based on the embedding strategy above, $\forall$ $\mathbf{p}$ in V_{n-1}, there is a corresponding $\mathbf{p'}$ in V_n, which we assign the same color in V_n as was asisgned in V_{n-1}. We define the set of these corresponding vertices in V_n as S. For coloring vertices in $V_n \backslash S$, we observe that S divides V_n into two parts. One part is above S where we color all vertices with the new color n, while the other part is below S where we color them with $-n$.

Formally, we define the coloring function for $V_n \backslash S$ as follows:

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

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

Second part of the proof. We now prove the correctness of this reduction. First, we can easily check that (G', T_n) on V_n satisfies the *valid property* (defined in Lemma 2), i.e, $\forall \mathbf{q} \in \text{Boundary}(V_n)$, $g'(-\mathbf{q}) = -g'(\mathbf{q})$. Second, we show there is no other *new* complementary edge which is not in

original $(n-1)$-D General Octahedral Tucker (G, T_{n-1}) on V_{n-1}. This is because under **GOTT**: (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 General Octahedral Tucker.

- (a) is obviously correct since there are no $\pm n$ in S.
- 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 hyper grid. Thus, we can't link $\mathbf{p}, \mathbf{q}$ together in V_n under **GOTT**.
- For (c), we show **GOTT** will guarantee this claim. We only need to prove the **GOTT** in V_n maintains the original **GOTT** in V_{n-1} for S. For any Octahedral Hypergrid (Definition 8) $H_{\mathbf{p}}$ 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 **GOTT**, $H'_{\mathbf{p}}$ is triangulated by **SOTT**. 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 *valid property*, we have thus proved the correctness of this reduction. □

One natural thought to optimize the lemma is to add just one layer of vertices instead of two along the boundaries, resulting in the length of new dimension 6 instead of 8. The argument for why this cannot be done is subtle. Our definition of GOTT has the same triangulation for unit hyper grids centered around vertices of odd coordinates. In the reduction, every General Octahedral Tucker instance has the origin at the center of the hyper grid, and even lengths in each dimension. Thus, when reducing $(n-1)$-D General Octahedral Tucker into n-D General Octahedral Tucker, to retain the triangulation of V_{n-1}, the first vertex (vertex of lowest all-odd coordinates) of V_{n-1} must have all odd coordinates in V_n too. Thus, to keep interior vertices inside again, the *Fold* has to add an even number of vertices along boundaries. This results in the added layer's length along the new dimension being at least 2 on both sides. With one extra layer of vertices between folds, the net length of the new dimension thus can only be at least 8. For similar reasons, it is also interesting to note that the octahedral triangulation allows folds only after even length intervals, else as Figure 8 shows, we do not retain the triangulation, and add extra adjacencies and delete existing ones.

E Proof of Lemma 14 and Lemma 15

We prove two claims to complete the proof of correctness of the Wrap lemma in this section. The coloring function of the new General Octahedral Tucker instance formed by applying the Wrap lemma on an n-D General Octahedral Tucker instance, was described by the set of differentiating dimension coordinates of each vertex. The differentiating dimensions together are thought to form a 4-D hyper grid, further divided into 3 3-D cubes based on the coordinate value of the first dimension in this set.

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 of vertices belonging to each set, denoted by $\langle s_1, s_2, s_3 \rangle$ and $\langle e_1, e_2, e_3 \rangle$ respectively, are enumerated in Tables 3 and 4 respectively. Figure 9 marks these sets in a cube.

Proof of Lemma 14. We make the following observations on the new instance from Figure 5b:

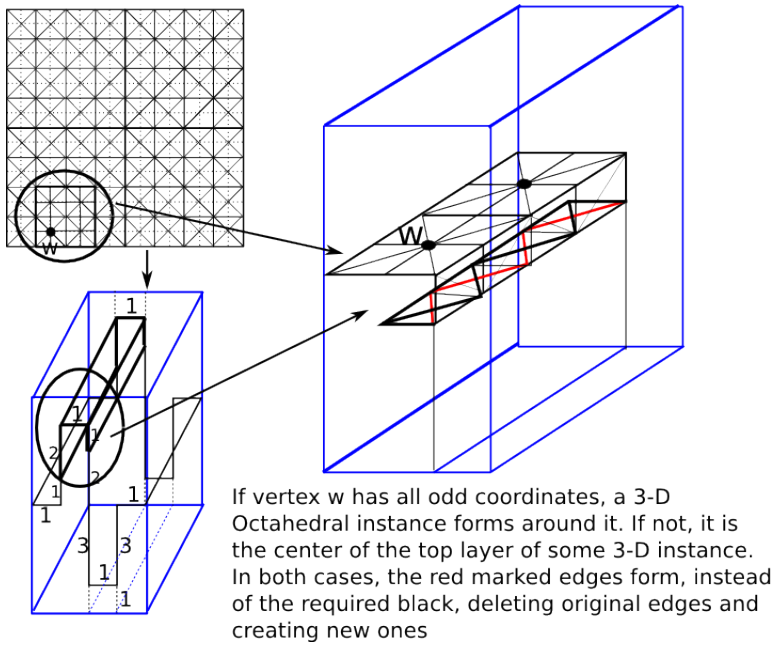


Figure 8: Attempt to fold 2-D 16×16 General Octahedral Tucker into $16 \times 8 \times 4$ 3-D General Octahedral Tucker by folding after odd lengths.

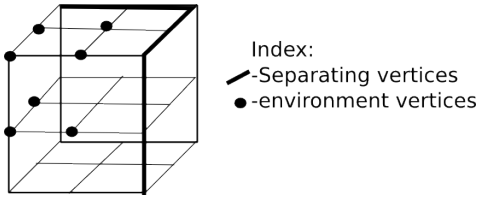


Figure 9: Separating and Environment vertex sets in a 3-D cube

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

$$\langle -1, -1/0, 0/1, -1/0 \rangle \backslash \langle -1, 0, 0, 0 \rangle$$

   Vertices belonging to the separating vertex set of the middle cube are of the form $\langle \mathbf{0}, -1, \mathbf{-1}, * \rangle$ or $\langle \mathbf{0}, \mathbf{-1}, *, 1 \rangle$ or $\langle \mathbf{0}, 0, 1, \mathbf{1} \rangle$. 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 $\langle -1, 0, 0, 0\rangle$. 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 hyper grid 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 $\pm(n+3)$ □

Proof of Lemma 15. To prove this lemma, we need to prove all edges that existed in the n-D General Octahedral Tucker instance exist in the $(n+3)$-D General Octahedral Tucker 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 13, 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 13 (For instance, if the coordinate of u was -3 in the n-D graph, its coordinates in the changed and new dimensions are $\langle -1, -1, -1, 0\rangle$ (and $\langle -1, 0, -1, -1\rangle$). 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 $\langle -1, -1, -1, -1\rangle$, $\langle -1, -1, -1, 0\rangle$ or $\langle -1, -1, -1, 1\rangle$(and $\langle -1, 0, -1, -1\rangle$ or $\langle -1, 1, -1, -1\rangle$) 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 13, 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 Octahedral Tucker conditions, implying these sets of coordinates of the vertices u and v are not adjacent in the $(n+3)$-D instance too. □

F Coloring Scheme used in reduction of lemma 13

Lemma 13 described a reduction process to fold a length 8 side in n-D General Octahedral Tucker to 4 sides of length 2 in an $(n+3)$-D General Octahedral Tucker instance. We

$\langle \mathbf{k} \rangle$	$\langle \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \rangle$
$\langle -4 \rangle$	$\langle -1, -1, -1, -1 \rangle$
$\langle -3 \rangle$	$\langle -1, -1, -1, 0 \rangle$and$\langle -1, 0, -1, -1 \rangle$
$\langle -2 \rangle$	$\langle -1, -1, -1, 1 \rangle$and$\langle -1, 1, -1, -1 \rangle$
$\langle -1 \rangle$	$\langle -1, -1, 0, 1 \rangle$and$\langle -1, 1, 0, -1 \rangle$
$\langle 0 \rangle$	$\langle -1, -1, 1, 1 \rangle$and$\langle -1, 1, 1, -1 \rangle$
$\langle 1 \rangle$	$\langle -1, 0, 1, 1 \rangle$and$\langle -1, 1, 1, 0 \rangle$
$\langle 2 \rangle$	$\langle -1, 1, 1, 1 \rangle$
$\langle 3 \rangle$	$\langle 0, 1, 1, 1 \rangle$

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

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

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

formally enumerate the coordinates of old vertices from n-D instance in the $(n+3)$-D instance, and the coloring scheme of the new $(n+3)$-D instance in Tables 2 and 5 respectively below. We also enumerate the coordinate values of 'separating vertices' and 'environment vertices', which are definitions introduced to ease exposition, in Tables 3 and 4 respectively.

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

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

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

Table 5: Coloring function of the General Octahedral Tucker instance reduced from a lower dimension instance.$\langle \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \rangle$ 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. $\langle \mathbf{k} \rangle$ 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 $\{-1, 0, 1\}$.

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il