The Complexity of Splitting Necklaces and Bisecting Ham Sandwiches*

Aris Filos-Ratsikas
École Polytechnique Fédérale de Lausanne
Lausanne, Switzerland
aris.filosratsikas@epfl.ch

Paul W. Goldberg
University of Oxford
Oxford, United Kingdom
Paul.Goldberg@cs.ox.ac.uk

ABSTRACT

We resolve the computational complexity of two problems known as Necklace-splitting and Discrete Ham Sandwich showing that they are PPA-complete. For Necklace-splitting, this result is specific to the important special case in which two thieves share the necklace. We do this via a PPA-completeness result for an approximate version of the Consensus-halving problem, strengthening our recent result that the problem is PPA-complete for inverse-exponential precision. At the heart of our construction is a smooth embedding of the high-dimensional Möbius strip in the Consensus-halving problem. These results settle the status of PPA as a class that captures the complexity of "natural" problems whose definitions do not incorporate a circuit.

CCS CONCEPTS

• **Theory of computation → Complexity classes**; **Problems, reductions and completeness**; *Algorithmic game theory*.

KEYWORDS

PPA-Completeness, Necklace Splitting, Ham Sandwich, Consensus Halving

ACM Reference Format:
Aris Filos-Ratsikas and Paul W. Goldberg. 2019. The Complexity of Splitting Necklaces and Bisecting Ham Sandwiches. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC '19), June 23–26, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3313276.3316334

1 INTRODUCTION

The complexity classes PPA and PPAD were introduced in a seminal paper of Papadimitriou [48] in 1994, in an attempt to classify several natural problems in the class TFNP [46]. TFNP is the class of *total search problems* in NP for which a solution exists for every instance, and solutions can be efficiently verified. Various important problems were subsequently proven to be complete for the class PPAD, such as the complexity of many versions of Nash equilibrium [13, 14, 17, 22, 47, 50, 51], market equilibrium computation [11, 15, 16, 52, 55], and others [20, 34]. As evidence of computational hardness, PPA-completeness is stronger than PPAD-completeness, i.e., PPAD $\subseteq$ PPA. Indeed, Jeřábek [32] shows that it indicates cryptographic hardness in a strong sense: [32] gives a randomised reduction from FACTORING to PPA-complete problems. This is not known for PPAD-complete problems. For more details, and the significance of PPA-completeness, we refer the reader to the related discussion in [24]. PPA is the class of problems reducible to Leaf (Definition 1), and a PPA-complete problem is polynomial-time equivalent to Leaf.

Definition 1. *An instance of the problem* Leaf *consists of an undirected graph G whose vertices have degree at most 2; G has 2^n vertices represented by bitstrings of length n; G is presented concisely via a circuit that takes as input a vertex and outputs its neighbour(s). We stipulate that vertex 0^n has degree 1. The challenge is to find some other vertex having degree 1.*

Complete problems for the class PPA seemed to be much more elusive than PPAD-complete ones, especially when one is interested in "natural" problems, where "natural" here has the very specific meaning of problems that do not explicitly contain a circuit in their definition. Besides Papadimitriou [48], other papers asking about the possible existence of natural PPA-complete problems include [12, 17, 18, 29]. In a recent precursor [24] to the present paper we identified the first example of such a problem, namely the approximate Consensus-halving problem, dispelling the suspicion that such problems might not exist. In this paper we build on that result and settle the complexity of two natural and important problems whose complexity status were raised explicitly as open problems in Papadimitriou's paper itself, and in many other papers beginning in the 1980s. Specifically, we prove that Necklace-splitting (with two thieves, see Definition 2) and Discrete Ham Sandwich are both PPA-complete.

Definition 2 (Necklace Splitting). *In k-*Necklace-splitting *there is an open necklace with ka_i beads of colour i, for $1 \leq i \leq n$. An "open necklace" means that the beads form a string, not a cycle. The task is to cut the necklace in $(k-1) \cdot n$ places and partition the resulting substrings into k collections, each containing precisely a_i beads of colour i, $1 \leq i \leq n$.*

In Definition 2, k is thought of as the number of thieves who desire to split the necklace in such a way that the beads of each colour are equally shared. In this paper, usually we have $k = 2$ and we refer to this special case as Necklace-splitting.

*This work was supported by the Swiss National Science Foundation under contract No. 200021 165522 and by the ERC Advanced Grant 321171 (ALGAME). The authors would like to thank Alexandros Hollender for detailed and insightful proof-reading of earlier versions of this paper.

Definition 3 (Discrete Ham Sandwich). *In the* Discrete Ham Sandwich *problem, there are n sets of points in n dimensions having integer coordinates (equivalently one could use rationals). A solution consists of a hyperplane that splits each set of points into subsets of equal size (if any points lie on the plane, we are allowed to place them on either side, or even split them arbitrarily).*

In Definition 3, each point set represents an ingredient of the sandwich, which is to be cut by a hyperplane in such a way that all ingredients are equally split.

The *necklace-splitting problem* was introduced in a 1982 paper of Bhatt and Leiserson ([8], Section 5), where it arose in the context of VLSI circuit design (the version defined in [8] is the 2-thief case proved PPA-complete in the present paper). In 1985 and 1986, the 2-thief case was shown to have guaranteed solutions (as defined in Definition 2) by Goldberg and West [27] and Alon and West [5] and then in 1987, Alon [2] proved existence of solutions for k thieves as well. Early papers that explicitly raise its complexity-theoretic status as an open problem are Goldberg and West [27] and Alon [3, 4]. Subsequently, the necklace-splitting problem was found to be closely related to "paint-shop scheduling", a line of work in which several papers such as [42–44] explicitly mention the question of the computational complexity of necklace-splitting. Meunier [42] notes that the search for a *minimum* number of cuts admitting a fair division (which may be smaller than the number $(k-1)n$ that is guaranteed to suffice) is NP-hard, even for a subclass of instances of the 2-thief case. (That is a result of Bonsma et al. [10], for the "paint shop problem with words", equivalent to 2-thief Necklace-splitting with 2 beads of each colour).

In [24], we showed Necklace-splitting to be computationally equivalent to ε-Consensus-halving for inverse-polynomial precision parameter ε, but the PPA-completeness of ε-Consensus-halving was only shown for inverse-exponential ε. [24] established PPAD-hardness of Necklace-splitting, applying the main result of [23]. In this paper, we prove that ε-Consensus-halving is PPA-complete for ε inversely polynomial, thus obtaining the desired PPA-completeness of Necklace-splitting. While some structural parts of our reduction are extensions of those presented in [24], obtaining the result for inverse-polynomial precision is much more challenging, as the construction needs to move to a high-dimensional space (rather than the two-dimensional space which is sufficient for the result in [24]). We highlight the main new techniques that we have developed in this paper in Section 2.1, where we provide an overview of the reduction. Our PPA-completeness result gives a convincing negative answer to Meunier and Neveu's questions [43] about possible polynomial-time solvability or membership of PPAD for Necklace-splitting; likewise it runs counter to Alon's cautious optimism at ICM 1990 ([4], Section 4) that the problem may be solvable in polynomial time.

The *Ham Sandwich Theorem* [54] is of enduring and widespread interest due to its colourful and intuitive statement, and its relevance and applications in topology, social choice theory, and computational geometry. Roughly, it states that given d measures in Euclidean d-space, there exists a hyperplane that cuts them all simultaneously in half. Early work on variants and applications of the theorem focused on non-constructive existence proofs and mostly did not touch on the algorithmics. A 1983 paper by Hill [30] hints at possible interest in the corresponding computational challenge, in the context of a related land division problem. The computational problem (and its complexity) was first properly studied in a line of work in computational geometry beginning in the 1980s, for example [21, 36, 37, 40]. The problem envisages input data consisting of d sets of n points in Euclidean d-space, and asks for a hyperplane that splits all point sets in half. (The problem Discrete Ham Sandwich (Definition 3) as named in [48] is essentially this, with d set equal to n to emphasise that we care about the high-dimensional case). In this work in computational geometry, the emphasis has been on efficient algorithms for small values of d; Lo et al. [37] improve the dependence on d but it is still exponential, and the present paper shows for the first time that we should *not* expect to improve on that exponential dependence. More recently, Grandoni et al. [28] apply the "Generalized Ham Sandwich Theorem" to a problem in multi-objective optimisation and note that a constructive proof would allow a more efficient algorithm to emerge. The only computational hardness result we know of is Knauer et al. [35] who obtain a $W[1]$-hardness result for a constrained version of the problem; [35] points out the importance of the computational complexity of the general problem. The PPA-completeness result of the present paper is the first hardness result *of any kind* for Discrete Ham Sandwich, and as we noted, is a strong notion of computational hardness. Karpic and Saha [33] showing a form of equivalence between the Ham Sandwich Theorem and Borsuk-Ulam, explicitly mention the possible PPA-completeness of Discrete Ham Sandwich as an "interesting and challenging open problem".

We prove the PPA-completeness of Discrete Ham Sandwich via a simple reduction from Necklace-splitting. Ours is not the first paper to develop the close relationship between the two problems: Blagojević and Soberón [9] show a generalisation, where multiple agents may share a "sandwich", dividing it into convex pieces. Further papers to explicitly point out their computational complexity as open problems include Deng et al. [19] (mentioning that both problems "show promise to be complete for PPA"), Aisenberg et al. [1], and Belovs et al. [7].

2 PROBLEMS AND RESULTS

We present and discuss our main results, and in Section 2.1 we give an overview of the proof and new techniques, in particular with respect to the precursors [23, 24] to this paper.

Definition 4 (ε-Consensus Halving [24, 53]). *An instance I_{CH} incorporates, for $1 \leq i \leq n$, a non-negative measure μ_i of a finite line interval $A = [0, x]$, where each μ_i integrates to 1 and $x > 0$ is part of the input. We assume that μ_i are step functions represented in a standard way, in terms of the endpoints of intervals where μ_i is constant, and the value taken in each such interval. We use the bit model (logarithmic cost model) of numbers. I_{CH} specifies a value $\varepsilon \geq 0$ also using the bit model. We regard μ_i as the value function held by agent i for subintervals of A.*

A solution consists firstly of a set of n cut points *in A (also given in the bit model of numbers). These points partition A into (at most) $n+1$ subintervals, and the second element of a solution is that each subinterval is labelled A_+ or A_-. This labelling is a correct solution provided that for each i, $|\mu_i(A_+) - \mu_i(A_-)| \leq \varepsilon$, i.e. each agent has* *a value in the range* $[\frac{1}{2} - \frac{\varepsilon}{2}, \frac{1}{2} + \frac{\varepsilon}{2}]$ *for the subintervals labelled* A_+ *(hence also values the subintervals labelled* A_- *in that range).*

We assume without loss of generality that in a valid solution, labels A_+ and A_- alternate. We also assume that the alternating label sequence begins with label A_+ on the left-hand side of A (i.e. A_+ denotes the leftmost label in a Consensus-halving solution).

The Consensus-halving problem of Definition 4 is a computational version of the *Hobby-Rice theorem* [31]. Most of the present paper is devoted to proving the following theorem.

Theorem 1. ε-Consensus-halving *is* PPA*-complete for some inverse-polynomial* ε.

As mentioned in the introduction, in [24] it was proven that 2-thief Necklace-splitting and ε-Consensus-halving for ε being inversely-polynomial are computationally equivalent, i.e. they reduce to each other in polynomial time. Therefore, by [24] and the fact that the problem is in PPA (see the full version [25] for the proof), we immediately get the following corollary.

Theorem 2. Necklace-splitting *is* PPA*-complete when there are* $k = 2$ *thieves.*

If we knew that k-Necklace-splitting belonged to PPA for other values of k, we could of course make the blanket statement "Necklace-splitting is PPA-complete". Alas, the proofs showing that Necklace-splitting is a total search problem for $k > 2$ [2, 45] do *not* seem to boil down to the parity argument on an undirected graph! That being said, we do manage to establish membership of PPA for k being a power of 2 (essentially an insight of [2], see the Appendix of the full version [25], for the details and a related discussion). Of course, the $k = 2$ result strongly suggests that k-Necklace-splitting is a hard problem for other values of k.

As it happens, the PPA-completeness of Discrete Ham Sandwich follows straightforwardly, and we present that next. The basic idea of Theorem 3 of embedding the necklace in the moment curve appears already in [41, 49] and [38], p.48.

Theorem 3. Discrete Ham Sandwich *is* PPA*-complete.*

Proof. Inclusion in PPA is shown in the Appendix of the full version. For PPA-hardness, we obtain a reduction from 2-thief Necklace-splitting which is PPA-complete by Theorem 2.

The idea is to embed the necklace into the *moment curve* $\gamma = \{(\alpha, \alpha^2, \ldots, \alpha^n) : \alpha \in [0,1]\}$. Assume all beads lie in the unit interval $[0,1]$. A bead having colour $i \in [n]$ located at $\alpha \in [0,1]$ becomes a point mass of ingredient i of the ham sandwich located at $(\alpha, \alpha^2, \ldots, \alpha^n) \in \mathbb{R}^n$. It is known that any hyperplane intersects the moment curve γ in at most n points, (e.g. see [41], Lemma 5.4.2), therefore a solution to Discrete Ham Sandwich corresponds directly to a solution to Necklace-splitting, where the two thieves splitting the necklace take alternating pieces. (In the $k = 2$ case, we may assume without loss of generality that they do in fact take alternating pieces). □

A limitation to Theorem 3 is that the coordinates may be exponentially large numbers; they could not be written in unary. We leave it as an open problem whether a unary-coordinate version is also PPA-complete. As defined in [48], Discrete Ham Sandwich stipulated that each of the n sets of points is of size $2n$, whereas Definition 3 allows polynomial-sized sets. We can straightforwardly extend PPA-completeness to the version of [48] by adding "dummy dimensions" whose purpose is to allow larger sets of each ingredient; the new ingredients that are introduced, consist of compact clusters of point masses, each cluster in general position relative to the other clusters and the subspace of dimension n that contains the points of interest.

Notation: We use the standard notation $[n]$ for the set $\{1, \ldots, n\}$, and we also use $\pm[n]$ to denote $\{1, -1, 2, -2, \ldots, n, -n\}$. We often refer to elements of $\pm[n]$ as "labels" or "colours". λ is usually used to denote a labelling function (so its codomain is $\pm[n]$).

Let A denote the domain of an instance of Consensus-halving; if that has complexity n then A will be the interval $[0, poly(n)]$, where $poly(n)$ is some number bounded by a polynomial in n. Recall by Definition 4 that μ_a denotes the value function, or measure, of agent a on the domain A, in a Consensus-halving instance. We also associate each agent with its own cut (recall that the number of agents and cuts is supposed to be equal) and we let $c(a)$ denote the cut associated with agent a.

We let $p^C(n)$ be a polynomial that represents the number of "circuit-encoders" that we use in our reduction (see Section 5.1); we usually denote it p^C, dropping the n from the notation.

Finally, B denotes the n-cube (or "box") $[-1,1]^n$.

Terminology: In an instance of Consensus-halving, a *value-block* of an agent a denotes a sub-interval of the domain where a possesses positive value, uniformly distributed on that interval. In our construction, value-blocks tend to be scattered rather sparsely along the domain.

Remark: Some of the structural parts of the paper, including lemmas, theorems and their proofs are omitted from this version due to lack of space. We refer the reader to the full version of the paper [25] for the detailed exposition of the results.

2.1 Overview of the Proof

We review the ground covered by the precursors [23, 24] to this paper, then we give an overview of the technical innovations of the present paper.

2.1.1 Ideas From Previous Works. In [23] the PPAD-hardness of Consensus-halving was established. An arithmetic circuit can be encoded by an instance to ε-Consensus-halving, by letting each gate have a cut whose location (in a solution) represents the (approximate) value taken at that gate. Agents' valuation functions ensure that values taken at the gates behave according to the type of gate. A "PPAD circuit" can then be represented using an instance of Consensus-halving.

[24] noted that the search space of solutions to instances as constructed by [23], is oriented. A radical new idea was needed to encode the non-oriented feature of topological spaces representable by PPA. That was achieved by using two cuts to represent the coordinates of a point on a triangular region faving two sides identified to form a Möbius strip. (These cuts are the only ones that lie in a specific subinterval of the interval A of a Consensus-halving instance, called the "coordinate-encoding (c-e) region".

The two cuts are called the "coordinate-encoding cuts"). Identifying two sides in this way is done by exploiting the equivalence of taking a cut on the LHS of the c-e region, and moving it to the RHS. In order to embed a hard search problem into the surface of a standard 2-dimensional Möbius strip, it was necessary to work at exponentially-fine resolution, which immediately required inverse-exponential ε for instances of ε-Consensus-halving. In [24] we reduced from the PPA-complete problem 2D-Tucker [1], a search problem on an exponential-sized 2-dimensional grid.

In [24], the rest of A is called the "circuit-encoding region" R, and the cuts occurring within R do the job of performing computation on the location of cuts in the c-e region. The present paper retains this high-level structure (Section 4.1). As in [24] we use multiple copies of the circuit that performs the computation, each in its own subregion of R. Here we use $p^C(n)$ copies where p^C is a polynomial; in [24] we used 100 copies. Each copy is called a *circuit-encoder*. The purpose of multiple copies is to make the system robust; a small fraction of copies may be unreliable: as in [24] we have to account for the possibility that one of the c-e cuts may occur in the circuit-encoding region, rendering one of the copies unreliable. We re-use the "double negative lemma" of [24] that such a cut is not too harmful. We also adapt a result of [24] that when a cut is moved from the one end to the other end of the c-e region, this corresponds to identifying two facets of a simplex to form a Möbius strip.

[24] uses a sequence of "sensor agents" to identify the endpoints of intervals labelled A_+ and A_- in the coordinate-encoding region, and feed this information into the above mentioned circuit-encoders, which perform computation on those values. As in [24] we use sensor agents. We obtain a simplification with respect to [24] which is that we do not need the gadgets used there to perform "bit-extraction" (converting the position of a c-e cut into n boolean values). In [24], a solution to an instance of Consensus-halving was associated with a sequence of 100 points in the Möbius-simplex (referred there as the "simplex-domain"), and the "averaging manoeuvre" introduced in [17] was applied. In this paper, for a polynomial $p^C(n)$, we sample a sequence of p^C points in a more elegant manner, again exploiting the inverse-polynomial precision of solutions that we care about.

2.1.2 Technical Innovations. As in [24], we reduce from the PPA-complete problem 2D-Tucker [1]. That computational problem uses an exponentially-fine 2D grid, and (unlike [24]), in Section 3 we apply the *snake-embedding* technique invented in [13] (versions of which are used in [18, 19] in the context of PPA) to convert this to a grid of fixed resolution, at the expense of going from 2 to n dimensions. The new problem, Variant high-D Tucker (see the full version for the formal definition) envisages a $7 \times 7 \times \cdots \times 7$ grid. Here, we design the snake-embedding in such a way that PPA-completeness holds for instances of the high-dimensional problem that obey a further constraint on the way the high-dimensional grid is coloured, that we exploit subsequently. A further variant, New variant high-D Tucker (Definition 5) switches to a "dual" version where a hypercube is divided into cubelets, and points in the hypercube are coloured such that interiors of cubelets are monochromatic. A pair of points is sought having equal and opposite colours and distant by much less than the size of the cubelets.

We encode a point in n dimensions using a solution to an instance of Consensus-halving as follows. Instead of having just 2 cuts in the coordinate-encoding region (as in [24]), suppose we ensure that up to n cuts lie there. These cuts split this interval into $n + 1$ pieces whose total length is constant, so represent a point in the unit n-simplex (in [24], the unit 2-simplex). This "Möbius-simplex" (Definition 14; Figure 2) has the further property that two facets are identified with each other in a way that effectively turns the simplex into an n-dimensional Möbius strip.

In Section 5.2 we define a crucially-important coordinate transformation (see Figure 11 of the full version [25]) with the following key properties:

[-] the transformation and its inverse can be computed efficiently, and distances between transformed coordinate vectors are polynomially related to distances between untransformed vectors;

[-] at the two facets that are identified with each other, the coordinates of corresponding points are the negations of each other; our colouring function (that respects Tucker-style boundary conditions) has the effect that antipodal points get equal and opposite colours, and *no undesired solutions are introduced at these facets*.

This is the "smooth embedding" referred to in the abstract.

With the aid of the above coordinate transformation, we divide up the Möbius-simplex:

[-] The *twisted tunnel* (Definition 18) is an inverse-polynomially thick strip, connecting the two facets that are identified in forming the Möbius-simplex. It contains at its centre an embedded copy of the hypercube domain of an instance I_{VT} of New variant high-D Tucker. Outside of this embedded copy, it is "coloured in" (using our new coordinate system) in a way that avoids introducing solutions that do not encode solutions of I_{VT}.

[-] The *Significant Region* contains the twisted tunnel and constitutes a somewhat thicker strip connecting the two facets. It serves as a buffer zone between the twisted tunnel and the rest of the Möbius-simplex. It is subdivided into subregions where each subregion has a unique set of labels, or colours, from $\pm[n]$. (We sometimes refer to these as "colour-regions"). It is shown that any solution to an instance of Consensus-halving constructed as in our reduction, represents a point in the Significant Region.

[-] If, alternatively, a set of cuts represents a point from outside the Significant Region, then certain agents (so-called "blanket-sensor agents") will observe severe imbalances between labels A_+ and A_-, precluding a solution.

In [24], it was relatively straightforward to integrate the subset of the 2-dimensional Möbius-simplex that corresponds with the twisted tunnel, with the parts of the domain where the blanket-sensor agent became active (ruling out a solution) in a way that avoided introducing solutions that fail to encode solutions of the Tucker problem. In the present paper, that gap has to be "coloured-in" in a carefully-designed way (Section 5.3, list item 3), and this is the role of the part of the Significant Region that is not the twisted tunnel. The proofs that they work correctly become more complicated.

3 SNAKE EMBEDDING REDUCTION

The purpose of this section is to establish the PPA-completeness of New variant high-D Tucker, Definition 5. The snake embedding construction was devised in [13], in order to prove that ε-Nash equilibria are PPAD-complete to find when ε is inverse polynomial; without this trick the result is just obtained for ε being inverse exponential. We do a similar trick here. We will use as a starting-point the PPA-completeness of 2D-Tucker, from [1]. Due to lack of space, we omit all the details here, except the definition of the variant of Tucker that we will reduce from. The section establishes that this variant is PPA-complete; see the full version [25] for the details.

Definition 5. *An instance of* New variant high-D Tucker *in n dimensions is presented by a boolean circuit C_{VT} that takes as input coordinates of a point in the hypercube $B = [-1, 1]^n$ and outputs a label in $\pm[n]$ (assume C_{VT} has $2n$ output gates, one for each label, and is syntactically constrained such that exactly one output gate will evaluate to* true*), having the following constraints that may be enforced syntactically.*

(1) Dividing B into 7^n cubelets of edge length $2/7$ using axis-aligned hyperplanes, all points in the same cubelet get the same label by C_{VT};

(2) Interiors of antipodal boundary cubelets get opposite labels;

(3) Points on the boundary of two or more cubelets get a label belonging to one of the adjacent cubelets;

(4) Facets of B are coloured with labels from $\pm[n]$ such that all colours are used, and opposite facets have opposite labels. For $2 \leq i \leq n$ it also holds that the facet with label i (resp. $-i$) does not intersect any cubelet having label i (resp. $-i$). Facets coloured ± 1 are unrestricted (we call them the "panchromatic facets").

A solution consists of a polynomial number p^C of points that all lie within an inverse polynomial distance $\delta(n)$ of each other (for concreteness, assume $\delta(n) = 1/100n$). At least two of those points should receive equal and opposite labels by C_{VT}.

New variant high-D Tucker corresponds to the problem Variant Tucker in [24]; in that paper a solution only contained 100 points, while here we use p^C points. Here we need more points since we are in n dimensions, and our analysis needs to tolerate n points receiving unreliable labels.

4 SOME BUILDING-BLOCKS AND DEFINITIONS

Here we set up some of the general structure of the instances of Consensus-halving constructed in our reduction. We identify some basic properties of solutions to these instances. We define the *Möbius-simplex* and the manner in which a solution encodes a point on the Möbius-simplex. The encoding of the circuitry is covered in Section 5.

Useful quantities: We use the following values through the paper.

[-] δ^{tiny} is an inverse-polynomial quantity in n, chosen to be substantially smaller than any other inverse-polynomial quantity that we use in the reduction, apart from ε (below).

[-] δ^T is an inverse-polynomial quantity in n, which is smaller than any other inverse-polynomial quantity apart from δ^{tiny} and is larger than δ^{tiny} by an inverse-polynomial amount. The quantity δ^T denotes the width of the so-called "twisted tunnel" (see Definition 18).

[-] p^{huge} denotes a large polynomial in n; specifically we let $p^{\text{huge}} = n/\delta^{\text{tiny}}$. The quantity p^{huge} represents the number of sensor agents for each circuit encoder (see Definition 10).

[-] p^{large} denotes a large polynomial in n, which is however smaller than p^{huge} by a polynomial factor. The quantity p^{large} will be used in the definition of the "blanket-sensor agents" (see Definition 11) and will quantify the extent to which the cuts in the "coordinate-encoding region" (Definition 6) are allowed to differ from being evenly spaced, before the blanket-sensor agents become active (see Section 4). The choice of p^{large} controls the value δ_w of the radius of the Significant Region (see Proposition 7), with larger p^{large} meaning larger δ_w.

[-] ε is the precision parameter in the Consensus-Halving solution, i.e. each agent i is satisfied with a partition as long as $|\mu_i(A_+) - \mu_i(A_-)| \leq \varepsilon$. Henceforth, we will set $\varepsilon = \delta^{\text{tiny}}/10$.

4.1 Basic Building-blocks

We consider instances I_{CH} of Consensus-halving that have been derived from instances I_{VT} of New variant high-D Tucker in n dimensions. The general aim is to get any solution of such an instance I_{CH} to encode a point in n dimensions that "localises" a solution to I_{VT}, by which we mean that from the solution of I_{CH}, we will be able to find a point on the I_{VT} instance that can be transformed to a solution of I_{VT} in polynomial time and fairly straightforwardly.

Definition 6. **Coordinate-encoding region (c-e region)** *For a given instance of* Variant high-D Tucker *in n dimensions, the corresponding instance of* Consensus-halving *has a* coordinate-encoding region*, the interval $[0, n]$, a (prefix) subinterval of A.*

The valuation functions of the agents in an instance I_{CH} of Consensus-halving obtained by our reduction from an instance of New variant high-D Tucker in n dimensions, will be designed in such a way that either $n-1$ or n cuts (typically n) must occur in the coordinate-encoding region, in any solution. Furthermore, the distance between consecutive cuts must be close to 1 (an additive difference from 1 that is upper-bounded by an inverse polynomial), shown in Proposition 7.

Definition 7. **Coordinate-encoding agents (c-e agents).** *For a given instance of* New variant high-D Tucker *in n dimensions, the corresponding instance of* Consensus-halving *has n* coordinate-encoding agents *denoted $\{a_1, \ldots, a_n\}$.*

The n c-e agents have associated n coordinate-encoding cuts (Definition 8). It will be seen that the c-e cuts typically occur in the c-e region. The c-e agents do *not* have any value for the coordinate-encoding region; their value functions are only ever positive elsewhere. In particular, they have blocks of value whose labels A_+/A_- are affected by the output gates of the circuitry that is encoded to the right of the c-e region.

Definition 8. **Coordinate-encoding cuts (c-e cuts).** *We identify n cuts as the* coordinate-encoding cuts. *In the instances of the* Consensus-halving *problem that we construct, in any (sufficiently good approximate) solution to the* Consensus-halving *instance, all other cuts will be constrained to lie outside the c-e region (and it will be straightforward to see that the value functions of their associated agents impose this constraint). A c-e cut is not straightforwardly constrained to lie in the c-e region, but it will ultimately be proved that in any approximate solution, the c-e cuts do in fact lie in the c-e region.*

Recall that $p^{\text{huge}} = n/\delta^{\text{tiny}}$ from Section 2, which implies that the c-e region can be divided into p^{huge} intervals of length δ^{tiny} (see also Figure 1).

Definition 9. **σ-shifted version.** *Given a value function f (or measure) on the c-e region $[0, n]$, we say that another function f' on the c-e region is a σ-*shifted version *of f, when we have that $f'((x - \sigma) \mod n) = f(x)$.*

Recall that the circuit-encoding region (details in Section 5) contains p^C *circuit-encoders*, mentioned in the following definitions.

Definition 10. **Sensor agents.** *Each circuit-encoder C_i, for $i = 1, \ldots, p^C$, has a set $\mathcal{S}_i$ of sensor agents. $\mathcal{S}_i = \{s_{i,1}, \ldots, s_{i,p^{\text{huge}}}\}$ where the $s_{i,j}$ are defined as follows. When $i = 1$, $s_{1,j}$ has value $\frac{1}{10}$ uniformly distributed over the interval*

$$\left[(j-1)\delta^{\text{tiny}}, (j-1)\delta^{\text{tiny}} + \frac{\delta^{\text{tiny}}}{p^C}\right].$$

For $i > 1$, $s_{i,j}$ is a $\frac{1}{p^C}(i-1)\delta^{\text{tiny}}$-shifted version of $s_{1,j}$.

Each sensor agent $s_{i,j}$ also has valuation outside the c-e region, in non-overlapping intervals of the circuit-encoding region R_i (see Section 5.1). This valuation consists of two valuation blocks of value $\frac{9}{20}$ each, with no other valuation block in between. These are exactly as described in [24].

This value gadget for $s_{i,j}$ causes the j-th input gate in the circuit-encoder C_i to be set according to the label received by s_i's block of value in the c-e region, i.e. jump to the left or to the right in order to indicate that the corresponding value-block of s_i in the c-e region is labelled A_+ or A_-.

According to the definitions above, C_1 has a sequence of (a large polynomial number of) sensor agents that have blocks of value in a sequence of small intervals going from left to right of the c-e region (see Figure 1). For $1 < i \leq p^C$, C_i has a similar sequence, shifted slightly to the right on the c-e region (by $\delta^{\text{tiny}}(i-1)/p^C$). For $j \in [p^{\text{huge}}]$, the intervals defined by the value-blocks of the sensor agents $s_{1,j}, \ldots, s_{p^C,j}$ (for $C_1, \ldots, C_{p^C}$) partition the interval $[(j-1)\delta^{\text{tiny}}, j\delta^{\text{tiny}}]$.

Remark: Note that a c-e cut may divide one of the above value-blocks held by a sensor agent in the c-e region, and in that case the input being supplied (using the gadget of [24]) to its circuit-encoder is *unreliable*. However, only n sensor agents may be affected in that way, and their circuit-encoders will get "out-voted" by the ones that receive valid boolean inputs. This is part of the reason why we use p^C circuit-encoders in total. More details on this averaging argument are provided in Section 5.

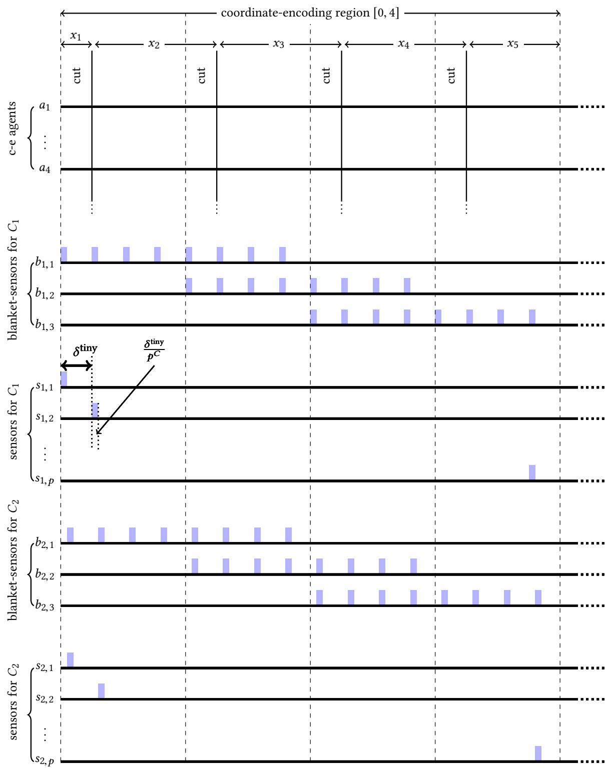

Figure 1: Sensor illustration: example of $n = 4$ c-e cuts representing 5 coordinates summing to 1 (a typical point in the Möbius-simplex). Vertical lines depict the cuts, resulting in labels that alternate between A_+ and A_-, starting with A_+. Shaded blocks over agents' lines indicate value-blocks of their value functionsWe only depict sensors for circuit-encoders C_1 and C_2.

Definition 11. **Blanket-sensor agents.** Each circuit-encoder C_i shall have $n - 1$ *blanket-sensor agents* $b_{i,2}, \ldots, b_{i,n}$. We omit the formal definition here, but we refer the reader to Figure 1 and to the full version [25]. Each blanket-sensor agent $b_{i,j}$ has an associated cut $c(b_{i,j})$ that lies in the subinterval $I_{i,j}$. Agent $b_{i,j}$ "monitors" an interval of length 2, namely the interval $[j-2, j]$ within which the sequence of $2/\delta^{\text{tiny}}$ value-blocks lie. If, in this interval, the number of these value-blocks labelled A_+ exceeds the number labelled A_- by at least p^{large} (recall that p^{large} is a large polynomial which is however polynomially smaller than p^{huge}) then (in any ε-approximate solution to I_{CH}, where, recall, $\varepsilon = \delta^{\text{tiny}}/10$), the cut $c(b_{i,j})$ in $I_{i,j}$ lies in either the right-hand or the left-hand value-block, otherwise it lies in the central value-block. Note that these three possible positions may be converted to boolean values that influence circuit-encoder C_i; this was referred to as a "bit-detection gadget" in [24], see the full version [25] for more details.

Definition 12 (Active blanket-sensor). *We say that blanket-sensor $b_{i,j}$ is* active *if $b_{i,j}$ in fact observes a sufficiently large label discrepancy in the c-e region, that $c(b_{i,j})$ lies in one of the two outer positions, left or right, and not in the central position. We say that $b_{i,j}$*

is active towards A_+ *if* A_+ *is the overrepresented label, with similar terminology for* A_-.

When blanket-sensor agent $b_{i,j}$ *is active, it provides input to* C_i *that causes* C_i *to label the value held by* a_j *and controlled by* C_i*, to be either* A_+ *or* A_-*; the choice depends on the over-represented label in* $[j-2, j]$ *and the parity of the index of the blanket-sensor agent. The precise feedback mechanism to the c-e agent* a_j *by the blanket-sensor* $b_{i,j}$ *is described in Section 5.1.3.*

When no blanket-sensors are active, the sequence of c-e cuts encodes a point in the Significant Region (Definition 15).

4.2 Features of Solutions

The main result of this section is Proposition 7, that in a solution to approximate Consensus-halving as constructed here, the sequence of cuts in the c-e region are "evenly spaced" in the sense that the gap between consecutive cuts differs from 1 by at most an inverse-polynomial.

Observation 4 (At most n cuts in the c-e region). *Given an instance* I_{CH} *derived by our reduction from an instance of* New variant high-D Tucker *in n dimensions, any inverse-polynomial approximate solution of* I_{CH} *has the property that at most n cuts lie in the coordinate-encoding region. This is because all other cuts are associated with agents who have at least* $9/10$ *of their value strictly to the right of the c-e region, thus in a solution, those cuts cannot lie in the c-e region.*

Definition 13 (Reliable input). *We will say that a circuit-encoder* receives reliable input*, if no coordinate-encoding cut passes through value-blocks of its sensor agents.*

Observation 5. *At most n circuit-encoders fail to receive reliable input (by Observation 4 and the fact that sensors of distinct circuit-encoders have value in distinct intervals).*

When a circuit-encoder receives reliable input, it is straightforward to interpret the labels allocated to its sensors, as boolean values, and simulate a circuit computation on those values, ultimately passing feedback to the c-e agents via value-blocks that get labelled according to the output gates of the circuit being simulated. This is done in a conceptually similar way to that described in [24] (e.g. see Sections 4.4.2 and 4.6 in [24]).

Definition 14. **The Möbius-simplex.** *The* Möbius-simplex in n dimensions *consists of points* $\mathbf{x}$ *in* $\mathbb{R}^{n+1}$ *whose coordinates are non-negative and sum to 1. We identify every point* $(x_1, \ldots, x_n, 0)$ *with the point* $(0, x_1, \ldots, x_n)$*, for all non-negative* $x_1, \ldots, x_n$ *summing to 1. We use the following metric* $d(\cdot,\cdot)$ *on the Möbius-simplex, letting* L_1 *be the standard* L_1 *distance on vectors:*

$$d(\mathbf{x}, \mathbf{x}') = \min\Big(L_1(\mathbf{x}, \mathbf{x}'), \min_{\mathbf{z},\mathbf{z}':\mathbf{z}\equiv\mathbf{z}'} (L_1(\mathbf{x}, \mathbf{z}) + L_1(\mathbf{z}', \mathbf{x}'))\Big) \qquad (1)$$

where $(0, x_1, \ldots, x_n) \equiv (x_1, \ldots, x_n, 0)$.

How a consensus-halving solution encodes a point in the Möbius-simplex. Let I_{CH} be an instance of Consensus-halving, obtained by reduction from New variant high-D Tucker in n dimensions, hence having c-e region $[0, n]$. Note that, by Observation 4, at most n cuts may lie in the c-e region. A set of $k \leq n$ cuts of the coordinate-encoding region splits it into $k+1$ pieces. We associate such a split with a point $\mathbf{x}$ in $\mathbb{R}^{n+1}$ as follows. The first coordinate is the distance from the LHS of the consensus-halving domain to the first cut, divided by n, the length of the c-e region. For $2 \leq i \leq k+1$, the i-th coordinate of $\mathbf{x}$ is the distance between the $i-1$-st and i-th cuts, divided by n. Remaining coordinates are 0.

If there are $n-1$ cuts in the c-e region, suppose we add a cut at either the LHS or the RHS. These two alternative choices correspond to a pair of points that have been identified as the same point, as described in Definition 14. (Observation 10 makes a similar point regarding transformed coordinates).

Observation 6. *Each circuit-encoder reads in "input" representing a point in the Möbius-simplex. Any circuit-encoder* C_i $(i \in [p^C])$ *behaves like* C_1 *on a point* $\mathbf{x}_i$*, for which (for all* $i, j \in [p^C]$*)* $d(\mathbf{x}_i, \mathbf{x}_j) \leq \delta^{\text{tiny}}$ *(recall d is defined in (1)). Consequently their collective output (the split between* A_+ *and* A_- *of the value held by the c-e agents) is the output of a single circuit-encoder averaged over a collection of* p^C *points in the Möbius-simplex, all within* δ^{tiny} *of each other.*

This follows by inspection of the way the p^C *circuit-encoders differ from each other: their sensor-agents are shifted but their internal circuitry is the same.*

Definition 15. **The Significant Region of the Möbius simplex** D. *The* Significant Region *of D consists of all points in D where no blanket-sensors are active (where "blanket-sensors" and "active" are defined in Definition 12).*

Proposition 7. *There is an inverse-polynomial value* δ^w *such that all points* $\mathbf{x} = (x_1, \ldots, x_{n+1})$ *in the Significant Region have coordinates* x_i *that for* $2 \leq i \leq n$ *differ from* $1/n$ *by at most* δ^w*, if* $\mathbf{x}$ *is encoded by the c-e cuts of an* ε*-approximate solution to one of our instances of* Consensus-halving*. (Recall that* $\varepsilon = \delta^{\text{tiny}}/10$*).*

Thus, if an instance I_{CH} *of* Consensus-halving *(obtained using our reduction) has a solution* S_{CH}*, then all the c-e cuts in* S_{CH} *have the property that the distance between two consecutive c-e cuts differs from 1 by at most some inverse-polynomial amount.*

The proof of the proposition can be found in the full version [25].

Remarks: Looking ahead, certain points in the Significant Region encode New variant high-D Tucker (namely, the ones in the "twisted tunnel", Definition 18). The Significant Region contains the twisted tunnel, being a somewhat wider 1-dimensional "tunnel" of inverse-polynomial width at most $1/p_w(n)$, whose central axis is the set of points $(\alpha, 1/n, \ldots, 1/n, 1/n - \alpha)$, where the endpoints are identified together (noting Definition 14). Topologically, the Significant Region is a high-dimensional Möbius strip.

5 REDUCING FROM NEW VARIANT HIGH-D TUCKER TO CONSENSUS HALVING

In Sections 5.1 we give an overview of aspects of how we construct an instance I_{CH} of ε-Consensus-halving (in poly-time) from an instance of New variant high-D Tucker, for inverse polynomial ε. Section 5.2 describes the new coordinate system for the Möbius-simplex D and establishes key properties. Section 5.3 presents a colouring function of D in terms of the coordinate system of Section 5.2. Section 5.4 describes how to construct a purported solution to n-dimensional New variant high-D Tucker from a solution to ε-Consensus-halving. In Section 6 we prove that a solution to New variant high-D Tucker that is obtained by reducing to ε-Consensus-halving, solving it, and converting that solution to a solution to n-dimensional New variant high-D Tucker, really is a valid solution.

5.1 Overview of the Construction of an Instance of Consensus Halving From an Instance of New Variant High-D Tucker

We define the reduction from New variant high-D Tucker (Definition 5) to ε-Consensus-halving.

Let I_{VT} be an instance of New variant high-D Tucker in n dimensions; let C_{VT} be the boolean circuit that represents it. I_{CH} will be the corresponding instance of Consensus-halving. We list ingredients of I_{CH} and give notation to represent them, as follows. A is the consensus-halving domain, an interval of the form $[0, poly(n)]$. Any agent a has a measure $\mu_a : A \longrightarrow \mathbb{R}$ represented as a step function (thus having a polynomial number of steps).

[-] I_{CH} has n *coordinate-encoding agents* $a_1, \ldots, a_n$ (Definition 7).

[-] The consensus-halving domain A of I_{CH} has a region called the *coordinate-encoding region* (c-e region) (Definition 6) consisting of the interval $[0, n]$.

[-] I_{CH} has p^C *circuit-encoders* (Sections 5.1.1, 5.1.4), $C_1, \ldots, C_{p^C}$.

- Each C_i has a set $\mathcal{A}_i$ of agents, which includes C_i's sensor agents, also circuit-encoding agents (below).
- Each C_i has an associated circuit-encoding region R_i of A; each R_i is an interval of polynomial length, and the R_i do not intersect with each other or with the coordinate-encoding region.
- $\mathcal{A}_i$ contains a polynomial number of *circuit-encoding agents* (one for each gate of C_{VT}), having value in R_i.
- Each C_i has p^{huge} *sensor agents* as defined in Definition 10 each of which has a block of value $1/10$ in a small subinterval of the c-e region as specified in Definition 10, and further value in region R_i.
- Each C_i has $n-1$ *blanket-sensor agents* as in Definition 11.

Remarks: We associate one cut with each agent; let $c(a)$ be the cut associated with agent a. The cuts $c(a_i)$ for coordinate-encoding agents, are called the *coordinate-encoding cuts* (or c-e cuts). A rather straightforward consequence of Proposition 7 is that in any solution, either all n, or $n-1$, of the coordinate-encoding cuts must lie in the coordinate-encoding region. All other cuts must lie in the regions R_i, indeed, every cut, other than the c-e cuts, is constrained by the value of its associated agent, to lie in a small interval that does not overlap any other such intervals. In the event that a c-e cut lies outside the c-e region, we refer to it as a "stray cut", and while such a cut may initially appear to interfere with the functioning of the circuitry, similarly to [24] we have that the duplication of the circuit using p^C circuit-encoders, allows the circuitry to be robust to this problem. See the full version for more details.

5.1.1 Construction of C_1. Recall C_{VT} is the boolean circuit in the instance I_{VT} of New variant high-D Tucker.

[-] We assume that C_{VT} has $2n$ output gates, namely $g_1, \ldots, g_n$ and $g_{-1}, \ldots, g_{-n}$ having the property that exactly one of them will take value true (this may be enforced syntactically). g_i getting value 1 (true) means that the point at coordinates represented by the input gets coloured i.

[-] C_{VT} has $n \cdot \text{polylog}(n)$ input gates, representing the coordinates of a point in $B = [-1, 1]^n$, each represented with inverse-polynomial precision.

We describe how circuit-encoder C_1 is derived from C_{VT}. The subsequent circuit-encoders can then be specified in terms of C_1. Each gate g of C_{VT} is simulated using a "gate agent" $a(g)$, as constructed in [24]. $a(g)$'s cut $c(a(g))$ occupies a right position, or a left position, representing true or false, as a function of the cut(s) that represent boolean inputs to g.

The circuit-encoding agents $\mathcal{A}_1$ of C_1 thus include $2n$ gate agents whose corresponding cuts simulate the values of the output gates of C_{VT}, provided that the input represented by the c-e cuts lies in the "Significant Region" (Definition 15). The positions of these cuts affect the labels of blocks of value held by the n coordinate-encoding agents, as detailed in Section 5.1.2.

Definition 16. **Reference sensor-agent.** *From Definition 10, the sensor agents for C_i are denoted by $\mathcal{S}_i = \{s_{i,1}, \ldots, s_{i,p^{\text{huge}}}\}$. We let $s_{1,1}$ be the* reference sensor-agent*: Outputs produced by the circuit C_i are taken with reference to the value[1] $s_{1,1}$, in the sense that after simulating C_{VT} we take the exclusive-or with $s_{1,1}$.*

We used this crucial technique of Definition 16 in [24]: it performs the task of disorienting the domain while at the same time ensuring continuity when we move a cut from the left-hand side of the c-e region to the right-hand side.

Preprocessing, prior to simulating C_{VT}. For each C_i, we take all p^{huge} input bits, which appear in up to $n+1$ blocks of consecutive 1's and 0's, and convert them into the coordinates of a point in the Möbius-simplex (Definition 14). As noted earlier (Observation 5) at most n circuit-encoders may receive ill-defined inputs caused by c-e cuts cutting through value-blocks in the c-e region that belong to their sensor agents; we simply assume that the output of those agents is unreliable, indeed adversarially chosen.

We then perform a coordinate transformation described in Section 5.2. A subset of points in the Möbius-simplex maps to a copy of the domain B of the instance I_{VT} (recall Definition 5). These points get their coordinates passed directly to a copy of C_{VT}, and the outputs of C_{CV} are used to provide feedback to the c-e agents as described in Section 5.1.2 (and discussed in Observations 5 and 6). Other points get coloured in a manner that avoids allowing bogus solutions to I_{CH} (ones that do not encode solutions to I_{VT}).

5.1.2 Output gates of C_1, and the Feedback they Provide to the Coordinate-encoding Agents. The following generalises [24]. C_{VT} has output gates g_j, $j \in \pm[n]$, with the property that when inputs are well-defined, exactly one output gate evaluates to true. A circuit-encoder simulates C_{VT} using the gate gadgets introduced in [23, 24]. Let $x_{REF} \in \{\text{true}, \text{false}\}$ be the negation of the value of the reference sensor (Definition 16). We use additional gates g'_j, $j \in \pm[n]$ where

- if $g_j = g_{-j} = \text{false}$ then $g'_{|j|} = \text{true}$ and $g'_{-|j|} = \text{false}$;

[1] We are using $s_{1,1}$ to denote the boolean value taken by $s_{1,1}$ as well as the sensor.

- if $j > 0$ and $g_j = \text{TRUE}$ (so $g_{-j} = \text{FALSE}$) then $g'_j = g'_{-j} = \text{TRUE} \oplus x_{REF}$;
- if $j < 0$ and $g_j = \text{TRUE}$ (so $g_{-j} = \text{FALSE}$) then $g'_j = g'_{-j} = \text{FALSE} \oplus x_{REF}$;

Each of the c-e agents $a_1, \ldots, a_n$ has 2 value-blocks of value $1/(2p^C)$ in region R_1, and each gate g'_j of C_1 is able to select the label of one of these value-blocks (recall that the boolean value at a gate is represented by two positions that may be taken by the corresponding cut, so that a block of value lies between these two positions).

5.1.3 How C_1's Blanket-sensors Affect the Feedback Mechanism. Let $A_j \in \{A_+, A_-\}$ and let $A_{-j} \in \{A_+, A_-\}$, $A_{-j} \neq A_j$ be the complementary label. The blanket-sensor agents $b_{1,2}, \ldots, b_{1,n}$ affect the output of the circuit-encoders as follows:

(1) If none are active, the $2n$ outputs of C_1 are computed as described in Section 5.1.2.
(2) If j is *odd* and $b_{1,j}$ is active in direction A_j then the output gates g'_j, g'_{-j} are both set to the value that causes c-e agent a_j to observe more A_j.
(3) If j is *even* and $b_{1,j}$ is active in direction A_j then the output gates g'_j, g'_{-j} are both set to the value that causes c-e agent a_j to observe more A_{-j}.

Rules 2 and 3 override Rule 1, which allocates values that directly encode values output by C_{VT}. Note that the gadgetry of the circuit can ensure that either an excess of A_+ or an excess of A_- can be shown to the corresponding c-e agent as feedback, as the circuit can convert the input value encoded by the value gadget of the blanket-sensor agent in R_1 to either a "right" or "left" output position, depending on the parity of the index. Also, if more than one blanket-sensor agent is active, they all affect their corresponding gates. The reason for requiring blanket-sensors of different parities to feedback different labels to the c-e agents is to be consistent with a property that we refer to as "consistent colours", which is used for arguing that the blanket-sensors do not introduce artificial solutions in Section 6. The formal definition of consistent colours is given in the full version [25].

Note that we do not define the behaviour of the blanket-sensor agents in terms of the reference sensor. They essentially look for an imbalance between A_+ and A_- within some interval of length 2, and when they find a sufficiently large imbalance, they force the circuit C_1 to show their associated c-e agent more of the over-represented or under-represented label, depending on their parity, which can be done using gadgetry of [24].

5.1.4 Construction of Circuit-encoders $C_2, \ldots, C_{p^C}$. The contruction of the remaining circuit-encoders is done in a manner very similar to [24]; we refer the reader to the full version for the details.

Observation 8. *The value that is labelled A_+ held by any c-e agent a_j, is the average of the output values that the C_i's allocate to a_j. If, say, all the C_i receive inputs representing a point in the significant region with label ℓ, then a_ℓ observed an imbalance between A_+ and A_-, but a_j for $j \neq \ell$ will have g'_j output the opposite value to g'_{-j}, resulting in a_j's value-blocks receiving opposite labels.*

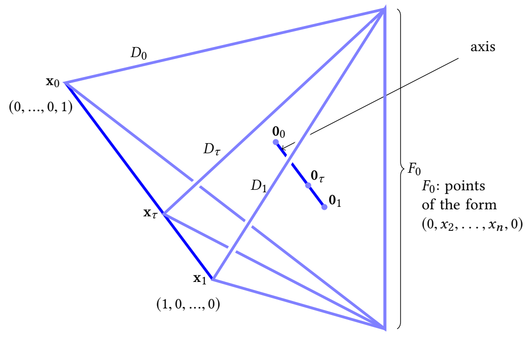

Figure 2: Subspaces of the Möbius-simplex D: D_0 is the triangle spanned by $\mathbf{x}_0$ and F_0, and $\mathbf{0}_0$ is its centre; similarly for D_τ and D_1.

5.2 An Alternative Coordinate System for the Möbius-simplex

Recall that the Möbius-simplex D is the n-simplex consisting of points $(x_1, \ldots, x_{n+1})$ whose components are non-negative and sum to 1. Furthermore, a typical point in D is directly encoded via the positions of n cuts in the c-e region.

Here we specify a transformed coordinate system that is needed in order to encode instances of New variant high-D Tucker. We will embed the hypercube-shaped domain of an instance of New variant high-D Tucker in a hypercube in the transformed coordinates, and then use properties of the transformed coordinate system to extend the labelling function to the rest of the domain in a way that does not introduce bogus solutions (i.e. fixpoints of the extended function that lie outside the hypercube and do not encode solutions of New variant high-D Tucker).

Let F_0 be the set of points in D of the form $(0, x_2, \ldots, x_n, 0)$; thus F_0 is a $(n-2)$-face of D. See Figure 2. For $\tau \in [0, 1]$, let $\mathbf{x}_\tau$ be the point

$$\mathbf{x}_\tau := \tau(1, 0, \ldots, 0) + (1-\tau)(0, \ldots, 0, 1) = (\tau, 0, \ldots, 0, 1-\tau).$$

(So, $\mathbf{x}_0$ and $\mathbf{x}_1$ are the endpoints of the 1-dimensional edge of D that is not contained in F_0). Let D_τ be the $(n-1)$-simplex consisting of convex combinations of F_0, and $\mathbf{x}_\tau$. Thus D_0 and D_1 are the two facets of D that have been identified together as in Definition 14.

D_τ contains the point $\mathbf{0}_\tau = (\tau/n, 1/n, \ldots, 1/n, (1-\tau)/n)$, which we regard as the origin of D_τ. The set of points $\{\mathbf{0}_\tau : 0 \leq \tau \leq 1\}$ will be referred to as the *axis*; it will transpire that all solutions must lie within an inverse polynomial distance from the axis (in particular will be in the Significant Region).

We then refer to points in D_τ by means of the coordinates in a coordinate system that itself is a linear function of τ. With respect to any fixed $\tau \in [0, 1]$ we define $n-1$ vectors $(d_2^\tau, \ldots, d_n^\tau)$ as follows. A key feature is that $(d_2^\tau, \ldots, d_n^\tau)$ form a basis of D_τ (so that with respect to the origin $\mathbf{0}_\tau$, any point in D_τ has unique coordinates). The other key feature (Observation 10) is that at $\tau = 0$ the coordinate/directions are "equal and opposite" to the coordinates at $\tau = 1$. Also, the coordinate system varies suitably smoothly.

As a warm-up we start by considering d_2^τ:

$$d_2^\tau := (1-\tau)(0,1,-1,0,\dots,0) + \tau(-1,1,0,\dots,0).$$

d_2^τ consists of increasing the second coordinate at the expense of its neighbours. For small τ we increase mainly at the expense of the third coordinate and as τ increases, we increase the second coordinate more at the expense of the first. Notice in particular that at $\tau = \frac{1}{2}$ we have $d_2^\tau = (-\frac{1}{2}, 1, -\frac{1}{2}, 0, \dots, 0)$.

Generally, for $2 \le i \le n$ we define

$$d_i^\tau := (1-\tau)(\underbrace{0,\dots,0}_{i-1 \text{ zeroes}}, 1, -1, \underbrace{0,\dots,0}_{n-i}) + \tau(\underbrace{0,\dots,0}_{i-2 \text{ zeroes}}, -1, 1, \underbrace{0,\dots,0}_{n-i+1}) \tag{2}$$

Thus, again this consists of the i-th coordinate increasing at the expense of its neighbours, and we have in particular

$$d_i^{\frac{1}{2}} = (\underbrace{0,\dots,0}_{i-2 \text{ zeroes}}, -\frac{1}{2}, 1, -\frac{1}{2}, 0, \dots, 0)$$

For $i = 2,\dots,n$, define

$$d_{-i}^\tau := -d_i^\tau. \tag{3}$$

Observation 9 makes the important point that by linearity, the vectors d_i^τ, $i = 2,\dots,n$, can be used as a coordinate system to refer to points in D_τ.

Observation 9. *Any point* $\mathbf{x}$ *in* D_τ *can be uniquely expressed as a sum*

$$\mathbf{x} = \mathbf{0}_\tau + \sum_{i=2}^{n} \alpha_i d_i^\tau. \tag{4}$$

To see this, note first that D_0 *is points in* D *of the form* $(0, x_2, \dots, x_{n+1})$, *and* D_1 *is points of the form* $(x_1, \dots, x_n, 0)$. *Note that the observation certainly works for* $\tau = 0$ *or* $\tau = 1$. *To see that it works for intermediate* τ, *note that the vectors* d_i^τ *are linearly independent, and the reason why they span* D_τ *is that any vector* d_i^τ *is equal to* $(1-\tau)$ *multiplied by a vector in* D_0, *added to* τ *multiplied by an equal-length vector in* D_1. *So these vectors do indeed lie in* D_τ.

Definition 17. *For a point* $\mathbf{x} \in D_\tau$ *as in (4) we say that the* transformed coordinates *of* $\mathbf{x}$ *are* $(\alpha_2, \dots, \alpha_n)$. *More generally, a point* $\mathbf{x} \in D$ *can be expressed as* $(\tau; \alpha_2, \dots, \alpha_n)$, *where* τ *is chosen such that* $\mathbf{x} \in D_\tau$. *We use the following metric* $\tilde{d}(\cdot,\cdot)$ *on transformed coordinate vectors, where similarly to (1),* L_1 *denotes the standard* L_1 *distance on vectors.*

$$\tilde{d}(\mathbf{x}, \mathbf{x}') = \min\Big(L_1(\mathbf{x}, \mathbf{x}'), \min_{\mathbf{z}, \mathbf{z}': \mathbf{z} \equiv \mathbf{z}'} (L_1(\mathbf{x}, \mathbf{z}) + L_1(\mathbf{z}', \mathbf{x}'))\Big) \tag{5}$$

where $(0; \alpha_2, \dots, \alpha_n) \equiv (1; -\alpha_2, \dots, -\alpha_n)$.

Observation 10. *With regard to Definition 17, consider two points* $\mathbf{x} = (0; \alpha_2, \dots, \alpha_n)$, $\mathbf{x}' = (1; -\alpha_2, \dots, -\alpha_n)$, *that have been equated with each other. Assume these points are near the axis, specifically* $|\alpha_j| < 1/10n$ *for all j. Notice that*

- *the cuts in the c-e region for* $\mathbf{x}$ *and* $\mathbf{x}'$ *partition the c-e region in the same way.*
- *When we move from a point in* $D_{1-\varepsilon}$ *to a nearby point in* D_ε, *for any* $j \in \{2, \dots, n\}$, *the direction of increasing* α_j *segues* smoothly *to the direction of* decreasing α_j *(see Figure 11 in the full version for a reference).*

Our assumption that $|\alpha_j| < 1/10n$ *ensures that cuts are fairly evenly-spaced, and movement in any of the directions* d_j^τ *does not cause the cuts to cross each other.*

Proposition 11 says that if we perturb a point $\mathbf{x} \in D$ that lies close to the axis, then the total perturbation of the transformed coordinates of $\mathbf{x}$ is polynomially related to the total perturbation of the untransformed coordinates; d and $\tilde{d}$ are polynomially related.

Proposition 11 (Polynomial distance relation). *There is some polynomial* $p(n)$ *such that for all* $\mathbf{x}, \mathbf{x}' \in D$ *within Euclidean distance* $1/10n^2$ *of the axis, letting* $\tilde{\mathbf{x}}$ *and* $\tilde{\mathbf{x}}'$ *be their transformed coordinates, and letting* $d, \tilde{d}$ *be the metrics defined as in (1),(5), we have*

$$\frac{1}{p(n)} \le \frac{d(\mathbf{x}, \mathbf{x}')}{\tilde{d}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')} \le p(n).$$

Note that a similar result would hold if in the definitions of the metrics d and $\tilde{d}$, we replace the L_1 metric with, say, L_2 or L_∞, since these are polynomially related to L_1.

Note that Proposition 11 does not hold for all points in D; the restriction to a neighbourhood of the axis is needed. For points on F_0, all values of τ are equivalent, and for points close to F_0, perturbed versions of them could result in large perturbations of τ.

5.2.1 *Computation of the Transformation, and its Inverse.* The coordinate transformation, and its inverse, can be computed in polynomial time, for points in D that are within some inverse polynomial distance from the axis. Furthermore, small perturbations of inputs lead to small perturbations of the outputs (in either direction). We leave the details for the full version [25] in the interest of space.

5.3 A (Poly-time Computable) Partial Colouring Function

This section defines a partial function $f : D \longrightarrow \{-1, 0, 1\}^n$ (D being the n-dimensional Möbius-simplex (Definition 14)). However, some of the details are omitted due to lack of space; the reader is referred to [25] for the full exposition.

Let B be the n-dimensional "box" associated with I_{VT} (recall I_{VT} is represented by circuit C_{VT} that maps points in B to $\pm[n]$). We embed a copy of B in D as follows. Recall the way facets of B are coloured in Definition 5. Let $(x_1, \dots, x_n)$ denote a typical point in B, and assume that the facets of B with maximum and minimum x_1 (i.e. $x_1 = 1$ and $x_1 = -1$ respectively) are the panchromatic facets of B (as in Definition 5), and for $i \ge 2$ the facet of B with maximum x_i ($x_i = 1$) consists of points that do not have colour i, and the the facet of B with minimum x_i ($x_i = -1$) consists of points that do not have colour $-i$.

Definition 18. *The* twisted tunnel T *is defined as follows. The* axis *of* T *is the set of all points* $\mathbf{0}_\tau$ *as defined in Section 5.2. The twisted tunnel is the set of all points with transformed coordinates* $(\tau; \alpha_2, \dots, \alpha_n)$ *such that for all i,* $|\alpha_i| < \delta^T$. *Note that* δ^T *is an inverse polynomial quantity sufficiently small that T is a subset of the Significant Region; this is achieved since by definition,* δ^T *is polynomially smaller than* δ^w *of Proposition 7. Thus, T has (with respect to the transformed coordinates) a* $(n-1)$*-cube-shaped intersection with any* D_τ.

We define the behaviour of f over the Significant Region (Definition 15) in 3 stages, as follows.

(1) **Embedding B in D (recall $B = [-1, 1]^n$)**

A point $\mathbf{x} = (x_1, \ldots, x_n)$ in B is mapped to a point $g(\mathbf{x})$ in D as follows. $g(\mathbf{x})$ lies in D_τ, where we choose $\tau = \frac{1}{2} + \delta^T \cdot x_1$. Then (noting (4)) we set $g(\mathbf{x})$ equal to $\mathbf{0}_\tau + \sum_{i=2}^n \delta^T \cdot x_i d_i^\tau$ (i.e. $g(\mathbf{x})$ has transformed coordinates $(\frac{1}{2} + \delta^T x_1; \delta^T x_2, \ldots, \delta^T x_n)$). $g(\mathbf{x})$ will receive a single colour; the colour of $g(\mathbf{x})$ — i.e. the non-zero entry of $f(g(\mathbf{x}))$ — is set equal to the colour allocated to $\mathbf{x}$ in B by I_{VT}. (Notice that the centre of B is mapped to $(1/2n, 1/n, \ldots, 1/n, 1/2n)$, which is the origin of $D_{\frac{1}{2}}$, and the centre of the Significant Region. This point has (recalling Definition 17) transformed coordinates $(\frac{1}{2}; 0, \ldots, 0)$ where the first entry is the value of τ).

(2) **Extending f to be defined on T**

We also colour other points in T as follows — these will also receive single colours. Suppose $\mathbf{y}$ belongs to D_τ, where $\tau < \frac{1}{2} - \delta^T$ or $\tau > \frac{1}{2} + \delta^T$. According to (4), $\mathbf{y} = \mathbf{0}_\tau + \sum_{i=2}^n \alpha_i d_i^\tau$, and $\mathbf{y}$ has transformed coordinates $(\tau; \alpha_2, \ldots, \alpha_n)$. Suppose all the α_i lie in the range $[-\delta^T, \delta^T]$. Then if $\tau < \frac{1}{2} - \delta^T$, we set the colour of $\mathbf{y}$ to the colour of a point $\mathbf{y}' = (\frac{1}{2} - \delta^T; \alpha_2, \ldots, \alpha_n)$. Thus $\mathbf{y}' \in D_{\frac{1}{2} - \delta^T}$, and the other transformed coordinates (Definition 17) are the same for $\mathbf{y}$ and for $\mathbf{y}'$. We do a similar thing for points in D_τ for $\tau > \frac{1}{2} + \delta^T$. That is, if $\mathbf{y}$ has transformed coordinates $(\tau; \alpha_2, \ldots, \alpha_n)$ where $\tau > \frac{1}{2} + \delta^T$ and the α_i are all at most δ^T in absolute value, then $\mathbf{y}$ gets the same colour as a point $\mathbf{y}'$ whose transformed coordinates are $(\frac{1}{2} + \delta^T; \alpha_2, \ldots, \alpha_n)$.

(3) **Extending f to the Significant Region**

The Significant Region (Definition 15) is points in D where no blanket-sensor agents are active, a subset of points that are close to the axis in the sense of Proposition 7. Consider $\mathbf{x} \in D \setminus T$ with transformed coordinates $(\tau; \alpha_2, \ldots, \alpha_n)$.

(1) For each $j \in \{2, \ldots, n\}$ if $\alpha_j > \delta^T$ then $\mathbf{x}$ gets colour $-j$;
(2) For each $j \in \{2, \ldots, n\}$ if $\alpha_j < -\delta^T$ then $\mathbf{x}$ gets colour j;
(3) these are not mutually exclusive, $\mathbf{x}$ gets at least one colour, possibly more.

Notice that (within the subspace D_τ) the side(s) of the twisted tunnel T closest to $\mathbf{x}$ is guaranteed not to be opposite to any colour of $\mathbf{x}$. For a subset S of colours, let $R(S)$ be the region with colours in S. We call these the “outer regions”.

In Section 6 (of the full version), we prove that when colour-regions meet each other at opposite ends of T (which have been identified with each other according to the definition of the Significant Region), they will have equal and opposite colours.

5.4 Mapping the Solutions

This section deals with how to compute a solution to New variant high-D Tucker from a solution to Consensus-halving. Let $\mathbf{x}$ be the point in the Möbius-simplex represented by the c-e cuts of S_{CH}. Proposition 7 already tells us that $\mathbf{x}$ must lie close to the axis. We prove that $\mathbf{x}$ must lie within, or very close to, the twisted tunnel. We leave the details for the full version, due to lack of space.

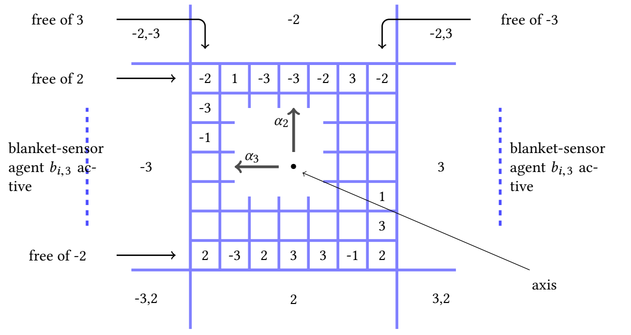

Figure 3: Cross-section of the twisted tunnel (for $n = 3$), with examples of possible labels of regions. Note that the outer regions of the Significant Region are not adjacent to any cubelet having an opposite colour to that outer region. For example, the left-hand column is free of cubelets with colour 3, and is adjacent to the outer region with colour -3.

6 COMPLETING THE PROOF

This section is devoted to completing the reduction. In particular, we first construct a Borsuk-Ulam “style” function f' based on f, which simulates the effect of the blanke-sensor agents, as described in Section 5.1.3. Then, we explain how a solution to an instance of New variant high-D Tucker can be reconstructed by a set of cuts of Consensus-halving representing a point x in the twisted tunnel. Finally, we argue that this function does not introduce any artificial solutions at the boundary of the Significant Region or due to the connecting facet. Due to lack of space, we include all the details in the full version [25].

7 FURTHER WORK

What is the complexity of k-thief Necklace-splitting, for k not a power of 2? As discussed in [39, 45], the proof that it is a total search problem, does *not* seem to boil down to the PPA principle. Right now, we do not even even know if it belongs to PTFNP [26].

In classifying a problem as polynomial-time solvable versus NP-complete, this is usually seen as a statement about its computational (in)tractability. The distinction between PPAD-completeness and PPA-completeness is one of expressive power: we believe that PPAD-complete problems are hard, meanwhile PPA-complete problems are “at least as hard”, but of course are still in NP. The expressive power of totality principles that underpin TFNP problems is a topic of enduring interest [6, 26]; note also the related work on Bounded Arithmetic discussed in [26]. Our results highlight the distinction between computational (in)tractability and expressive power. In analysing the relationships between these complexity classes, it may be fruitful to focus on expressive power.

Finally, [43] initiates an interesting experimental study of path-following algorithms for 2-thief Necklace-splitting, obtaining positive results when the number of bead colours is not too large. However, path-following seems to be inapplicable for, say, 3 thieves. The Necklace-splitting problem may constitute an interesting class of challenge-instances for SAT-solvers, now that it is known to be a very hard total search problem.

REFERENCES

[1] J. Aisenberg, M.L. Bonet, and S. Buss. 2-D Tucker is PPA complete. *Electronic Colloquium on Computational Complexity*, TR15-163 (2015).

[2] N. Alon. Splitting Necklaces. *Advances in Mathematics*, **63**(3), pp. 247–253 (1987).

[3] N. Alon. Some recent Combinatorial Applications of Borsuk-type Theorems. *London Mathematical Society Lecture Notes* Series 131, Algebraic, Extremal and Metric Combinatorics 1986, pp. 1–12. CUP (1988).

[4] N. Alon. Non-Constructive Proofs in Combinatorics. *Proceedings of International Congress of Mathematicians 1990*, pp. 1421–1429. Tokyo, Springer-Verlag (1991).

[5] N. Alon and D.B. West. The Borsuk-Ulam theorem and bisection of necklaces. *Proceedings of the American Mathematical Society*, **98**(4), pp. 623–628 (1986).

[6] P. Beame, S. Cook, J. Edmonds, R. Impagliazzo and T. Pitassi. The relative complexity of NP search problems. *Journal of Computer and System Sciences* **57**, pp. 3–19 (1998).

[7] A. Belovs, G. Ivanyos, Y. Qiao, M. Santha, and S. Yang. On the Polynomial Parity Argument complexity of the Combinatorial Nullstellensatz. *Proceedings of the 32nd Computational Complexity Conference*, Article No. 30 (2017).

[8] S.N. Bhatt and C.E. Leiserson. How to Assemble Tree Machines (Extended Abstract). *Proceedings of the 14th annual ACM Symposium on Theory of Computing (STOC)*, pp. 77–84 (1982).

[9] P.V.M. Blagojević and P. Soberón. Thieves can make sandwiches. *Bulletin of the London Mathematical Society*, 50(1) pp. 108–123 (2018).

[10] P. Bonsma, Th. Epping, and W. Hochstättler. Complexity results on restricted instances of a paint shop problem for words. *Discrete Applied Mathematics*, 154, pp. 1335–1343 (2006).

[11] X. Chen, D. Dai, Y. Du, and S.-H. Teng. Settling the Complexity of Arrow-Debreu Equilibria in Markets with Additively Separable Utilities. *Proceedings of FOCS*, Atlanta, USA, pp. 273–282 (2009).

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

[13] X. Chen, X. Deng, and S.-H. Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM* **56**(3) pp. 1–57 (2009).

[14] X. Chen, D. Durfee, and A. Orfanou. On the Complexity of Nash Equilibria in Anonymous Games. *Proceedings 47th STOC*, Portland, USA, pp. 381–390 (2015).

[15] X. Chen, D. Paparas, and M. Yannakakis. The complexity of non-monotone markets. *Journal of the ACM*, **64**(3), article 20 (2017)..

[16] B. Codenotti, A. Saberi, K. Varadarajan, and Y. Ye. The complexity of equilibria: Hardness results for economies via a correspondence with games. *Theoretical Computer Science*, **408** pp. 188–198 (2008).

[17] C. Daskalakis, P.W. Goldberg, and C.H. Papadimitriou. The Complexity of Computing a Nash Equilibrium. *SIAM Journal on Computing* **39**(1) pp. 195–259 (2009).

[18] X. Deng, J.R. Edmonds, Z. Feng, Z. Liu, Q. Qi, and Z. Xu. Understanding PPA-completeness. *31st Conference on Computational Complexity*, article no. 23, pp. 23:1–23:25 (2016).

[19] X. Deng, Z. Feng, and R. Kulkarni. Octahedral Tucker is PPA-complete. *Electronic Colloquium on Computational Complexity* Report TR17-118 (2017).

[20] X. Deng, Q. Qi, and A. Saberi. On the Complexity of Envy-Free Cake Cutting. *CoRR*, arXiv:0907.1334 (2009).

[21] H. Edelsbrunner and R. Waupotitsch. Computing a Ham-Sandwich Cut in Two Dimensions. *Journal of Symbolic Computation*, **2**(2) pp. 171–178 (1986).

[22] E. Elkind, L.A. Goldberg and P.W. Goldberg. Nash Equilibria in Graphical Games on Trees Revisited. *Proceedings of the 7th ACM Conference on Electronic Commerce (ACM EC)*, Ann Arbor, Michigan, USA, pp. 100–109 (2006).

[23] A. Filos-Ratsikas, S. K. S. Frederiksen, P.W. Goldberg, and J. Zhang. Hardness Results for Consensus-Halving. *Proceedings of the 43rd MFCS Symposium (MFCS 2018)*, pp. 24:1–24:16 (2018).

[24] A. Filos-Ratsikas and P.W. Goldberg. Consensus-Halving is PPA-Complete. *Proceedings of the 50th STOC*, pp. 51–64 (2018).

[25] A. Filos-Ratsikas and P.W. Goldberg. The Complexity of Splitting Neckclaces and Bisecting Ham Sandwiches. *Full Version*, arXiv preprint arXiv:1805.12559 (2019).

[26] P.W. Goldberg and C.H. Papadimitriou. Towards a Unified Complexity Theory of Total Functions. *Journal of Computer and System Sciences*, **94**, pp. 167–192 (2018).

[27] C.H. Goldberg and D.B. West. Bisection of Circle Colorings. *SIAM Journal on Algebraic Discrete Methods* **6** (1) pp. 93–106 (1985).

[28] F. Grandoni, R. Ravi, M. Singh, and R. Zenklusen. New approaches to multi-objective optimization. *Mathematical Programming Series A* **146** pp. 525–554 (2014).

[29] M. Grigni. A Sperner Lemma Complete for PPA. *Information Processing Letters* **77** (5–6) pp. 255–259 (2001).

[30] T.P. Hill. Determining a Fair Border. *The American Mathematical Monthly*, **90**(7) pp. 438–442 (1983).

[31] C.R. Hobby and J.R. Rice, A Moment Problem in L_1 Approximation. *Proceedings of the American Mathematical Society* 16(4) pp.665–670 (1965).

[32] E. Jeřábek. Integer factoring and modular square roots. *Journal of Computer and System Sciences* **82**(2) pp. 380–394 (2016).

[33] C.S. Karthik and A. Saha. Ham Sandwich is Equivalent to Borsuk-Ulam. *Symposium on Computational Geometry* pp. 24:1–24:15 (2017).

[34] S. Kintali, L.J. Poplawski, R. Rajaraman, R. Sundaram, and S.-H. Teng. Reducibility among Fractional Stability Problems. *SIAM Journal on Computing*, **42**(6) pp. 2063–2113 (2013).

[35] C. Knauer, H.R. Tiwary, D. Werner. On the computational complexity of Ham-Sandwich cuts, Helly sets, and related problems. *Proceedings of STACS* pp. 649–660 (2011).

[36] C-Y Lo, J. Matoušek, and W. Steiger. Ham-sandwich cuts in R^d. *Proceedings of the 24th STOC*, pp. 539–545 (1992).

[37] Chi-Yuan Lo, Jiri Matoušek, and William Steiger. Algorithms for Ham-Sandwich Cuts. *Discrete and Computational Geometry* **11**(4) pp. 433–452 (1994).

[38] J.A. De Loera, X. Goaoc, F. Meunier, and N. Mustafa. The discrete yet ubiquitous theorems of Carathéodory, Helly, Sperner, Tucker, and Tverberg. *CoRR*, arXiv:1706.05975 (2017).

[39] M. de Longueville and R.T. Živaljević. The Borsuk-Ulam-property, Tucker-property and constructive proofs in combinatorics. *Journal of Combinatorial Theory, Series A* 113 pp. 839–850 (2006).

[40] J. Matoušek. Geometric range Searching. *ACM Computing Surveys.* **26**(4) pp. 421–461 (1994).

[41] Matoušek, Jiří. Lectures on discrete geometry. Vol. 212. New York: Springer, 2002.

[42] F. Meunier. Discrete Splittings of the Necklace. *Mathematics of Operations Research*, 33, No. 3, pp. 678–688 (2008).

[43] F. Meunier and B. Neveu. Computing solutions of the paintshop-necklace problem. *Computers and Operations Research*, 39(11), pp. 2666–2678 (2012).

[44] F. Meunier and A. Sebő. Paintshop, odd cycles and necklace splitting. *Discrete Applied Mathematics*, 157, pp. 780–793 (2009).

[45] F. Meunier. Simplotopal maps and necklace splitting. *Discrete Mathematics*, 323, pp. 14–26 (2014).

[46] N. Megiddo and C.H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, **81**(2) pp. 317–324 (1991).

[47] R. Mehta. Constant rank bimatrix games are PPAD-hard. *Proceedings of 46th STOC*, New York, USA, pp. 545–554 (2014).

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

[49] S. Roy and W. Steiger. Some Combinatorial and Algorithmic Applications of the Borsuk-Ulam Theorem. *Graphs and Combinatorics*, 23, Supplement 1, pp. 331–341 (2007).

[50] A. Rubinstein. Inapproximability of Nash equilibrium. *Procs. of the 47th STOC*, Portland, USA, pp. 409–418 (2015).

[51] A. Rubinstein. Settling the complexity of computing approximate two-player Nash equilibria. *Procs. of the 57th FOCS*, New Brunswick, USA, pp. 258–265 (2016).

[52] S. Schuldenzucker, S. Seuken, and S. Battiston. Finding clearing payments in financial networks with credit default swaps is PPAD-complete. *Proceedings of the 8th Innovations in Theoretical Computer Science (ITCS) Conference*, Berkeley, USA, (2017).

[53] F.W. Simmons and F.E. Su. Consensus-halving via theorems of Borsuk-Ulam and Tucker. *Mathematical Social Sciences* 45(1) pp. 15–25, (2003).

[54] A. H. Stone and J.W. Tukey. Generalized "sandwich" theorems. *Duke Mathematical Journal*, **9**: pp. 356–359, (1942).

[55] V.V. Vazirani and M. Yannakakis. Market equilibrium under separable, piecewise-linear, concave utilities. *Journal of the ACM*, **58** (3), Article 10 (2011).