Two's Company, Three's a Crowd: Consensus-Halving for a Constant Number of Agents

Argyrios Deligkas
Royal Holloway University of London, United Kingdom
Argyrios.Deligkas@rhul.ac.uk

Aris Filos-Ratsikas
University of Liverpool, United Kingdom
Aris.Filos-Ratsikas@liverpool.ac.uk

Alexandros Hollender
University of Oxford, United Kingdom
Alexandros.Hollender@cs.ox.ac.uk

Abstract

We consider the ε-Consensus-Halving problem, in which a set of heterogeneous agents aim at dividing a continuous resource into two (not necessarily contiguous) portions that all of them simultaneously consider to be of approximately the same value (up to ε). This problem was recently shown to be PPA-complete, for n agents and n cuts, even for very simple valuation functions. In a quest to understand the root of the complexity of the problem, we consider the setting where there is only a *constant* number of agents, and we consider both the *computational complexity* and the *query complexity* of the problem.

For agents with *monotone* valuation functions, we show a dichotomy: for two agents the problem is polynomial-time solvable, whereas for three or more agents it becomes PPA-complete. Similarly, we show that for two monotone agents the problem can be solved with polynomially-many queries, whereas for three or more agents, we provide exponential query complexity lower bounds. These results are enabled via an interesting connection to a *monotone Borsuk-Ulam* problem, which may be of independent interest. For agents with general valuations, we show that the problem is PPA-complete and admits exponential query complexity lower bounds, even for two agents.

1 Introduction

The *Consensus-Halving* problem is a classical problem in fair division whose origins date back to the 1940s and the work of Neyman [1946]. In the approximate version of the problem studied by Simmons and Su [2003], coined the ε-Consensus-Halving problem, a set of n agents with different and heterogeneous valuation functions over the unit interval $[0, 1]$ aim at finding a partition of the interval into pieces labelled either "$+$" or "$-$" using at most n cuts, such that the total value of every agent for the portion labelled "+" and for the portion labelled "−" is approximately the same, up to an additive parameter ε. Very much like other well-known problems in fair division, the existence of a solution to the exact version (for $\varepsilon = 0$) can be proven via fixed-point theorems, whereas the approximate version ($\varepsilon > 0$) admits constructive solutions via exponential-time algorithms [Simmons and Su, 2003]. In fact, the Consensus-Halving problem is a continuous variant of the well-known Necklace Splitting problem [Goldberg and West, 1985; Alon, 1987] with two thieves, and can also be seen as a generalisation of the Hobby-Rice Theorem [Hobby and Rice, 1965].

The ε-Consensus-Halving problem received considerable attention in the literature of computer science over the past few years, as it was proven to be the first "natural" PPA-complete problem [Filos-Ratsikas and Goldberg, 2018], i.e., a problem that does not have a polynomial-sized circuit explicitly in its definition, answering a decade-old open question [Papadimitriou, 1994; Grigni, 2001]. Additionally, Filos-Ratsikas and Goldberg [2019], reduced from this problem to establish the PPA-completeness of Necklace Splitting with two thieves; these PPA-completeness results provided the first definitive evidence of intractability for these two classical problems, establishing for instance that solving them is at least as hard as finding a Nash equilibrium of a strategic game [Daskalakis et al., 2009; Chen et al., 2009]. Filos-Ratsikas et al. [2020] improved on the results for the ε-Consensus-Halving problem, by showing that the problem remains PPA-complete, even if one restricts the attention to very small classes of agents' valuations, namely piecewise-uniform valuations with only two valuation blocks.

This latter result falls under the general umbrella of imposing restrictions on the structure of the problem, to explore if the computational hardness persists or whether we can obtain polynomial-time algorithms. Filos-Ratsikas et al. [2020] applied this approach along the axis of the valuation functions, while considering a general number of agents, similarly to [Filos-Ratsikas and Goldberg, 2018, 2019]. In this paper, we take a different approach, and we restrict the number of agents to be *constant*. This is in line with most of the theoretical work on fair division, which is also concerned with solutions for a small number of agents[1] and it is also quite relevant from a practical standpoint, as fair division among a few participants is quite common. We believe that such investigations are necessary in order to truly understand the computational complexity of the problem. To this end, we state our first main question:

> *What is the computational complexity of* ε-Consensus-Halving *for a constant number of agents?*

Since the number of agents is now fixed, any type of computational hardness must originate from the structure of the valuation functions. We remark that the existence results for ε-Consensus-Halving are fairly general, and in particular do not require assumptions like additivity or monotonicity of the valuation functions. For this reason, the sensible approach is to start from valuations that are as general as possible (for which hardness is easier to establish), and gradually constrain the domain to more specific classes, until eventually polynomial-time solvability becomes possible. Indeed, in a paper that is conceptually similar to ours, Deng et al. [2012] studied the

[1] For example, the existence of bounded protocols for cake-cutting for more than 3 agents was not known until very recently [Aziz and Mackenzie, 2016a,b], but such protocols for 2 and 3 agents were known since the 1960s (see [Brams and Taylor, 1996; Robertson and Webb, 1998]).

computational complexity of the *contiguous envy-free cake-cutting problem*[2] and proved that the problem is PPAD-complete, even for a constant number of agents, when agents have *ordinal preferences* over the possible pieces. These types of preferences induce no structure on the valuation functions and are therefore as general as possible. In contrast, the authors showed that for 3 agents and *monotone valuations*, the problem becomes polynomial-time solvable, leaving the case of 4 or more agents as an open problem. We adopt a similar approach in this paper for ε-Consensus-Halving, and we manage to completely settle the complexity of the problem when the agents have monotone valuations, among other results, which are highlighted in Section 1.1.

Another relevant question that has been surprisingly overlooked in the related literature is the *query complexity* of the problem. In this regime, the algorithm interacts with the agents via *queries*, asking them to provide their values for different parts of the interval $[0, 1]$, and the complexity is measured by the number of queries required to find an ε-approximate solution. This brings us to our second main question:

What is the query complexity of ε-Consensus-Halving for a constant number of agents?

As we will explain in more detail in Section 2, we develop appropriate machinery that allows us to answer both of our main questions at the same time. In a nutshell, for the positive results, we design algorithms that run in polynomial time and can be recast as query-based algorithms that only use a polynomial number of queries. For the negative results, we construct reductions from "hard" computational problems which allow us to simultaneously obtain computational hardness results and query complexity lower bounds.

1.1 Our Results

In this section, we list our main results regarding the computational complexity and the query complexity of the ε-Consensus-Halving problem. We offer more details about the results and our techniques in Section 2.

Computational Complexity: We start from the *computational complexity* of the problem for a constant number of agents. We prove the following main results, parameterised by (a) the number of agents and (b) the structure of the valuation functions.

- For a *single agent* and *general valuations*, the problem is polynomial-time solvable. The same result applies to the case of any number of agents with *identical* general valuations.
- For *two or more agents* and *general valuations*, the problem is PPA-complete.
- For *two agents* and *monotone valuations*, the problem is polynomial-time solvable. This result holds even if one of the two agents has a general valuation.
- For *three or more agents* and *monotone valuations*, the problem is PPA-complete.

Finally, the ε-Consensus-Halving problem with 2 agents coincides with the well-known *ε-Perfect Division* problem for cake-cutting (e.g., see [Brânzei and Nisan, 2017, 2019]), and thus naturally

[2]This version of the classic envy-free cake-cutting problem requires that every agent receives a single, connected piece.

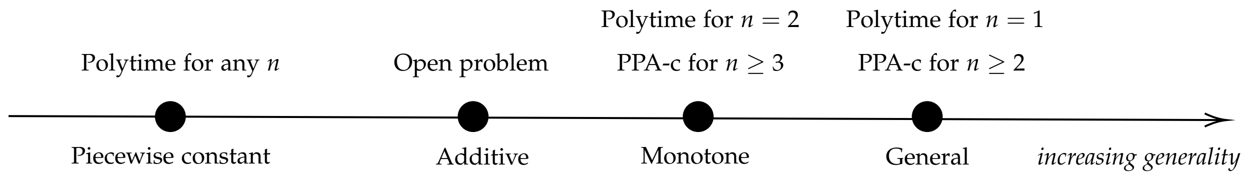

Figure 1: A classification of ε-Consensus-Halving for a constant number n of agents, in terms of increasing generality of the valuation functions.

our results imply that ε-Perfect Division with 2 agents with monotone valuations can be done in polynomial time, whereas it becomes PPA-complete for 2 agents with general valuations.

Before we proceed, we offer a brief discussion on the different cases that are covered by our results. The distinction on the number of agents is straightforward. For the valuation functions, we consider mainly *general valuations* and *monotone valuations*. Note that neither of these functions is additive, meaning that the value that an agent has for the union of two disjoint intervals $[a, b]$ and $[c, d]$ is not necessarily the sum of her values for the two intervals. For monotone valuations, the requirement is that for any two subsets I and I' of $[0, 1]$ such that $I \subseteq I'$, the agent values I' at least as much as I, whereas for general valuations there is no such requirement.

We remark that for agents with piece-wise constant valuations (i.e., the valuations used in [Filos-Ratsikas and Goldberg, 2018] to obtain the PPA-completeness of the problem for many agents), the problem can be solved rather straightforwardly in polynomial time for a constant number of agents, using linear programming (see Lemma A.1 in Appendix A). In terms of the classification of the complexity of the problem in order of increasing generality of the valuation functions, this observation provides the "lower bound" whereas our PPA-hardness results for monotone valuations provide the "upper bound", see Figure 1. While the precise point of the phase transition has not yet been identified, our results make considerable progress towards this goal.

Query Complexity: Besides the computational complexity of the problem, we are also interested in its query complexity. In this setting, one can envision an algorithm which interacts with the agents via a set of queries, and aims to compute a solution to ε-Consensus-Halving using the minimum number of queries possible. In particular, a query is a question from the algorithm to an agent about a subset of $[0, 1]$, who then responds with her value for that set. We provide the following results, where L denotes the Lipschitz parameter of the valuation functions:

- For a *single* agent and *general valuations*, the query complexity of the problem is $\Theta\left(\log \frac{L}{\varepsilon}\right)$. The same result applies for any number of agents with *identical* general valuations.
- For $n \geq 2$ *agents* and *general valuations*, the query complexity of the problem is $\Theta\left(\left(\frac{L}{\varepsilon}\right)^{n-1}\right)$.
- For *two agents* and *monotone valuations*, the query complexity of the problem is $O\left(\log^2 \frac{L}{\varepsilon}\right)$. This result holds even if one of the two agents has a general valuation.
- For $n \geq 3$ *agents* and *monotone valuations*, the query complexity of the problem is between

$\Omega\left(\left(\frac{L}{\varepsilon}\right)^{n-2}\right)$ and $O\left(\left(\frac{L}{\varepsilon}\right)^{n-1}\right)$.

To put these results into context, we remark that when studying the query complexity of the problem, the input consists of the error parameter ε and the Lipschitz parameter L, given by their *binary representation*. In that sense, a $\Theta(\log^k(L/\varepsilon))$ number of queries, for some constant k, is polynomial, whereas a $\Theta(L/\varepsilon)$ number of queries is exponential. Not surprisingly, our PPA-hardness results give rise to exponential query complexity lower bounds, whereas our algorithms can be transformed into query-based algorithms of polynomial query complexity. We remark however that beyond this connection, our query complexity analysis is in fact *quantitative*, as we provide tight or almost tight bounds on the query complexity as a function of the number of agents n, for both general and monotone valuation functions.

Finally, for the case of monotone valuations, we consider a more expressive query model, which is an appropriate extension of the well-known *Robertson-Webb* query model [Robertson and Webb, 1998; Woeginger and Sgall, 2007], the predominant query model in the literature of fair cake-cutting [Brams and Taylor, 1996]; we refer to this extension as the *Generalised Robertson-Webb (GRW)* query model. We show that our bounds extend to this model as well, up to logarithmic factors.

1.2 Related Work

As we mentioned in the introduction, the origins of the Consensus-Halving problem can be traced back to the 1940s and the work of Neyman [1946], who studied a generalisation of the problem with k labels instead of two ("+", "−"), and proved the existence of a solution when the valuation functions are probability measures and there is no constraint on the number of cuts used to obtain the solution. The existence theorem for two labels is known as the *Hobby-Rice Theorem* [Hobby and Rice, 1965] and has been studied rather extensively in the context of the famous *Necklace Splitting problem* [Goldberg and West, 1985; Alon and West, 1986; Alon, 1987]. In fact, most of the proofs of existence for Necklace Splitting (with two thieves) were established via the Consensus-Halving problem, which was at the time referred to as *Continuous Necklace Splitting* [Alon, 1987]. The term "Consensus-Halving" was coined by Simmons and Su [2003], who studied the continuous problem in isolation, and provided a constructive proof of existence which holds for very general valuation functions, including all of the valuation functions that we consider in this paper. Interestingly, the proof of Simmons and Su [2003] reduces the problem to finding edges of complementary labels on a triangulated n-sphere, labelled as prescribed by *Tucker's lemma*, a fundamental result in topology.

While not strictly a reduction in the sense of computational complexity, the ideas of [Simmons and Su, 2003] certainly paved the way for subsequent work on the problem in computer science. The first computational results were obtained by Filos-Ratsikas et al. [2018], who proved that the associated computational problem, ε-Consensus-Halving, lies in PPA (adapting the constructive proof of Simmons and Su [2003]) and that the problem is PPAD-hard, for n agents with piece-wise constant valuation functions. Filos-Ratsikas and Goldberg [2018] proved that the problem is in fact PPA-complete, establishing for the first time the existence of a "natural" problem complete for PPA, i.e., a problem that does not contain a polynomial-sized circuit explicitly in its definition, answering a long-standing open question of Papadimitriou [1994]. In a follow-up paper, [Filos-Ratsikas and Goldberg, 2019] used the PPA-completeness of Consensus-Halving to prove that the Necklace Splitting problem with two thieves is also PPA-complete. Very recently, Filos-Ratsikas et al. [2020] strengthened the PPA-hardness result to the case of very simple valuation functions, namely piece-wise constant valuations with at most two blocks of value. Deligkas et al. [2021] studied the computational complexity of the *exact* version of the problem, and obtained among other results its membership in a newly introduced class BU (for "Borsuk-Ulam" [Borsuk, 1933]) and its computational hardness for the well-known class FIXP of Etessami and Yannakakis [2010].

Importantly, none of the aforementioned results apply to the case of a constant number of agents, which was prior to this paper completely unexplored. Additionally, none of these works consider the query complexity of the problem. A very recent work [Alon and Graur, 2020] studies ε-Consensus-Halving in a hybrid computational model which includes query access to the valuations, but contrary to our paper, their investigations are not targeted towards a constant number of agents, and the agents have additive valuation functions.

A relevant line of work is concerned with the query complexity of *fair cake-cutting* [Brams and Taylor, 1996; Procaccia, 2016], a related but markedly different fair-division problem. Conceptually closest to ours is the paper by Deng et al. [2012], who study both the computational complexity and the query complexity of *contiguous envy-free cake-cutting*, for agents with either general[3] or monotone valuations. For the latter case, the authors obtain a polynomial-time algorithm for three agents, and leave open the complexity of the problem for four or more agents. In our case, for ε-Consensus-Halving, we completely settle the computational complexity of the problem for agents with monotone valuations.

In the literature of fair cake-cutting, most of the related research (e.g., see [Brams and Taylor, 1996; Aziz and Mackenzie, 2016a,b; Amanatidis et al., 2018; Brânzei and Nisan, 2017]) has focused on the well-known *Robertson-Webb (RW)* query model, in which agents interact with the protocol via two types of queries, *evaluation queries* (eval) and *cut* queries (cut). As the name suggests, this query model is due to Robertson and Webb [1995, 1998], but the work of Woeginger and Sgall [2007] has been rather instrumental in formalising it in the form that it is being used today. Given the conceptual similarity of fair cake-cutting with Consensus-Halving, it certainly makes sense to study the latter problem under this query model as well, and in fact, the queries used by Alon and Graur [2020] are essentially RW queries. As we show in Section 7, our bounds are qualitatively robust when migrating to this more expressive model, meaning that they are preserved up to logarithmic factors.

Finally, among the aforementioned works in fair cake-cutting, Brânzei and Nisan [2017] study among other problems, the problem of *ε-Perfect Division*, which stipulates a partition of the cake into n pieces, such that each of the n agents interprets *all* pieces to be of approximate equal value (up to ε). For the case of $n = 2$, this problem coincides with ε-Consensus-Halving, and thus one can interpret our results for $n = 2$ as extensions of the results in [Brânzei and Nisan, 2017] (which are only for additive valuations) to the case of monotone valuations (for which the problem is solvable with polynomially-many queries) and to the case of general valuations (for which the problem admits exponential query complexity lower bounds).

[3]More accurately, Deng et al. [2012] prove their impossibility results for *ordinal preferences*, where for each possible division of the cake, the agent specifies the piece that she prefers. In particular, if one were to define valuation functions consistent with these preferences, the value of an agent for a piece would have to depend also on the way the rest of the cake is divided among the agents.

2 Overview of our Results and Techniques

In this section, we provide a high-level overview of our approach for obtaining our results on the computational complexity and the query complexity of ε-Consensus-Halving. We first describe our main technique and then discuss its implications to each particular case (i.e., agents with general valuations or agents with monotone valuations).[4]

2.1 Input Models

As we mentioned in the Introduction, we will be considering valuations beyond the case of additive functions, namely *monotone* and *general* valuations. For such functions, we need to specify the manner in which they will be accessed by an algorithm for ε-Consensus-Halving. We consider the following two standard models.

- In the ***black-box model***, the valuation functions v_i can be *arbitrary functions*, and are accessed via *queries*. Intuitively, a query to v_i inputs a subset of $[0,1]$ and outputs the non-negative real value of the agent for that set. This input model is appropriate for studying the *query complexity* of the problem, where the complexity is measured as the number of queries to the valuation function v_i.

- In the ***white-box model***, the valuation functions v_i are explicitly given in the input as *polynomial-time algorithms*, mapping subsets of $[0,1]$ (represented as unions of intervals) to non-negative rational numbers. This input model is appropriate for studying the *computational complexity* of the problem, where the complexity is measured as usual by the running time of the algorithm.

2.2 Computational/query complexity results via efficient black-box reductions

At the heart of our techniques lies the concept of *black-box reductions* (see Section 4). Intuitively speaking, a black-box reduction from Problem A to Problem B is a procedure which

- Translates a solution for Problem B to a solution for Problem A.
- Uses *oracle calls* (queries) for Problem B which contain as a subroutine oracle calls to the instance of Problem A.

This type of reduction enables us to obtain query complexity upper and lower bounds for a "target" problem in the black-box model, via reductions from and to some known computational problems for which such query complexity bounds are known. To obtain computational complexity results in the white-box model, we can use the same reduction, as long as the computation performed within the procedure requires polynomial-time, and the number of oracle calls to the instance of Problem B for each oracle call to the instance of Problem A is bounded by a polynomial.

More precisely, we will call a black-box reduction *efficient*, if it runs in polynomial time and performs only a *constant* number of oracle calls to the instance of the problem that we reduce from, for each oracle call of the "target" problem. Using these reductions, we will be able to

[4] To avoid having the reader refer back to this section, similar high-level descriptions are also present in the corresponding sections.

obtain PPA-membership results/query complexity upper bounds and PPA-hardness results/query complexity lower bounds for ε-Consensus-Halving, via the same set of reductions respectively.

The starting point for our results will be the computational problem nD-Tucker (Definition 4). This problem is the computational version of Tucker's Lemma [Tucker, 1945], and for which the following results (stated informally below) are known.

Informal Theorem 1 ([Aisenberg et al., 2020; Papadimitriou, 1994], see Theorem 3.1)**.** *For $n \geq 2$, nD-Tucker is PPA-complete.*

Informal Theorem 2 ([Deng et al., 2011], see Theorem 3.2)**.** *For $n \geq 2$, nD-Tucker admits exponential query complexity lower and upper bounds.*

According to our discussion above, if we constructed efficient black-box reductions from ε-Consensus-Halving to nD-Tucker and vice-versa, we would obtain the desired results for the former problem. Indeed, in Section 4.2, we present such a reduction from ε-Consensus-Halving to nD-Tucker, which enables us to obtain that (a) ε-Consensus-Halving with at least 2 agents with general valuations is in PPA and (b) that the problem can be solved with a number of queries which is of the same order as the upper bound of the query complexity of nD-Tucker.

For the other direction, we will obtain the reduction via the use of an *intermediate problem*, the nD-Borsuk-Ulam problem, which is the computational version of the Borsuk-Ulam Theorem [Borsuk, 1933]. More precisely, in Section 4.1, we construct a general efficient black-box reduction from nD-Borsuk-Ulam to ε-Consensus-Halving with a constant number n of agents. This reduction has the special feature that *it preserves a certain set of properties*, namely (see Definition 3)

- If the function F of nD-Borsuk-Ulam is *normalised*, then the valuation functions of ε-Consensus-Halving are normalised.

- If the function F of nD-Borsuk-Ulam is *monotone*, then the valuation functions of ε-Consensus-Halving are monotone.

The first of those properties will allow us to show our PPA-hardness results and query complexity lower bounds for ε-Consensus-Halving, even in the case where the valuation functions must satisfy a standard normalisation condition, making them stronger. The latter property is fundamental in establishing our impossibility results for the case of ε-Consensus-Halving, when agents have monotone valuation functions. In particular, the property gives rise to a *monotone Borsuk-Ulam* computational problem, which rather naturally corresponds to the Consensus-Halving problem when agents have monotone valuations, and could be of independent interest. Our general reduction in Section 4.1 thus allows us to reduce the problem of proving impossibility results for ε-Consensus-Halving to the task of proving such results of nD-Borsuk-Ulam and monotone nD-Borsuk-Ulam. These results will be established via efficient black-box reductions from nD-Tucker, in Section 5 and Section 6, respectively.

2.3 Agents with general valuations

Using our general approach described in the previous section, we can prove the following theorem, stated informally.

Informal Theorem 3 (see Theorem 5.6, Theorem 5.8)**.** *For any constant $n \geq 2$, ε-*Consensus-Halving *with n agents and general valuations is PPA-complete, and admits asymptotically matching exponential query complexity upper and lower bounds. This result holds even if the valuations are normalised.*

To obtain this result, in Section 5.1, we construct an efficient black-box reduction from nD-Tucker to nD-Borsuk-Ulam, thus inheriting both its computational hardness and its query complexity lower bounds. On a high-level, we interpolate the nD-Tucker instance in order to obtain a continuous function, and then embed it in the nD-Borsuk-Ulam domain. Some extra care must be taken to ensure that we obtain a *normalised* nD-Borsuk-Ulam instance.

The only case left uncovered by the theorem above is the case of a single agent with a general valuation function. We observe that a simple binary search-like procedure can find a solution to ε-Consensus-Halving in this case both in polynomial time, and using a polynomial number of queries (see Theorem 5.1). The result trivially extends to the case of multiple agents with identical general valuations.

2.4 Agents with monotone valuations

For agents with monotone valuations, again we employ our general approach, but this time we need to prove computational and query-complexity hardness of the monotone nD-Borsuk-Ulam problem; the corresponding impossibility results for ε-Consensus-Halving with agents with monotone valuations then follow from our property-preserving reduction in Section 4. To this end, we in fact construct an efficient black-box reduction from $(n-1)D$-Tucker to monotone nD-Borsuk-Ulam, i.e., we reduce from the corresponding version of nD-Tucker of one lower dimension. In order to achieve this, we once again interpolate the $(n-1)D$-Tucker instance to obtain a continuous function, but, this time, we embed it in a very specific lower dimensional subset of the nD-Borsuk-Ulam domain. We then show that the function can be extended to a monotone function on the whole domain.

The "drop in dimension" which is featured in our reduction has the following effects:

- Since $1D$-Tucker is solvable in polynomial-time, we can only obtain the PPA-hardness of monotone nD-Borsuk-Ulam for $n \geq 3$, and therefore the PPA-hardness of ε-Consensus-Halving for three or more monotone agents.
- The query complexity lower bounds that we "inherit" from $(n-1)D$-Tucker do not *exactly* match our upper bounds, obtained via the reduction from ε-Consensus-Halving to nD-Tucker in Section 4.2.

We have the following theorem, stated informally.

Informal Theorem 4 (see Theorem 6.6, Theorem 6.8)**.** *For any constant $n \geq 3$, ε-*Consensus-Halving *with n agents with monotone valuations is PPA-complete, and admits* almost *asymptotically matching exponential query complexity upper and lower bounds. This result holds even if the valuations are normalised.*

It turns out that the inability of our reduction to give us hardness for 2 agents in the monotone case is not a coincidence. In Section 6, we construct a polynomial-time algorithm for solving the problem when $n = 2$ in the white-box model, which can also be interpreted as an algorithm of polynomial query complexity in the black-box model. We state the corresponding theorem informally.

Informal Theorem 5 (Theorem 6.1)**.** *ε-*Consensus-Halving *with two agents and monotone valuation functions is solvable in polynomial time and has polynomial query complexity.*

At a high level, the algorithm (see Algorithm 1) performs a *nested binary search* procedure. It starts with an interval, initially defined by the positions of the two cuts that "satisfy" the first agent, when the leftmost cut sits on the left endpoint 0 of the interval. Then, the algorithm "shrinks" this interval based on whether the other agent "sees" more "+" or "−" by this set of cuts, maintaining the following invariant: (a) the first agent is always satisfied and (b) the solution that satisfies the other agent is contained in the considered interval that is being shrunk. Intuitively, the algorithm is moving on the "indifference curve" with respect to the valuation of the first agent, it "offers" the other agent points (i.e., sets of cuts), to which the agent replies with either "+" or "−", indicating the label of her discrepancy. Using the answers of the second agent, we can move in the appropriate direction whilst shrinking the interval in question.

2.5 Query Complexity in the Generalised Robertson-Webb model

Finally, we establish a connection between our results and the well-known *Robertson-Webb (RW)* query model for fair cake-cutting [Robertson and Webb, 1998; Woeginger and Sgall, 2007]. In this model, the algorithm interacts with the agents via *evaluation* (eval) and *cut* (cut) queries, and these queries have only been defined for agents with additive valuation functions.

Our first contribution in this section is to define an appropriate generalisation of the RW model that can be used for agents with monotone valuations, which we refer to as the *Generalised Robertson Webb (GRW) query model*. It turns out that the straightforward generalisation of eval queries is in fact the type of queries that we use in our black-box model defined above. Therefore, the GRW model is a stronger query model, equipped with—what we consider to be—the most natural generalisation of cut queries. Our main result in this section is that the query lower bounds in this more expressive query model are *qualitatively* the same, as stated informally in the following theorem.

Informal Theorem 6 (see Theorem 7.1)**.** *ε-*Consensus-Halving *with $n \geq 3$ agents with monotone valuations requires an exponential number of queries, even in the GRW model.*

Using a straightforward binary search approach, it is possible to simulate an *approximate* cut query using a logarithmic number of eval queries. In order to obtain our stronger result, which holds even for *exact* cut queries, we construct "hard" *nD*-Borsuk-Ulam instances that are also piece-wise linear. We then show that in the resulting ε-Consensus-Halving instances, cut queries can be answered exactly by performing a logarithmic number of eval queries instead.

3 Preliminaries

In the ε-Consensus-Halving problem, there is a set of n agents with *valuation functions v_i* (or simply *valuations*) over the interval $[0, 1]$, and the goal is to find a partition of the interval into subintervals labelled either "+" or "−", using at most n cuts. This partition should satisfy that for every agent i, the total value for the union of subintervals $\mathcal{I}^+$ labelled "$+$" and the total value for the union of subintervals $\mathcal{I}^-$ labelled "$-$" is the same up to ε, i.e., $|v_i(\mathcal{I}^+) - v_i(\mathcal{I}^-)| \leq \varepsilon$. In this paper we will assume n to be a constant and therefore the inputs to the problem will only be ε and the valuation functions v_i.

We will be interested in fairly general valuation functions; intuitively, these will be functions mapping *measurable subsets* $A \subseteq [0,1]$ to non-negative real numbers. Formally, let $\Lambda([0,1])$ denote the set of Lebesgue-measurable subsets of the interval $[0,1]$ and $\lambda : \Lambda([0,1]) \to [0,1]$ the Lebesgue measure. We consider valuation functions $v_i : \Lambda([0,1]) \to \mathbb{R}_{\geq 0}$, with the interpretation that agent i has value $v_i(A)$ for the subset $A \in \Lambda([0,1])$ of the resource. Similarly to [Deng et al., 2012; Brânzei and Nisan, 2017, 2019; Barman and Rathi, 2020], we also require that the valuation functions be Lipschitz-continuous. Following [Deng et al., 2012], a valuation function v_i is said to be Lipschitz-continuous with Lipschitz parameter $L \geq 0$, if for all $A, B \in \Lambda([0,1])$, it holds that $|v_i(A) - v_i(B)| \leq L \cdot \lambda(A \triangle B)$. Here $\triangle$ denotes the symmetric difference, i.e., $A \triangle B = (A \setminus B) \cup (B \setminus A)$.

Valuation Classes: We will be particularly interested in the following three valuation classes, in terms of decreasing generality:

- *General valuations*, in which there is no further restriction to the functions v_i.
- *Monotone valuations*, in which $v_i(A) \geq v_i(A')$ for any two Lebesgue-measurable subsets A and A' such that $A' \subseteq A$. Intuitively, for this type of functions, when comparing two sets of intervals such that one is a subset of the other, an agent can not have a smaller value for the set that contains the other.
- *Additive valuations*, in which v_i is a function from individual intervals in $[0,1]$ to $\mathbb{R}_{\geq 0}$ and for a set of intervals $\mathcal{I}$, it holds that $v_i(\mathcal{I}) = \sum_{I \in \mathcal{I}} v_i(I)$. Note that if v_i is an additive valuation function, then it is in fact a measure. We will not prove any results for this type of valuation functions in this paper, but we define them for reference and comparison (e.g., see Figure 1).

Normalisation: We will also be interested in valuation functions that satisfy some standard normalisation properties. We say that a valuation function v_i is *normalised*, if the following properties hold:

1. $v_i(A) \in [0,1]$ for all $A \in \Lambda([0,1])$,
2. $v_i(\emptyset) = 0$ and $v_i([0,1]) = 1$.

In other words, we require the agents' values to lie in $[0,1]$ and that their value for the whole interval is normalised to 1. These are the standard assumptions in the literature of the problem for additive valuations [Alon, 1987], as well as in the related problem of fair cake-cutting [Procaccia, 2016]. We will only consider normalised valuation functions for our lower bounds and hardness results, whereas for the upper bounds and polynomial-time algorithms we will not impose any normalisation; this only makes both sets of results even stronger.

With regard to the valuation classes defined above, we will often be referring to their normalised versions as well, e.g., *normalised general valuations* or *normalised monotone valuations*.

Input models: Given the fairly general nature of the valuation functions, we need to specify the manner in which they will be accessed by an algorithm for ε-Consensus-Halving. Since we are interested in both the computational complexity and the query complexity of the problem, we will assume the following standard two ways of accessing these functions.

- In the ***black-box model***, the valuation functions v_i can be *arbitrary functions*, and are accessed via *queries* (sometimes also referred to as *oracle calls*). A query to the function v_i inputs a Lebesgue-measurable subset A (intuitively a set of subintervals) of $[0,1]$ and outputs $v_i(A) \in \mathbb{R}_{\geq 0}$. This input model is appropriate for studying the *query complexity* of the problem, where the complexity is measured as the number of queries to the valuation function v_i.

  We will also consider the following weaker version of the black-box model, which we will use in our query complexity upper bounds, thus making them stronger: In the ***weak black-box model*** the input to a valuation function v_i is some set $\mathcal{I}$ of intervals, obtained by using at most n cuts, where n is the number of agents.

- In the ***white-box model***, the valuation functions v_i are *polynomial-time algorithms*, mapping sets of intervals to non-negative rational numbers. These polynomial-time algorithms are given explicitly as part of the input, including the Lipschitz parameter L.[5] This input model is appropriate for studying the *computational complexity* of the problem, where the complexity is measured as usual by the running time of the algorithm.

We provide the formal definitions of the problem in the black-box model and the white-box model below.

Definition 1 (ε-Consensus-Halving *(black-box model)*)**.** For any constant $n \geq 1$, the problem ε-Consensus-Halving *with n agents* is defined as follows:

- **Input:** $\varepsilon > 0$, the Lipschitz parameter L, *query access* to the functions v_i.
- **Output:** A partition of $[0,1]$ into two sets of intervals $\mathcal{I}^+$ and $\mathcal{I}^-$ such that for each agent i, it holds that $|v_i(\mathcal{I}^+) - v_i(\mathcal{I}^-)| \leq \varepsilon$, using at most n cuts.

Definition 2 (ε-Consensus-Halving *(white-box model)*)**.** For any constant $n \geq 1$, the problem ε-Consensus-Halving *with n agents* is defined as follows:

- **Input:** $\varepsilon > 0$, the Lipschitz parameter L, polynomial-time algorithms v_i.
- **Output:** A partition of $[0,1]$ into two sets of intervals $\mathcal{I}^+$ and $\mathcal{I}^-$ such that for each agent i, it holds that $|v_i(\mathcal{I}^+) - v_i(\mathcal{I}^-)| \leq \varepsilon$, using at most n cuts.

Terminology: When the valuation functions are normalised, we will refer to the problem as ε-Consensus-Halving *with n normalised agents*. When the valuation functions are monotone, we

[5] To avoid introducing too many technical details here, we refer the reader to Appendix B for the fully formal definition. We also remark that some related work [Brânzei and Nisan, 2017; Alon and Graur, 2020] takes L to be bounded by a constant, and thus it does not appear in their bounds.

will refer to the problem as ε-Consensus-Halving *with n monotone agents*. If both conditions are true, we will use the term ε-Consensus-Halving *with n normalised monotone agents*.

3.1 Borsuk-Ulam and Tucker

For our PPA-hardness results and query complexity lower bounds, we will reduce from a well-known problem, the computational version of the Borsuk-Ulam Theorem [Borsuk, 1933], which states that for any continuous function F from S^n to $\mathbb{R}^n$ there is a pair of antipodal points (i.e., $x, -x$) which are mapped to the same point. There are various equivalent versions of the problem (e.g., see [Matoušek, 2008]); we will provide a definition that is most appropriate for our purposes. In fact, we will include several "optional" properties of the function F in our definition, which will map to properties of the valuation functions v_i when we construct our reductions in subsequent sections. Specifically, we will impose conditions for *normalisation* and *monotonicity*, which will correspond to normalised valuation functions for our lower bounds/hardness results of Section 5, and to normalised monotone valuation functions for our lower bounds/hardness results of Section 6. Let $B^n = [-1,1]^n$ and let $\partial(B^n)$ denote its boundary. As before, we will require that the functions we consider be Lipschitz-continuous. We say that $F : B^{n+1} \to B^n$ is Lipschitz-continuous with parameter L, if $\|F(x) - F(y)\|_\infty \leq L \cdot \|x - y\|_\infty$ for all $x, y \in B^{n+1}$.

Definition 3 (nD-Borsuk-Ulam)**.** For any constant $n \geq 1$, the problem nD-Borsuk-Ulam is defined as follows:

- **Input:** $\varepsilon > 0$, the Lipschitz parameter L, a function $F : B^{n+1} \to B^n$.
- **Output:** A point $x \in \partial(B^{n+1})$ such that $\|F(x) - F(-x)\|_\infty \leq \varepsilon$.
- **Optional Properties:**

  Normalisation:
  - $F(1, 1, \ldots, 1) = (1, 1, \ldots, 1)$.
  - $F(-x) = -F(x)$, for all $x \in B^{n+1}$.

  Monotonicity:
  - If $x \leq y$, then $F(x) \leq F(y)$, for all $x, y \in B^{n+1}$, where "$\leq$" denotes coordinate-wise comparison.

In the *normalised* nD-Borsuk-Ulam *problem*, where F is normalised, we instead ask for a point $x \in \partial(B^{n+1})$ such that $\|F(x)\|_\infty \leq \varepsilon$. By using the fact that F is an odd function ($F(-x) = -F(x)$, for all $x \in B^{n+1}$), it is easy to see that this is equivalent to $\|F(x) - F(-x)\|_\infty \leq \varepsilon/2$. We will also use the term *normalised monotone* nD-Borsuk-Ulam to refer to the problem when both the normalisation and monotonicity properties are satisfied for F.

In the black-box version of nD-Borsuk-Ulam, we can query the value of the function F at any point $x \in B^{n+1}$. In the white-box version of this problem, we are given a polynomial-time algorithm that computes F. Since the number of inputs of F is fixed, we can assume that we are given an arithmetic circuit with $n+1$ inputs and n outputs that computes F. Following the related literature [Daskalakis and Papadimitriou, 2011], we will consider circuits that use the arithmetic gates $+, -, \times, \max, \min, <$ and rational constants.[6]

Another related problem that will be of interest to us is the computational version of Tucker's Lemma [Tucker, 1945]. Tucker's lemma is a discrete analogue of the Borsuk-Ulam theorem, and its computational counterpart, nD-Tucker, is defined below.

Definition 4 (nD-Tucker)**.** For any constant $n \geq 1$, the problem nD-Tucker is defined as follows:

- **Input:** grid size $N \geq 2$, a labelling function $\ell : [N]^n \to \{\pm 1, \pm 2, \ldots, \pm n\}$ that is antipodally anti-symmetric (i.e., for any point p on the boundary of $[N]^n$, we have $\ell(\overline{p}) = -\ell(p)$, where $\overline{p}_i = N + 1 - p_i$ for all i).
- **Output:** Two points $p, q \in [N]^n$ with $\ell(p) = -\ell(q)$ and $\|p - q\|_\infty \leq 1$.

In the black-box version of this problem, we can query the labelling function for any point $p \in [N]^n$ and retrieve its label. In the white-box version, ℓ is given in the form of a Boolean circuit with the usual gates $\wedge, \vee, \neg$.

In the white-box model, nD-Tucker was recently proven to be PPA-hard for any $n \geq 2$, by Aisenberg et al. [2020]; the membership of the problem in PPA was known by [Papadimitriou, 1994].

Theorem 3.1 (Aisenberg et al. [2020]; Papadimitriou [1994])**.** *For any constant $n \geq 2$, nD-Tucker is PPA-complete.*

The computational class PPA was defined by Papadimitriou [1994], among several subclasses of the class TFNP [Megiddo and Papadimitriou, 1991], the class of problems with a guaranteed solution which is verifiable in polynomial time. PPA is defined with respect to a graph of exponential size, which is given *implicitly* as input, via the use of a circuit that outputs the neighbours of a given vertex, and the goal is to find a vertex of degree 1, given another such vertex as input.

Given the close connection between Tucker's Lemma and the Borsuk-Ulam Theorem, the PPA-hardness of *some appropriate computational version* of Borsuk-Ulam follows as well [Aisenberg et al., 2020]. However, this does not apply to the version of nD-Borsuk-Ulam defined above, especially when one considers the additional properties of the function F required for normalisation and monotonicity, as discussed earlier. We will reduce from nD-Tucker to our version of nD-Borsuk-Ulam, to obtain its PPA-hardness, which will then imply the PPA-hardness of ε-Consensus-Halving, via our main reduction in Section 4.

In the black-box model, Deng et al. [2011], building on the results of Chen and Deng [2008], proved both query complexity lower bounds and upper bounds for nD-Tucker.

Theorem 3.2 (Deng et al. [2011])**.** *For any constant $n \geq 2$, the query complexity of nD-Tucker is $\Theta\left(N^{n-1}\right)$.*

[6]See [Etessami and Yannakakis, 2010] for more details on arithmetic circuits for fixed point-related problems.

The reductions that we will construct (from nD-Tucker to nD-Borsuk-Ulam to ε-Consensus-Halving) will be *black-box reductions* (see Section 4), and therefore they will also allow us to obtain query complexity lower bounds for ε-Consensus-Halving in the black-box model, given the corresponding lower bounds of Theorem 3.2. For the upper bounds, we will reduce directly from ε-Consensus-Halving to nD-Tucker, again via a black-box reduction. We remark that Deng et al. [2011] use a version of nD-Tucker that is slightly different from the one that we defined above, but their results apply to this version as well.

4 Black-box reductions to and from Consensus-Halving

In this section we develop our main machinery for proving both PPA-completeness results and query complexity upper and lower bounds for ε-Consensus-Halving. Our techniques will be based on the standard notion of *black-box reductions*.[7] Roughly speaking, a black-box reduction from Problem A to Problem B is a procedure by which we can answer oracle calls (queries) for an instance of Problem B by using an oracle for some instance of Problem A, such that a solution to the instance of Problem B yields a solution to the instance of Problem A. For example, a black-box reduction from nD-Borsuk-Ulam to ε-Consensus-Halving is a procedure that solves the latter problem by accessing the function F of the former problem a number of times, and then translates the solution of ε-Consensus-Halving to a solution of nD-Borsuk-Ulam. The name "black-box" comes from the fact that this type of reduction does not need to know the structure of the functions v_i of ε-Consensus-Halving or F of nD-Borsuk-Ulam.

In order to prove lower bounds on the query complexity of some Problem B, it suffices to construct a black-box reduction from some Problem A, for which query complexity lower bounds are known; the obtained bounds will depend on the number k of oracle calls to the input of Problem A that are needed to answer an oracle call to the input of Problem B. We will say that a black-box reduction is *efficient* if k is a constant, and therefore the query complexity lower bounds of Problems A and B are of the same asymptotic order. To obtain upper bounds on the query complexity, we can construct a reduction in the opposite direction (from Problem B to Problem A), assuming that query complexity upper bounds for Problem A are known.

Ideally, we would like to use the same reduction to also obtain computational complexity results in the white-box model. For this to be possible, the procedure described above should actually be a polynomial-time algorithm. Slightly abusing terminology, we will use the term "efficient" to describe such a reduction in the white-box model as well. To this end, we have the following definition.

Definition 5. We will say that a black-box reduction from Problem A to Problem B is *efficient* if:

- in the black-box model, it uses a constant number of queries (oracle calls) to the function (oracle) of Problem A, for each query (oracle call) to the function of Problem B;
- in the white-box model, the condition above holds, and the reduction is also a polynomial-time algorithm.

Concretely for our case, all of our reductions will be efficient black-box reductions, thus allowing us to obtain both PPA-completeness results and query complexity bounds matching those of the

[7] To keep the exposition clean, we do not provide formal definitions of these concepts here, as they are rather standard; we refer the reader to the related works of [Beame et al., 1998; Komargodski et al., 2019] for more details.

problems that we reduce from/to. We remark that the reductions constructed for proving the PPA-hardness of the problem in previous works (for a non-constant number of agents) [Filos-Ratsikas and Goldberg, 2018, 2019; Filos-Ratsikas et al., 2020] are not black-box reductions, and therefore have no implications on the query complexity of the problem.

We summarise our general approach for obtaining positive and negative results below.

- For our *impossibility results* (i.e., computational hardness results in the white-box model and query complexity lower bounds in the black-box model), we will construct an efficient black-box reduction from nD-Borsuk-Ulam to ε-Consensus-Halving with n agents (Section 4.1). This reduction will preserve the optional properties of Definition 3, meaning that if the instance of nD-Borsuk-Ulam is normalised (respectively monotone), the valuation functions of the corresponding instance of ε-Consensus-Halving will be normalised (respectively monotone) as well. This will allow us in subsequent sections to reduce the problem of proving impossibility results for ε-Consensus-Halving to proving impossibility results for the versions of nD-Borsuk-Ulam with those properties. We will obtain these latter results via reductions from nD-Tucker, which for $n \geq 2$ is known to be PPA-hard (Theorem 3.1) and admit exponential query complexity lower bounds (Theorem 3.2).
- For our *positive results* (i.e., membership in PPA in the white-box model and query complexity upper bounds in the black-box model), we will construct an efficient black-box reduction from ε-Consensus-Halving to nD-Tucker (Section Section 4.2). We remark here that a similar reduction already exists in the related literature [Filos-Ratsikas et al., 2021], but only applied to the case of additive valuation functions. The extension to the case of general valuations follows along the same lines, and we provide it here for completeness. We also note that some of our positive results, namely the results for one general agent and two monotone agents, will not be obtained via reductions, but rather directly via the design of polynomial-time algorithms in the white-box model or algorithms of polynomial query complexity in the black-box model.

4.1 From Borsuk-Ulam to Consensus-Halving

In this section, we present an efficient black-box reduction from nD-Borsuk-Ulam to ε-Consensus-Halving with n agents. As we mentioned earlier, this reduction will preserve the properties of the Borsuk-Ulam function (normalisation and/or monotonicity), so that the resulting instance of ε-Consensus-Halving will exhibit valuations with those same properties.

Description of the reduction. Let $n \geq 1$ be a fixed constant. Let $\varepsilon > 0$ and let $F : B^{n+1} \to B^n$ be a Lipschitz-continuous function with Lipschitz parameter L. We now construct valuation functions $v_1, \dots, v_n$ for a Consensus-Halving instance.

Let $R_1, R_2, \dots, R_{n+1}$ denote the partition of interval $[0,1]$ into $n+1$ subintervals of equal length, i.e., $R_j = [\frac{j-1}{n+1}, \frac{j}{n+1}]$ for $j \in [n+1]$. For any $A \in \Lambda([0,1])$, we define $x(A) \in B^{n+1} = [-1,1]^{n+1}$ by

$$[x(A)]_j = 2(n+1) \cdot \lambda(A \cap R_j) - 1$$

for all $j \in [n+1]$. Recall that λ denotes the Lebesgue measure on the interval $[0,1]$. Note that since $\lambda(A \cap R_j) \in [0, \frac{1}{n+1}]$, we indeed have $[x(A)]_j \in [-1,1]$; see Figure 2 for a visualisation.

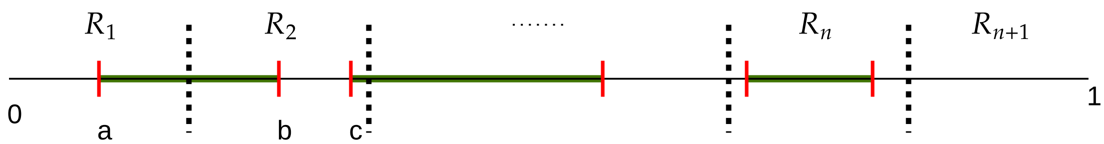

Figure 2: The partition of $[0,1]$ into $n+1$ subintervals of equal length and a set A, coloured by the green region, as it is defined by the red cuts. The first three cuts on the left are located at positions $a \leq b \leq c$, where $a \in R_1$ and $b, c \in R_2$. Here, since $\lambda(A \cap R_1) = 1/(n+1) - a$, we would obtain $[x(A)]_1 = 2(n+1)(1/(n+1) - a) - 1 = 1 - 2(n+1)a$. Similarly, $\lambda(A \cap R_2) = 1/(n+1) - (c-b)$, and thus $[x(A)]_2 = 1 - 2(n+1)(c-b)$.

For $i \in [n]$, the valuation function v_i of the ith agent is defined as

$$v_i(A) = \frac{F_i(x(A)) + 1}{2}$$

for any $A \in \Lambda([0,1])$, where $F_i : B^{n+1} \to [-1,1]$ is the ith output of F. Note that $v_i(A) \in [0,1]$, since $F_i(x(A)) \in [-1,1]$.

Lipschitz-continuity. For any $A, B \in \Lambda([0,1])$ it holds that

$$\begin{aligned}
|v_i(A) - v_i(B)| = \frac{1}{2}\,|F_i(x(A)) - F_i(x(B))| &\leq \frac{1}{2}\|F(x(A)) - F(x(B))\|_\infty \\
&\leq \frac{L}{2}\|x(A) - x(B)\|_\infty \\
&\leq (n+1)L \cdot \max_{j\in[n+1]} |\lambda(A \cap R_j) - \lambda(B \cap R_j)| \\
&\leq (n+1)L \cdot \max_{j\in[n+1]} \lambda((A \triangle B) \cap R_j) \\
&\leq (n+1)L \cdot \lambda(A \triangle B)
\end{aligned}$$

Thus, v_i is Lipschitz-continuous with Lipschitz parameter $(n+1)L$.

Correctness. Now consider an $\varepsilon/2$-Consensus-Halving of $v_1, \ldots, v_n$. Namely, let $\mathcal{I}^+, \mathcal{I}^-$ be a partition of $[0,1]$ using at most n cuts, such that $|v_i(\mathcal{I}^+) - v_i(\mathcal{I}^-)| \leq \varepsilon/2$ for all $i \in [n]$. Since $\mathcal{I}^+$ is obtained by using at most n cuts, it follows that there exists $j \in [n+1]$ such that R_j does not contain a cut. As a result, $\mathcal{I}^+ \cap R_j$ is either empty or equal to R_j. This implies that $\lambda(\mathcal{I}^+ \cap R_j) \in \{0, \frac{1}{n+1}\}$ and thus $[x(\mathcal{I}^+)]_j \in \{\pm 1\}$, i.e., $x(\mathcal{I}^+) \in \partial(B^{n+1})$. Furthermore, for any $j \in [n+1]$, we have

$$\begin{aligned}
[x(\mathcal{I}^+)]_j = 2(n+1) \cdot \lambda(\mathcal{I}^+ \cap R_j) - 1 &= 2(n+1) \cdot \left(\frac{1}{n+1} - \lambda(\mathcal{I}^- \cap R_j)\right) - 1 \\
&= -2(n+1) \cdot \lambda(\mathcal{I}^- \cap R_j) + 1 \\
&= -[x(\mathcal{I}^-)]_j.
\end{aligned}$$

Letting $y = x(\mathcal{I}^+) \in \partial(B^{n+1})$, we have that for any $i \in [n]$

$$|F_i(y) - F_i(-y)| = |F_i(x(\mathcal{I}^+)) - F_i(x(\mathcal{I}^-))| = 2|v_i(\mathcal{I}^+) - v_i(\mathcal{I}^-)| \leq \varepsilon.$$

Thus, y is a solution to the original nD-Borsuk-Ulam instance.

White-box model. This reduction yields a polynomial-time many-one reduction from nD-Borsuk-Ulam to ε-Consensus-Halving with n agents. Thus, if we show that nD-Borsuk-Ulam is PPA-hard for some n, then we immediately obtain that ε-Consensus-Halving with n agents is also PPA-hard.

Black-box model. It is easy to see that this is a black-box reduction. It can be formulated as follows: given access to an oracle for an instance of nD-Borsuk-Ulam with parameters (ε, L) we can simulate an oracle for an instance of ε-Consensus-Halving (with n agents) with parameters $(\varepsilon/2, (n+1)L)$ such that any solution of the latter yields a solution to the former. Furthermore, in order to answer a query to some v_i, we only need to perform a single query to F. Thus, we obtain the following query lower bound: solving an instance of ε-Consensus-Halving (with n agents) with parameters (ε, L) requires at least as many queries as solving an instance of nD-Borsuk-Ulam with parameters $(\varepsilon', L') = (2\varepsilon, \frac{L}{n+1})$. This means that if nD-Borsuk-Ulam has a query lower bound of $\Omega((\frac{L'}{\varepsilon'})^{n-1})$ for some n, then ε-Consensus-Halving (with n agents) has a query lower bound of $\Omega((\frac{L}{2\varepsilon(n+1)})^{n-1}) = \Omega((\frac{L}{\varepsilon})^{n-1})$, since n is constant.

Additional properties of the reduction. Some properties of the Borsuk-Ulam function F carry over to the valuation functions $v_1, \dots, v_n$. In particular, the following properties are of interest to us:

- **If F is monotone, then $v_1, \dots, v_n$ are monotone.** Indeed, consider $A, B \in \Lambda([0,1])$ with $A \subseteq B$. Then, it holds that $\lambda(A \cap R_j) \leq \lambda(B \cap R_j)$ for all $j \in [n+1]$, and as a result $x(A) \leq x(B)$ (coordinate-wise). By monotonicity of F, it follows that $F_i(x(A)) \leq F_i(x(B))$, and thus $v_i(A) \leq v_i(B)$ for all $i \in [n]$.

- **If F is normalised, then $v_1, \dots, v_n$ are normalised.** As noted earlier, we already have that $v_i(A) \in [0,1]$ for all $A \in \Lambda([0,1])$. Thus, it remains to prove that $v_i(\emptyset) = 0$ and $v_i([0,1]) = 1$. It is easy to see that $x([0,1]) = (1,1,\dots,1)$ and thus $F(x([0,1])) = (1,1,\dots,1)$ since F is normalised, which yields $v_i([0,1]) = 1$. On the other hand, we have $x(\emptyset) = (-1,-1,\dots,-1)$ and thus $F(x(\emptyset)) = -F(-x(\emptyset)) = -F(1,1,\dots,1) = (-1,-1,\dots,-1)$, which yields $v_i(\emptyset) = 0$. Here we also used the fact F is an odd function, since it is normalised. In fact, since F is odd, we also obtain that $v_i(A) + v_i(A^c) = 1$ for all $A \in \Lambda([0,1])$, where $A^c = [0,1] \setminus A$ denotes the complement of A. This can be shown by noting that $x(A^c) = -x(A)$ (by using the same argument as for $\mathcal{I}^+$ and $\mathcal{I}^-$ above) and then using the fact that $F(x(A^c)) = -F(x(A))$.

This means that if we are able to show (white- or black-box) hardness of nD-Borsuk-Ulam where F has additional properties, then the hardness will also hold for ε-Consensus-Halving with n agents that have the corresponding properties.

Furthermore, note that if F is a normalised nD-Borsuk-Ulam function, then an ε-approximate Consensus-Halving for $v_1, \ldots, v_n$ (i.e., $|v_i(\mathcal{I}^+) - v_i(\mathcal{I}^-)| \leq \varepsilon$), yields an ε-approximate solution to F, in the sense that $\|F(x(\mathcal{I}^+))\|_\infty \leq \varepsilon$. This is due to the fact that, by definition, F is an odd function, if it is normalised.

4.2 From Consensus-Halving to Tucker

The idea behind this reduction is the same as in [Filos-Ratsikas et al., 2021], where the result was only proved for additive valuations. We include the full reduction here for completeness.

Description of the reduction. Consider an instance of ε-Consensus-Halving with n agents with parameters ε, L. Let $v_1, \ldots, v_n$ denote the valuations of the agents. We consider the domain K_m^n, where $K_m = \{-1, -(m-1)/m, \ldots, -1/m, 0, 1/m, 2/m, \ldots, (m-1)/m, 1\}$, for $m = \lceil 2nL/\varepsilon \rceil$. A point in K_m^n corresponds to a way to partition the interval $[0, 1]$ into two sets $\mathcal{I}^+, \mathcal{I}^-$ using at most n cuts. A very similar encoding was also used by Meunier [2014] for the Necklace Splitting problem. A point $x \in K_m^n$ corresponds to the partition $\mathcal{I}^+(x), \mathcal{I}^-(x)$ obtained as follows.

1. Provisionally put the label "$+$" on the whole interval $[0, 1]$
2. For $\ell = 1, 2, \ldots, n$:
   - if $x_\ell > 0$, then put label "$+$" on the interval $[0, x_\ell]$;
   - if $x_\ell < 0$, then put label "$-$" on the interval $[0, -x_\ell]$.

Note that subsequent assignments of a label to an interval, "overwrite" previous assignments. One way of thinking about it, is that we are applying a coat of paint on the interval $[0, 1]$. Initially the whole interval is painted with colour "$+$", and as the procedure is executed, various subintervals will be painted over with colour "$-$" or "$+$". It is easy to check that the final partition into $\mathcal{I}^+(x), \mathcal{I}^-(x)$ that is obtained, uses at most n cuts. Furthermore, for any $x \in \partial K_m^n$, the partition $\mathcal{I}^+(-x), \mathcal{I}^-(-x)$ obtained from $-x$ corresponds to the partition $\mathcal{I}^+(x), \mathcal{I}^-(x)$ with labels "$+$" and "$-$" switched. In other words, $\mathcal{I}^+(-x) = \mathcal{I}^-(x)$ and $\mathcal{I}^-(-x) = \mathcal{I}^+(x)$. For a more formal definition of this encoding, see [Filos-Ratsikas et al., 2021].

We define a labelling $\ell : K_m^n \to \{\pm 1, \pm 2, \ldots, \pm n\}$ as follows. For any $x \in K_m^n$:

1. Let $i \in [n]$ be the agent that sees the largest difference between $v_i(\mathcal{I}^+(x))$ and $v_i(\mathcal{I}^-(x))$, i.e., $i = \arg\max_{i \in [n]} |v_i(\mathcal{I}^+(x)) - v_i(\mathcal{I}^-(x))|$, where we break ties by picking the smallest such i.
2. Pick a sign $s \in \{+, -\}$ as follows. If $v_i(\mathcal{I}^+(x)) > v_i(\mathcal{I}^-(x))$, then let $s = +$. If $v_i(\mathcal{I}^+(x)) < v_i(\mathcal{I}^-(x))$, then let $s = -$. If $v_i(\mathcal{I}^+(x)) = v_i(\mathcal{I}^-(x))$, then pick s such that $\mathcal{I}^s$ contains the left end of the interval $[0, 1]$.
3. Set $\ell(x) = +i$ if $s = +$, and $\ell(x) = -i$ otherwise.

With this definition, it is easy to check that $\ell(-x) = -\ell(x)$ for all $x \in \partial(K_m^n)$. By re-interpreting K_m^n as a grid $[N]^n$ with $N = 2m + 1$, we thus obtain an instance $\widehat{\ell} : [N]^n \to \{\pm 1, \pm 2, \ldots, \pm n\}$ of nD-Tucker. In particular, note that $\widehat{\ell}$ is antipodally anti-symmetric on the boundary, as required.

Correctness. Any solution to the nD-TUCKER instance $\widehat{\ell}$ yields $x, y \in K_m^n$ with $\|x-y\|_\infty \leq 1/m$ and $\ell(x) = -\ell(y)$. Without loss of generality, assume that $\ell(x) = +i$ for some $i \in [n]$. Since $\|x-y\|_\infty \leq 1/m$, we obtain that

$$\lambda(\mathcal{I}^+(x) \triangle \mathcal{I}^+(y)) \leq \sum_{j=1}^{n} |x_j - y_j| = \|x-y\|_1 \leq n\|x-y\|_\infty \leq n/m$$

and the same bound also holds for $\lambda(\mathcal{I}^-(x) \triangle \mathcal{I}^-(y))$. Since v_i is Lipschitz-continuous with parameter L, it follows that

$$|v_i(\mathcal{I}^+(x)) - v_i(\mathcal{I}^+(y))| \leq L \cdot \lambda(\mathcal{I}^+(x) \triangle \mathcal{I}^+(y)) \leq nL/m$$

and similarly for $|v_i(\mathcal{I}^-(x)) - v_i(\mathcal{I}^-(y))|$.

Since $\ell(x) = +i$, it follows that $v_i(\mathcal{I}^+(x)) \geq v_i(\mathcal{I}^-(x))$. For the sake of contradiction, let us assume that $v_i(\mathcal{I}^+(x)) > v_i(\mathcal{I}^-(x)) + \varepsilon$. Then, it follows that

$$v_i(\mathcal{I}^+(y)) - v_i(\mathcal{I}^-(y)) \geq v_i(\mathcal{I}^+(x)) - v_i(\mathcal{I}^-(x)) - 2nL/m > \varepsilon - 2nL/m \geq 0$$

since $m \geq 2nL/\varepsilon$. But this contradicts the fact that $\ell(y) = -i$. Thus, it must hold that $|v_i(\mathcal{I}^+(x)) - v_i(\mathcal{I}^-(x))| \leq \varepsilon$. Since $\ell(x) = +i$, it follows that for all $j \in [n]$

$$|v_j(\mathcal{I}^+(x)) - v_j(\mathcal{I}^-(x))| \leq |v_i(\mathcal{I}^+(x)) - v_i(\mathcal{I}^-(x))| \leq \varepsilon.$$

This means that $\mathcal{I}^+(x), \mathcal{I}^-(x)$ yields a solution to the original ε-CONSENSUS-HALVING instance.

Note that the reduction uses $N = 2m + 1 \leq 4nL/\varepsilon + 3$ for the nD-TUCKER instance. Furthermore, any query to the labelling function $\widehat{\ell}$ can be answered by performing $2n$ queries to the valuation functions $v_1, \ldots, v_n$.

Our efficient black-box reduction in this section has several corollaries. First, it implies that ε-CONSENSUS-HALVING is in PPA, since nD-TUCKER is in PPA, by Theorem 3.1. As we remarked earlier, this was known for additive valuations, via a similar proof [Filos-Ratsikas et al., 2021].

Proposition 4.1. *For any constant $n \geq 2$, ε-CONSENSUS-HALVING with n general agents is in PPA.*

The next implication is that ε-CONSENSUS-HALVING with $n \geq 2$ agents with general valuations (and thus also for monotone valuations) can be solved with $O((L/\varepsilon)^{n-1})$ queries. This is implied by the query complexity upper bounds of Theorem 3.2, as in our reduction N is at most $4nL/\varepsilon + 3$, and each query to the labelling function of nD-TUCKER requires $2n$ queries to the functions v_i, where n is constant.

Proposition 4.2. *For any constant $n \geq 2$, the query complexity of ε-CONSENSUS-HALVING with n general agents is $O((L/\varepsilon)^{n-1})$, even in the weak black-box model.*

By the results of Section 4.1 above, we also obtain corollaries for the nD-BORSUK-ULAM problem, since nD-BORSUK-ULAM essentially reduces to nD-TUCKER via ε-CONSENSUS-HALVING. For the computational complexity of the problem in the white-box model, we have

Proposition 4.3. *For any constant $n \geq 2$, nD-BORSUK-ULAM is in PPA.*

Similarly, for the query complexity of the problem in the black-box model, we have

Proposition 4.4. *For any constant $n \geq 2$, the query complexity of nD-BORSUK-ULAM is $O((L/\varepsilon)^{n-1})$.*

5 General Valuations

We are now ready to prove our main results for the ε-Consensus-Halving problem, starting from the case of general valuations. First, it is relatively easy to see that for a single agent with a general valuation function, a simple binary search procedure is sufficient to find an ε-Consensus-Halving with a polynomial number of queries and in polynomial time, therefore obtaining an efficient algorithm both in the white-box and in the black-box model.

Theorem 5.1. *For one agent with a general valuation function (or multiple agents with identical general valuations), there is a polynomial-time algorithm which solves ε-Consensus-Halving using $O\left(\log \frac{L}{\varepsilon}\right)$ queries, even in the weak black-box model.*

Proof. We will prove the theorem for the case of $n = 1$, as a solution to ε-Consensus-Halving for this case is also straightforwardly a solution to the problem with multiple agents with identical valuations. Our algorithm essentially simulates binary search. We say that the *label of a cut $x \in [0,1]$* is "+", if $v([0,x]) > v([x,1]) + \varepsilon$. Respectively, the label of cut x is "−", if $v([x,1]) > v([0,x]) + \varepsilon$. In any other case, the label of the cut is 0; if a cut has label 0, then it is a solution to ε-Consensus-Halving. Observe that in order for an interval $[a,b] \subseteq [0,1]$ to contain a solution, it suffices that the label of a be "−" and the label of b be "+" (or vice-versa); then there is definitely a point $x \in [a,b]$ where the label is 0 (by continuity of v).

Now let $a = 0$ and $b = 1$. If a or b has label 0, then we have immediately found a solution. Otherwise, note that if a has label "−", then b must have label "+", and vice-versa. For convenience, in what follows, we assume that a has label "−" and b has label "+". Our algorithm proceeds as follows in every iteration. Given an interval $[a,b]$ with label "−" for a and label "+" for b, it computes the label of $\frac{a+b}{2}$. This can be done via two eval queries. Then, if the label of $\frac{a+b}{2}$ is "+", it sets $b = \frac{a+b}{2}$; if the label is "−", it sets $a = \frac{a+b}{2}$; and if the label is 0 it outputs this cut.

We claim that the algorithm will always find a cut with label 0 after at most $\log \frac{L}{\varepsilon}$ iterations. For the sake of contradiction, assume that there is no such cut after $\log \frac{L}{\varepsilon}$ iterations. Observe that the length of $[a,b]$ in this case will be $\frac{\varepsilon}{L}$. In addition, we know the labels of a and b. Cut a has label "−", thus $v([a,1]) > v([0,a]) + \varepsilon$, and cut b has label "+", i.e., $v([0,b]) > v([b,1]) + \varepsilon$. Since $|b-a| \le \frac{\varepsilon}{L}$ and v is L-Lipschitz-continuous, it follows that

$$|v([a,1]) - v([b,1])| \le L \cdot \lambda([a,b)) \le L \cdot \frac{\varepsilon}{L} = \varepsilon$$

and similarly $|v([0,a]) - v([0,b])| \le \varepsilon$. Putting everything together, we obtain that

$$v([0,b]) \le v([0,a]) + \varepsilon < v([a,1]) \le v([b,1]) + \varepsilon$$

which contradicts the assumption that cut b has label "+". □

From Theorem 5.1, we have the following two corollaries. The first one follows from the trivial observation that a polynomial-time algorithm which queries the polynomial-time algorithm of the input $O\left(\log \frac{L}{\varepsilon}\right)$ times is a polynomial-time algorithm.

Corollary 5.2. *For one agent with a general valuation function (or multiple agents with identical general valuations), ε-Consensus-Halving is solvable in polynomial time.*

The second corollary regards the black-box model, where we have the following result.

Corollary 5.3. *For one agent with a general valuation function (or multiple agents with identical general valuations), the query complexity of* ε-Consensus-Halving *is* $\Theta\left(\log \frac{L}{\varepsilon}\right)$.

Again, the upper bound follows from Theorem 5.1 above, whereas the lower bound follows from our general reduction from nD-Borsuk-Ulam in Section 4, and the query lower bounds for the latter problem obtained in Section 5.1 below. In more detail, $1D$-Borsuk-Ulam (and thus ε-Consensus-Halving with a single agent) inherits its query complexity lower bounds from $1D$-Tucker, which can be easily seen to require at least $\Omega(\log N)$ queries in the worst-case. The latter bound naturally translates to a $\Omega(\log(L/\varepsilon))$ bound for ε-Consensus-Halving. We also remark that the upper bound holds for any version of the problem with general valuations, even in the weak black-box model, whereas the lower bound holds even for normalised general valuations and for the standard black-box model.

We now move to our results for two or more agents with general valuations. Here we obtain a PPA-completeness result for ε-Consensus-Halving, as well as exponential bounds on the query complexity of the problem. Our results demonstrate that for general valuations, even in the case of two agents, the problem is intractable in both the black-box and the white-box model.

Our positive results were already obtained and presented in Section 4.2, so in this section we will obtain the matching impossibility results. As we explained in Section 4, all the impossibility results will be established via an efficient black-box reduction from nD-Tucker to nD-Borsuk-Ulam, which is captured by the following lemma.

Lemma 5.4. *For any constant* $n \geq 1$, nD-Tucker *reduces to normalised* nD-Borsuk-Ulam, *via an efficient black-box reduction.*

Before we proceed with the proof of the lemma, we explain how we can obtain our results as its corollaries, together with the results in Section 4.2, both in the white-box and in the black-box model.

White-box model. The first corollary of Lemma 5.4 is the PPA-hardness of nD-Borsuk-Ulam. Together with Proposition 4.3, we have the following theorem.

Theorem 5.5. nD-Borsuk-Ulam *is PPA-complete, for any constant* $n \geq 2$. *This remains the case, even if (a) we fix* $\varepsilon \in (0, 1)$, *or (b) we fix* $L \geq 3$.

Note that while the "in PPA" result does not require any assumptions, the PPA-hardness result holds also for normalised nD-Borsuk-Ulam, making both results stronger. Also note that if both ε and L are fixed, then the problem can be solved in polynomial time.

From the reduction in Section 4.1, Theorem 5.5 and Proposition 4.1, we immediately have the following corollary, which is our main result of the section for the computational complexity of ε-Consensus-Halving.

Theorem 5.6. *For any constant* $n \geq 2$, ε-Consensus-Halving *with n normalised general agents is PPA-complete. This remains the case, even if (a) we fix* $\varepsilon \in (0, 1)$, *or (b) we fix* $L \geq 3(n + 1)$.

Black-box model. In the black-box model, Lemma 5.4 has similar implications. First, it implies that normalised nD-Borsuk-Ulam requires $\Omega((L/\varepsilon)^{n-1})$ queries. Together with Proposition 4.4, we obtain the following theorem.

Theorem 5.7. *Let $n \geq 2$ be any constant. There exists a constant $c > 0$ such that for any $\varepsilon \in (0, 1)$ and any $L \geq 3$ with $L/\varepsilon \geq c$, the query complexity of nD-Borsuk-Ulam is $\Theta((L/\varepsilon)^{n-1})$.*

Again, from the reduction in Section 4.1, Theorem 5.7 and Proposition 4.2, we immediately have the following corollary, which is our main result of the section for the query complexity of ε-Consensus-Halving.

Theorem 5.8. *Let $n \geq 2$ be any constant. There exists a constant $c > 0$ such that for any $\varepsilon \in (0, 1)$ and any $L \geq 3(n+1)$ with $L/\varepsilon \geq c$, the query complexity of ε-Consensus-Halving with n normalised general agents is $\Theta((L/\varepsilon)^{n-1})$ queries.*

Again, we remark that the lower bounds hold even for agents with normalised valuation functions, whereas the upper bounds hold for any valuation functions and even in the weak black-box model, therefore making both results stronger.

The next section is dedicated to the proof of Lemma 5.4.

5.1 Reducing nD-Tucker to normalised nD-Borsuk Ulam

Let $n \geq 1$ be any constant. Consider an instance $\ell : [N]^n \to \{\pm 1, \pm 2, \ldots, \pm n\}$ of nD-Tucker. Let $\varepsilon \in (0, 1)$. We will construct a normalised nD-Borsuk-Ulam function $F : B^{n+1} \to B^n$ that is Lipschitz-continuous with Lipschitz parameter $L = \max\{3, 4n^2(N-1)\varepsilon + 1\}$ and such that any $x \in \partial(B^{n+1})$ with $\|F(x)\|_\infty \leq \varepsilon$ yields a solution to the nD-Tucker instance.

Let $\delta = \min\{2\varepsilon, 1\}$. Note that $\delta \in (0, 1]$ and $\varepsilon < \delta < 2\varepsilon$. Without loss of generality, we can assume that for $p = (N, N, \ldots, N)$ it holds $\ell(p) = +1$. Indeed, it is easy to see that we can rename the labels to achieve this, without introducing any new solutions.

Step 1: Interpolating the nD-Tucker instance. The first step is to embed the nD-Tucker grid $[N]^n$ in $[-1/2, 1/2]^n$, define the value of the function at every grid point according to the labelling function ℓ and then interpolate to obtain a continuous function $f : [-1/2, 1/2]^n \to [-\delta \cdot n, \delta \cdot n]$.

We embed the grid $[N]^n$ in $[-1/2, 1/2]^n$ in the most straightforward way, namely $p \in [N]^n$ corresponds to $\widehat{p} \in [-1/2, 1/2]^n$ such that $\widehat{p}_j = -1/2 + (p_j - 1)/(N-1)$ for all $j \in [n]$. Note that antipodal grid points exactly correspond to antipodal points in $[-1/2, 1/2]^n$. In other words, p and q are antipodal on the grid, if and only if $\widehat{p} = -\widehat{q}$.

Next we define the value of the function $f : [-1/2, 1/2]^n \to [-\delta \cdot n, \delta \cdot n]$ at the embedded grid points as follows

$$f(\widehat{p}) = \delta \cdot n \cdot e_{\ell(p)}$$

for all $p \in [N]^n$. For $i \in [n]$, e_{+i} denotes the ith unit vector in $\mathbb{R}^n$, and $e_{-i} := -e_{+i}$. We then use Kuhn's triangulation on the embedded grid to interpolate between these values and obtain a function $f : [-1/2, 1/2]^n \to [-\delta \cdot n, \delta \cdot n]$ (see Appendix C for more details). We obtain:

- f is antipodally anti-symmetric on the boundary of $[-1/2, 1/2]^n$, i.e., $f(-x) = -f(x)$ for all $x \in \partial([-1/2, 1/2]^n)$.
- f is Lipschitz-continuous with Lipschitz parameter $2n^2(N-1)\delta$, since the grid size is $1/(N-1)$ and $\|f(\widehat{p})\|_\infty \leq \delta \cdot n$ for all $p \in [N]^n$.

- Any $x \in [-1/2, 1/2]^n$ such that $\|f(x)\|_\infty \le \varepsilon$ must lie in a Kuhn simplex that contains two grid points $p, q \in [N]^n$ such that $\ell(p) = -\ell(q)$, i.e., a solution to the nD-TUCKER instance. Indeed, let $p^0, p^1, \dots, p^n \in [N]^n$ be the grid points of the Kuhn simplex containing x. If $\{\ell(p^0), \ell(p^1), \dots, \ell(p^n)\}$ does not contain two opposite labels, then all the points in $V = \{f(\widehat{p^0}), f(\widehat{p^1}), \dots, f(\widehat{p^n})\}$ lie in the same orthant of $\mathbb{R}^n$. Since $\|f(\widehat{p^i})\|_\infty = \delta \cdot n$ for all i, it follows that any convex combination v of vectors in V must be such that $\|v\|_1 \ge \delta \cdot n$, and thus $\|v\|_\infty \ge \delta > \varepsilon$. Thus, if $\|f(x)\|_\infty \le \varepsilon$, then $\{\ell(p^0), \ell(p^1), \dots, \ell(p^n)\}$ must contain two opposite labels.

- $f(1/2, 1/2, \dots, 1/2) = \delta \cdot n \cdot e_{+1} = (\delta \cdot n, 0, 0, \dots, 0)$.

Now, we define $g : [-1/2, 1/2]^n \to [-\delta, \delta]^n$ to be the truncation of f to $[-\delta, \delta]^n$, namely

$$g_i(x) = \min\{\delta, \max\{-\delta, f_i(x)\}\}.$$

It is not hard to see that g is also antipodally anti-symmetric, Lipschitz-continuous with Lipschitz parameter $2n^2(N-1)\delta$ and $g(1/2, 1/2, \dots, 1/2) = \delta \cdot e_{+1}$. Furthermore, if $x \in [-1/2, 1/2]^n$ is such that $\|g(x)\|_\infty \le \varepsilon$, then, since $\varepsilon < \delta$, $\|f(x)\|_\infty \le \varepsilon$, and thus x again yields a solution to the nD-TUCKER instance.

Step 2: Extending to $[-1, 1]^n$. The goal of the next step is to define a function $h : [-1, 1]^n \to [-1, 1]^n$ that extends g and ensures that $h(1, 1, \dots, 1) = (1, 1, \dots, 1)$, while maintaining its other properties. For $x \in [-1, 1]^n$ we let $T(x) \in [-1/2, 1/2]^n$ denote its truncation to $[-1/2, 1/2]^n$, i.e., $[T(x)]_i = \min\{1/2, \max\{-1/2, x_i\}\}$ for all $i \in [n]$. The function $h : [-1, 1]^n \to [-1, 1]^n$ is defined as

$$h(x) = \begin{cases} (2\min_j x_j - 1) \cdot \mathbb{1} + (2 - 2\min_j x_j) \cdot \delta \cdot e_{+1} & \text{if } x_i \ge 1/2 \text{ for all } i \\ (2\min_j(-x_j) - 1) \cdot (-\mathbb{1}) + (2 - 2\min_j(-x_j)) \cdot \delta \cdot e_{-1} & \text{if } x_i \le -1/2 \text{ for all } i \\ g(T(x)) & \text{otherwise} \end{cases}$$

where $\mathbb{1} \in \mathbb{R}^n$ denotes the all-ones vector, i.e., $\mathbb{1} = (1, 1, \dots, 1)$. Clearly, it holds that $h(1, 1, \dots, 1) = \mathbb{1}$. It is also easy to see that $h(-x) = -h(x)$ for all $x \in \partial([-1, 1]^n)$, in particular because $T(-x) = -T(x)$. Furthermore, if $\|h(x)\|_\infty \le \varepsilon$, then it must be that $\|g(T(x))\|_\infty \le \varepsilon$, which yields a solution to the nD-TUCKER instance. Indeed, if $x_i \ge 1/2$ for all i, then

$$h_1(x) = (2\min_j x_j - 1) \cdot 1 + (2 - 2\min_j x_j) \cdot \delta \ge \delta > \varepsilon$$

so $\|h(x)\|_\infty > \varepsilon$. By the same argument, if $x_i < -1/2$ for all i, then we also have $\|h(x)\|_\infty > \varepsilon$.

Since $g(1/2, 1/2, \dots, 1/2) = \delta \cdot e_{+1}$ and $g(-1/2, -1/2, \dots, -1/2) = \delta \cdot e_{-1}$, it is easy to see that h is continuous. Furthermore, since for any $x, y \in [-1, 1]^n$ it holds that $\|T(x) - T(y)\|_\infty \le \|x - y\|_\infty$, it is easy to see that h is $2n^2(N-1)\delta$-Lipschitz-continuous outside of $\{x \in [-1, 1]^n | x_i \ge 1/2 \text{ for all } i \in [n]\} \cup \{x \in [-1, 1]^n | x_i \le -1/2 \text{ for all } i \in [n]\}$. For any $y, z \in \{x \in [-1, 1]^n | x_i \ge 1/2 \text{ for all } i \in [n]\}$, it holds that $|h_i(y) - h_i(z)| = 2|\min_j y_j - \min_j z_j| \le 2\|y - z\|_\infty$ for $i > 1$, and $|h_1(y) - h_1(z)| = 2(1 - \delta)|\min_j y_j - \min_j z_j| \le 2\|y - z\|_\infty$. Thus, h is 2-Lipschitz-continuous on $\{x \in [-1, 1]^n | x_i \ge 1/2 \text{ for all } i \in [n]\}$ and, by the same argument, also on $\{x \in [-1, 1]^n | x_i \le -1/2 \text{ for all } i \in [n]\}$.

As a result, h is Lipschitz-continuous on $[-1,1]^n$ with Lipschitz parameter $\max\{2, 2n^2(N-1)\delta\}$. Indeed, consider any $x, y \in [-1,1]^n$. If $x_i \geq 1/2$ and $y_i \leq -1/2$ for all i, then $\|x-y\|_\infty \geq 1$, and thus $\|h(x) - h(y)\|_\infty \leq 2 \leq 2\|x-y\|_\infty$. By symmetry, the only remaining case that we need to check is when $x_i \geq 1/2$ for all i, and y is such that there exists i with $y_i < 1/2$ and there exists i with $y_i > -1/2$. In that case, we consider the segment $[x,y]$ from x to y, and let $z \in [x,y]$ be the point that is the furthest away from x but such that $z_i \geq 1/2$ for all i. Note that there must exist i such that $z_i = 1/2$. This means $h(z) = g(T(z))$ and thus $\|h(z) - h(y)\|_\infty \leq 2n^2(N-1)\delta\|z-y\|_\infty$. On the other hand, we have $\|h(x) - h(z)\|_\infty \leq 2\|x-z\|_\infty$. Putting these two expressions together, we obtain that $\|h(x) - h(y)\|_\infty \leq \max\{2, 2n^2(N-1)\delta\}(\|x-z\|_\infty + \|z-y\|_\infty) = \max\{2, 2n^2(N-1)\delta\}\|x-y\|_\infty$. Here we used the fact that $z \in [x,y]$, which means that there exists $t \in [0,1]$ such that $z = x + t(y-x)$ and thus

$$\|x-z\|_\infty + \|z-y\|_\infty = t\|x-y\|_\infty + (1-t)\|x-y\|_\infty = \|x-y\|_\infty.$$

Step 3: Extending to $[-1,1]^{n+1}$. The final step is to define a normalised nD-Borsuk-Ulam function $F : [-1,1]^{n+1} \to [-1,1]^n$ such that any $x \in \partial([-1,1]^{n+1})$ with $\|F(x)\|_\infty \leq \varepsilon$ yields a solution to the nD-Tucker instance. For $x \in [-1,1]^{n+1}$ we write $x = (x', x_{n+1})$, where $x' \in [-1,1]^n$. We define

$$F(x) = F(x', x_{n+1}) = \frac{1+x_{n+1}}{2}h(x') + \frac{1-x_{n+1}}{2}(-h(-x')).$$

Since $h(x'), -h(-x') \in [-1,1]$ and $F(x)$ is a convex combination of these two, it follows that $F(x) \in [-1,1]^n$. Furthermore, we have $F(1,1,\dots,1) = h(1,1,\dots,1) = \mathbb{1}$. F is an odd function, since

$$F(-x) = F(-x', -x_{n+1}) = \frac{1-x_{n+1}}{2}h(-x') + \frac{1+x_{n+1}}{2}(-h(x')) = -F(x', x_{n+1}) = -F(x).$$

Consider any $x = (x', x_{n+1}) \in \partial([-1,1]^{n+1})$ with $\|F(x)\|_\infty \leq \varepsilon$. Since F is an odd function, we can assume that $x_{n+1} \geq 0$ (otherwise just use $-x$ instead of x). If $x_{n+1} = 1$, then $F(x', x_{n+1}) = h(x')$, and thus $\|h(x')\|_\infty \leq \varepsilon$, which yields a solution to the nD-Tucker instance. If $x_{n+1} \in [0,1)$, then $x' \in \partial([-1,1]^n)$ and thus $h(x') = -h(-x')$. This implies that $F(x', x_{n+1}) = h(x')$ in this case too.

Finally, let us determine the Lipschitz parameter of F. Let $x, y \in [-1,1]^{n+1}$. We have

$$\begin{aligned}\|F(x', x_{n+1}) - F(y', x_{n+1})\|_\infty &\leq \frac{1+x_{n+1}}{2}\|h(x') - h(y')\|_\infty + \frac{1-x_{n+1}}{2}\|h(-x') - h(-y')\|_\infty \\ &\leq \max\{2, 2n^2(N-1)\delta\}\|x'-y'\|_\infty\end{aligned}$$

and also

$$\|F(y', x_{n+1}) - F(y', y_{n+1})\|_\infty \leq \frac{|x_{n+1} - y_{n+1}|}{2}(\|h(y')\|_\infty + \|h(-y')\|_\infty) \leq |x_{n+1} - y_{n+1}|.$$

Putting these two expression together, it follows that

$$\|F(x) - F(y)\|_\infty \leq \max\{3, 2n^2(N-1)\delta + 1\}\|x-y\|_\infty.$$

Note that $\max\{3, 2n^2(N-1)\delta + 1\} \leq \max\{3, 4n^2(N-1)\varepsilon + 1\}$.

In the black-box model, in order to answer one query to F, we have to answer two queries to h, i.e., two queries to g. In order to answer a query to g, we have to answer one query to f, i.e., $n+1$ queries to the labelling function ℓ (in order to interpolate). Thus, one query to F requires $2(n+1)$ queries to ℓ.

This black-box reduction has various consequences.

White-box model. The reduction actually gives us a way to construct an arithmetic circuit that computes F, if we are given a Boolean circuit that computes ℓ. Indeed, using standard techniques [Chen et al., 2009; Daskalakis et al., 2009], the execution of the Boolean circuit on some input can be simulated by the arithmetic circuit. Furthermore, the input bits for the Boolean circuit can be obtained by using the $<$ gate. All the other operations that we used to construct F can be computed by the arithmetic gates $+, -, \times, \max, \min, <$ and rational constants. Thus, we obtain a polynomial-time many-one reduction from nD-Tucker to normalised nD-Borsuk-Ulam for all $n \geq 1$. Since nD-Tucker is PPA-hard for any $n \geq 2$, we immediately obtain the PPA-hardness of nD-Borsuk-Ulam. Together with the reduction from nD-Borsuk-Ulam to ε-Consensus-Halving, which is in PPA, we immediately obtain Theorem 5.5.

Black-box model. The reduction we have provided only requires black-box access to the labelling function ℓ of the nD-Tucker instance. In more detail, any query to F can be answered by making $2(n+1)$ queries to ℓ. Since n is a constant, the query lower bounds from nD-Tucker carry over to normalised nD-Borsuk-Ulam, and we obtain Theorem 5.7.

6 Monotone Valuations

In this section, we present our results for agents with monotone valuations. In contrast to the results of Section 5, here we prove that for two agents with monotone valuations, the problem is solvable in polynomial time and with a polynomial number of queries, and in fact this result holds even if only one of the two agent has a monotone valuation and the other has a general valuation. For three or more agents however, the problem becomes PPA-complete once again, and we obtain a corresponding exponential lower bound on its query complexity.

6.1 An efficient algorithm for two monotone agents

We start with our efficient algorithm for the case of two agents, which is a polynomial-time algorithm in the white-box model, as well as an algorithm of polynomial query complexity in the black-box model; see Algorithm 1. The algorithm is based on a *nested binary search* procedure: we start with an interval, initially defined by a position of the two cuts that "satisfies" the agent with the monotone valuation, when one of the two cuts sits on the left endpoint 0 of the interval. Then we iteratively "shrink" this interval based on the preference of the other agent, maintaining that (a) the first agent is always "satisfied" and (b) the solution that "satisfies" the other agent is contained in the considered interval that is being shrunk. Intuitively, we are moving on the "indifference curve" of the valuation function of the agent with the monotone valuation (see the red zig-zag line in Figure 3), and we "offer" the other agent points (i.e., sets of cuts), to which the agent replies with either "+" or "−", indicating the label of her discrepancy. Since there are "equivalent" cut positions in which the second agent responds with opposite labels (see the intersection points of the red line with the vertical axis and the horizontal axis in Figure 3), we know labels of the boundary points corresponding to this agent, and we can move in the direction towards the point with the opposite discrepancy label.

Before we proceed, we draw an interesting connection with *Austin's moving knife procedure* [Austin, 1982], an Exact-Division procedure for two agents with general valuations. The procedure is based on two moving knifes which one of the two agents simultaneously and continuously slides across the cake, maintaining that the corresponding cut positions ensure that she is satisfied with the partition. At some point during this process, the other agent becomes satisfied, which is guaranteed by the intermediate value theorem. Our algorithm can be interpreted as a discrete time implementation of this idea and quite interestingly, it results in a polynomial-time algorithm when one of the two agents has a monotone valuation, whereas it is computationally hard when both agents have general valuations, as shown in Section 5. On a more fundamental level, this demonstrates the intricacies of transforming moving-knife fair division protocols into discrete algorithms.

The main theorem of this section is the following.

Theorem 6.1. *For two agents with monotone valuations, there is a polynomial-time algorithm which solves* ε-Consensus-Halving *using* $O\left(\log^2 \frac{L}{\varepsilon}\right)$ *queries, even in the weak black-box model. This result holds even if one of the two agents has a general valuation.*

Before we present its proof, we state its corollaries. The first one follows from the fact that when v_i is a polynomial-time algorithm (in the white-box model), the whole algorithm is polynomial-time.

Corollary 6.2. *For two agents with monotone valuations,* ε-Consensus-Halving *is solvable in polynomial time. This result holds even if one of the two agents has a general valuation.*

For the query complexity of the problem, Theorem 6.1 straightforwardly implies the following theorem.

Theorem 6.3. *For two agents with monotone valuations, the query complexity of* ε-Consensus-Halving *is* $O\left(\log^2 \frac{L}{\varepsilon}\right)$. *This result holds even if one of the two agents has a general valuation.*

Next, we present the proof of Theorem 6.1.

6.1.1 Proof of Theorem 6.1

Before we proceed with our algorithm and its analysis, let us begin with some conventions that will make the presentation easier. Since we have to make two cuts, we can distinguish between them by calling them "left cut" and "right cut", and denoting $\textsc{cut}_\ell$ and $\textsc{cut}_r$ their positions respectively. We assume that $0 \leq \textsc{cut}_\ell \leq \textsc{cut}_r \leq 1$. In addition, the labels of the corresponding intervals are as follows: intervals $[0, \textsc{cut}_\ell]$ and $[\textsc{cut}_r, 1]$ have label "$+$", forming the positive piece, and interval $[\textsc{cut}_\ell, \textsc{cut}_r]$ has label "$-$", forming the negative piece. Given a pair of cuts $(\textsc{cut}_\ell, \textsc{cut}_r)$, agent i:

- prefers the positive piece, if $v_i([0, \textsc{cut}_\ell] \cup [\textsc{cut}_r, 1]) > v_i([\textsc{cut}_\ell, \textsc{cut}_r]) + \varepsilon$;

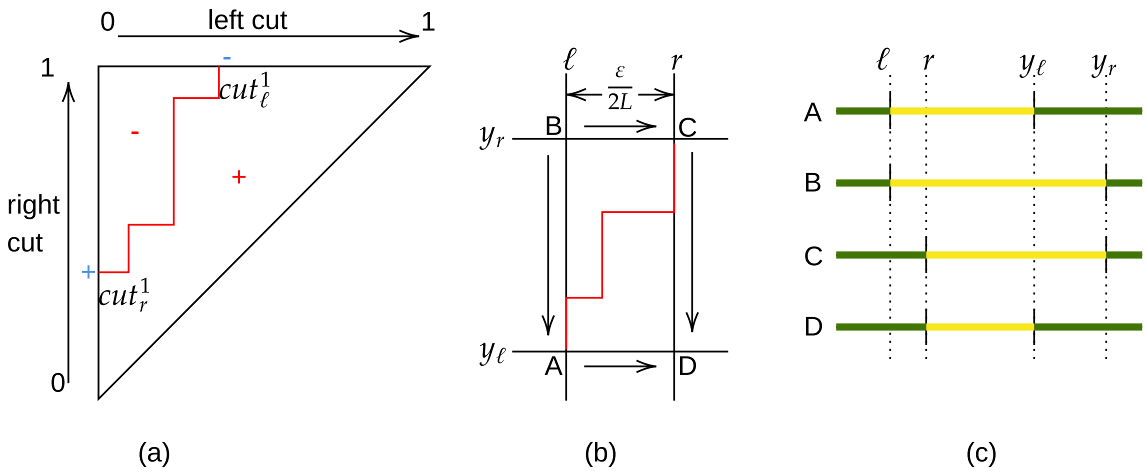

Figure 3: Visual interpretation of Algorithm 1. Subfigure (a) depicts our initial assumptions. The red line shows where Agent 1 is indifferent. The blue signs on $(0, \textsc{cut}_r^1)$ and $(\textsc{cut}_\ell^1, 1)$ show the preferences of Agent 2 under these pairs of cuts. Subfigure (b) shows a possible position for the cuts $r - \ell \leq \frac{\varepsilon}{2L}$. The arrows show how the difference between the values of the positive piece and the negative piece change between the four possible combinations of pairs of cuts. Subfigure (c) depicts the actual cuts on the cake: the green parts have label "+" and the yellow parts have label "−".

- prefers the negative piece, if $v_i([\textsc{cut}_\ell, \textsc{cut}_r]) > v_i([0, \textsc{cut}_\ell] \cup [\textsc{cut}_r, 1]) + \varepsilon$;
- is indifferent if $|v_i([\textsc{cut}_\ell, \textsc{cut}_r]) - v_i([0, \textsc{cut}_\ell] \cup [\textsc{cut}_r, 1])| \leq \varepsilon$.

The final assumption we need to make is regarding the preferences of Agent 2 for the two special pairs of $(0, \textsc{cut}_r^1)$ and $(\textsc{cut}_\ell^1, 1)$, where $\textsc{cut}_r^1 = \textsc{cut}_\ell^1$ and Agent 1 is indifferent under any pair of cuts. We note that we can find $\textsc{cut}_r^1$ and $\textsc{cut}_\ell^1$ efficiently using the algorithm from Theorem 5.1. Observe that both pairs of cuts cut the cake into two same pieces and only change the labels of the pieces. Hence, if Agent 2 prefers the positive piece under the pair of cuts $(0, \textsc{cut}_r^1)$, he *has to* prefer the negative piece under the pair of cuts $(\textsc{cut}_\ell^1, 1)$. In the description of the algorithm we will assume that this is indeed the case, which is without loss of generality. Using the above notation and assumptions we can now state Algorithm 1.

To prove the correctness of Algorithm 1 we need to prove that the following invariants are maintained through all iterations:

1. for any $x \in [\ell, r]$ there exists a $y \geq x$ such that the pair of cuts (x, y) makes Agent 1 indifferent between the two pieces;
2. for any $y = \frac{y_\ell + y_r}{2}$ there exists an $x \in [\ell, r]$ such that the pair of cuts (x, y) makes Agent 1 indifferent between the two pieces;
3. there exists a pair of cuts (x, y), where $x \in [\ell, r]$ and $y \geq [y_\ell, y_r]$ that makes Agent 2 indifferent between the two pieces.

ALGORITHM 1: ε-Consensus-Halving for two agents with monotone valuations

Set $\ell \leftarrow 0$ and $r \leftarrow \textsc{cut}^1_\ell$
Set $\textsc{cut}_\ell \leftarrow \textsc{cut}^1_\ell$ and $\textsc{cut}_r \leftarrow 1$
while *Agent 2 is not indifferent under the pair of cuts* $(\textsc{cut}_\ell, \textsc{cut}_r)$ **do**
  if $r - \ell > \frac{\varepsilon}{2L}$ **then**
    Find $y \geq \frac{\ell+r}{2}$, such that Agent 1 is indifferent under the pair of cuts $(\frac{\ell+r}{2}, y)$
    Set $\textsc{cut}_\ell \leftarrow \frac{\ell+r}{2}$ and $\textsc{cut}_r \leftarrow y$.
    if *Agent 2 prefers the positive piece under* $(\textsc{cut}_\ell, \textsc{cut}_r)$ **then**
      Set $\ell \rightarrow \frac{\ell+r}{2}$
    end
    if *Agent 2 prefers the negative piece under* $(\textsc{cut}_\ell, \textsc{cut}_r)$ **then**
      Set $r \rightarrow \frac{\ell+r}{2}$
    end
  else
    Let y_ℓ and y_r be such that Agent 1 is indifferent under the pairs of cuts (ℓ, y_ℓ) and (r, y_r)
    Find $x \in [\ell, r]$ such that Agent 1 is indifferent under the pair of cuts $(x, \frac{y_\ell+y_r}{2})$
    Set $\textsc{cut}_\ell \leftarrow x$ and $\textsc{cut}_r \leftarrow \frac{y_\ell+y_r}{2}$
    if *Agent 2 prefers the positive piece under* $(\textsc{cut}_\ell, \textsc{cut}_r)$ **then**
      Set $y_\ell \rightarrow \frac{y_\ell+y_r}{2}$
    end
    if *Agent 2 prefers the negative piece under* $(\textsc{cut}_\ell, \textsc{cut}_r)$ **then**
      Set $y_r \rightarrow \frac{y_\ell+y_r}{2}$
    end
  end
end

For the first invariant, observe the following. If the pair of cuts $(x, 1)$ make Agent 1 indifferent, then the claim holds. If on the other hand Agent 1 prefers a piece under this pair of cuts, it must be the case that he prefers the negative piece. This is due to monotonicity of v_1. In addition, observe that under the pair of cuts (x, x), the whole cake gets label "+", so in this case Agent 1 prefers the positive piece. From the intermediate value theorem we get that there must exist a $y \in [x, 1]$ such that $v_1([0, x] \cup [y, 1]) = v_1([x, y])$, i.e., that makes the agent indifferent.

For the second invariant, we need to consider two cases depending on whether $y_\ell \geq y_r$ or $y_\ell < y_r$. When $y_\ell \geq y_r$ we observe that due to monotonicity of the valuation of Agent 1 we get that

$$v_1([0,\ell] \cup [y_\ell, 1]) - v_1([\ell, y_\ell]) < v_1([0,\ell] \cup [y_r, 1]) - v_1([\ell, y_r]) < v_1([0,r] \cup [y_r, 1]) - v_1([r, y_r]).$$

In addition, since Agent 1 is indifferent under the pairs of cuts (ℓ, y_ℓ) and (r, y_r), we know that

$$|v_1([0,\ell] \cup [y_\ell, 1]) - v_1([\ell, y_\ell])| \leq \varepsilon \quad \text{and} \quad |v_1([0,r] \cup [y_r, 1]) - v_1([r, y_r])| \leq \varepsilon.$$

So, if $0 > v_1([0,\ell] \cup [y_\ell, 1]) - v_1([\ell, y_\ell]) > -\varepsilon$ it must be true that $0 > v_1([0,r] \cup [y_r, 1]) - v_1([r, y_r]) > -\varepsilon$. Thus, the existence of an x that makes Agent 1 indifferent under $(x, \frac{y_\ell + y_r}{2})$ follows from the intermediate value theorem. If $y_\ell < y_r$, then there are two cases.

- If $\text{sign}(v_1([0,\ell] \cup [y_\ell, 1]) - v_1([\ell, y_\ell])) \neq \text{sign}(v_1([0,r] \cup [y_r, 1]) - v_1([r, y_r]))$, then the invariant follows from the intermediate value theorem.

- If $\text{sign}(v_1([0,\ell] \cup [y_\ell, 1]) - v_1([\ell, y_\ell])) = \text{sign}(v_1([0,r] \cup [y_r, 1]) - v_1([r, y_r]))$, then using again the monotonicity of v_1, using the fact that Agent 1 is indifferent between the two pieces under the pairs of cuts (ℓ, y_ℓ) and (r, y_r), and the intermediate value theorem, we can again prove the claim.

For the third invariant, let us consider an iteration of the algorithm where $\textsc{cut}_\ell$ can be located on $[\ell, r]$. Let y_ℓ be chosen in such a way that Agent 1 is indifferent under the pair of cuts (ℓ, y_ℓ). Similarly, let y_r be such that Agent 1 is indifferent under the pair of cuts (r, y_r). Observe that, for any ℓ considered by Algorithm 1, Agent 2 prefers the positive piece for the pair of cuts (ℓ, y_ℓ). In addition, for any r considered by Algorithm 1, Agent 2 prefers the negative piece for the pair of cuts (r, y_r). This is due to the assumption that Agent 2 prefers the negative piece under the pair of cuts $(\textsc{cut}^1_\ell, 1)$ and that he prefers the positive cut under the pair of cuts $(0, \textsc{cut}^1_r)$, and the update rule used in Steps 8 and 12.

Next, in order to bound the running time of our algorithm we need to bound two things: the time needed for Steps 5 and 15, and the number of iterations of the algorithm. The overall running time of the algorithm will be bounded by the number of iterations multiplied by the time needed for Step 5, or Step 15. Firstly, observe that Steps 5 and 15 can be tackled using the algorithm from Theorem 5.1. This is because we can view the problem as a special case where we have one agent and we need to place a single cut in a specific subinterval that, due to the first two invariants, we know that a solution exists. Thus, each of these steps requires $O(\log \frac{L}{\varepsilon})$ time. In addition, observe that after $O(\log \frac{L}{\varepsilon})$ iterations we get that $r - \ell \leq \frac{\varepsilon}{2L}$. Similarly, after $O(\log \frac{L}{\varepsilon})$ we get that $|y_r - y_\ell| \leq \frac{\varepsilon}{2L}$ as well. So, after at most $O(\log^2 \frac{L}{\varepsilon})$ time we either get a pair of cuts $(\textsc{cut}_\ell, \textsc{cut}_r)$ that make both agents indifferent between the pieces, or we have two pairs of cuts (ℓ, y_ℓ) and (r, y_r) such that the following hold. Firstly, their ℓ_1 distance is at most $\frac{\varepsilon}{L}$. In addition, due to the updating rules at Steps 8, 12, 18, and 22, we get that $v_2([0,\ell] \cup [y_\ell, 1]) > v_2([[\ell, y_\ell]])$ and that $v_2([0,r] \cup [y_r, 1]) < v_2([[r, y_r]])$. These facts combined with the L-Lipschitz-continuity of valuation function v_2 imply that at least one pair of cuts between (ℓ, y_ℓ) and (r, y_r) makes Agent 2 indifferent. The proof for the two monotone agents is completed using the fact that Agent 1 is indifferent for any pair of cuts between (ℓ, y_ℓ) and (r, y_r) and the fact that we can simulate every step using queries; Steps 5 and 15 can be simulated with $O(\log \frac{\varepsilon}{2L})$ as it was already explained in the proof of Theorem 5.1 and each of Steps 8, 12, 18, and 22 can be simulated via two queries. Finally, we observe that we have not used the fact that Agent 2 is monotone. Hence, the Algorithm works even when Agent 2 has a general valuation function.

6.2 Results for three or more monotone agents

We now move on to the case of three or more monotone agents, for which we manage to show that the problem becomes computationally hard and has exponential query complexity. Our results thus show a clear dichotomy on the complexity of ε-Consensus-Halving with monotone agents, between the case of two agents and the case of three or more agents.

Central to all of the results in this section will be the following lemma, the proof of which we present in Section 6.2.1 below.

Lemma 6.4. *For any constant $n \geq 2$, $(n-1)D$-Tucker reduces to normalised monotone nD-Borsuk-Ulam in polynomial time, via an efficient black-box reduction.*

Lemma 6.4 has several implications, both in the white-box and the black-box model.

White-box model. The first corollary of Lemma 6.4 is the PPA-hardness of monotone nD-Borsuk-Ulam, for $n \geq 3$, since by Theorem 3.1 nD-Tucker is PPA-hard for $n \geq 2$. Together with Proposition 4.4, which holds for the general version of the problem (and thus also the monotone version), we have the following corollary.

Theorem 6.5. *Monotone nD-Borsuk-Ulam is PPA-complete, for any constant $n \geq 3$. This remains the case, even if (a) we fix $\varepsilon \in (0, 1)$, or (b) we fix $L \geq 3$.*

We remark that the PPA-hardness still holds even if we restrict our attention to the normalised nD-Borsuk-Ulam problem, whereas the PPA-membership holds under no assumptions, so both results are stronger.

From the reduction in Section 4.1, which is property-preserving, Theorem 6.5 and Proposition 4.1, we immediately obtain our main result of the section for the computational complexity of ε-Consensus-Halving with $n \geq 3$ monotone agents.

Theorem 6.6. *For any constant $n \geq 3$, ε-Consensus-Halving with n monotone agents is PPA-complete. This remains the case, even if (a) we fix $\varepsilon \in (0, 1)$, or (b) we fix $L \geq 3(n+1)$.*

Once again, we remark that the PPA-hardness still holds even if we restrict our attention to agents with normalised valuation functions, whereas the upper bound holds without any assumptions.

Black-box model. In the black-box model, Lemma 6.4 has similar implications. Interestingly, since we are reducing from an instance of $(n-1)D$-Tucker (i.e., of one dimension lower), we inherit the query complexity lower bounds for the problem for that case. Together with Proposition 4.4, we get the following theorem.

Theorem 6.7. *Let $n \geq 3$ and $t \in (0, 1)$ be any two constants. There exists a constant $c > 0$ such that for any $\varepsilon \in (0, t)$ and any $L \geq 3$ with $L/\varepsilon \geq c$, the query complexity of monotone nD-Borsuk-Ulam is between $\Omega((L/\varepsilon)^{n-2})$ and $O((L/\varepsilon)^{n-1})$.*

Again, the lower bounds also hold for the case of normalised monotone nD-Borsuk-Ulam. From the reduction in Section 4.1, which is property-preserving, together with Theorem 6.7 and Proposition 4.2, we obtain our main result for the query complexity of ε-Consensus-Halving for $n \geq 3$ monotone agents.

Theorem 6.8. *Let $n \geq 3$ and $t \in (0, 1)$ be any two constants. There exists a constant $c > 0$ such that for any $\varepsilon \in (0, t)$ and any $L \geq 3(n+1)$ with $L/\varepsilon \geq c$, the query complexity of ε-Consensus-Halving with n monotone agents is between $\Omega((L/\varepsilon)^{n-2})$ and $O((L/\varepsilon)^{n-1})$.*

The lower bounds again hold even if the valuation functions are normalised. There is a small gap between our lower and upper bounds; we conjecture that it should be possible to prove upper bounds that match the lower bounds of Theorem 6.8, at least up to logarithmic factors, but we leave this for future work.

We now proceed with the proof of Lemma 6.4, from which we obtain all the results of the section.

6.2.1 Reducing $(n-1)D$-Tucker to monotone nD-Borsuk-Ulam

Let $n \geq 2$ be any fixed constant. Consider an instance $\ell : [N]^{n-1} \to \{\pm 1, \pm 2, \dots, \pm(n-1)\}$ of $(n-1)D$-Tucker. Let $\varepsilon \in (0,1)$. We will construct a monotone normalised nD-Borsuk-Ulam function $F : [-1,1]^{n+1} \to [-1,1]^n$ that is Lipschitz-continuous with Lipschitz parameter $L = (4(n+1)^4 N\varepsilon + 2)/(\min\{2, 1/\varepsilon\} - 1)$ and such that any $x \in \partial(B^{n+1})$ with $\|F(x)\|_\infty \leq \varepsilon$ yields a solution to the $(n-1)D$-Tucker instance.

Step 1: From $(n-1)D$-Tucker to non-normalised $(n-1)D$-Borsuk-Ulam. Let $\delta = \min\{2\varepsilon, 1\}$. Note that $\delta \in (0,1]$ and $\varepsilon < \delta < 2\varepsilon$. The first step of the proof is identical to the proof of Lemma 5.4. Namely, we construct $g : [-1/2, 1/2]^{n-1} \to [-\delta, \delta]^{n-1}$ such that g is antipodally anti-symmetric and $4(n-1)^2(N-1)\varepsilon$-Lipschitz-continuous. Furthermore, any $x \in [-1/2, 1/2]^{n-1}$ with $\|g(x)\|_\infty < \delta$ yields a solution to the original $(n-1)D$-Tucker instance. Then, we define $h : [-1,1]^{n-1} \to [-\delta, \delta]^{n-1}$ by $h(x) = g(x/2)$. h has the same properties as g, except that it is $2(n-1)^2(N-1)\varepsilon$-Lipschitz-continuous.

Next, we extend h to a (non-normalised) $(n-1)D$-Borsuk-Ulam function $G : [-1,1]^n \to [-\delta, \delta]^{n-1}$ using the same construction as in the proof of Lemma 5.4. By the same arguments, it holds that F is an odd function and it is Lipschitz-continuous with parameter $2(n-1)^2(N-1)\varepsilon + \delta \leq 2n^2 N\varepsilon$. Furthermore, any $x \in \partial([-1,1]^n)$ with $\|G(x)\|_\infty < \delta$ yields a solution to the original $(n-1)D$-Tucker instance.

On a high-level, the rest of the reduction, which is the most interesting part, works by embedding G in an n-dimensional subspace of $[-1,1]^{n+1}$ and then carefully extending it to the rest of $[-1,1]^{n+1}$ in a monotonic way. This n-dimensional subspace is

$$\mathcal{D} := \left\{ x \in [-1,1]^{n+1} \,\middle|\, \sum_{i=1}^{n+1} x_i = 0 \right\}.$$

It has the nice property that for any $x, y \in \mathcal{D}$, if $x \leq y$, then $x = y$.

Step 2: Embedding into a function $\mathcal{D} \to [-\delta, \delta]^{n-1}$. We begin by defining a slightly modified version of G. The function $\widehat{G} : [-1,1]^n \to [-\delta, \delta]^{n-1}$ is defined as follows

$$\widehat{G}(x) = G((n+1) \cdot T_{\frac{1}{n+1}}(x))$$

where $T_r : [-1,1]^n \to [-r, r]^n$ denotes truncation to $[-r, r]$ in every coordinate. It is easy to see that $\widehat{G}$ remains an odd function and that it is Lipschitz-continuous with parameter $(n+1) \cdot 2n^2 N\varepsilon \leq 2(n+1)^3 N\varepsilon$. Furthermore, it holds that any $x \in [-1,1]^n \setminus (-\frac{1}{n+1}, \frac{1}{n+1})^n$ with $\|\widehat{G}(x)\|_\infty < \delta$ yields a solution to the $(n-1)D$-Borsuk-Ulam instance G.

Next, we embed $\widehat{G}$ into $\mathcal{D}$. Let $H : \mathcal{D} \to [-\delta, \delta]^{n-1}$ be defined by $H(x) = H(x', x_{n+1}) = \widehat{G}(x')$. Note that H remains an odd function and is also $2(n+1)^3 N\varepsilon$-Lipschitz-continuous. Any $x \in \mathcal{D}$ with $\|H(x)\|_\infty < \delta$ and such that there exists $i \in [n]$ with $|x_i| \geq 1/(n+1)$, yields a solution to $\widehat{G}$ and thus to the original $(n-1)D$-Tucker instance.

Step 3: Extending to a function $[-1,1]^{n+1} \to [-1,1]^{n-1}$. In the next step, we extend H to a function $F' : [-1,1]^{n+1} \to [-1,1]^{n-1}$. For $x \in [-1,1]^{n+1}$, we let $S(x) = \sum_{i=1}^{n+1} x_i \in [-(n+1), n+1]$ and $\Pi(x) = x - \langle x, \mathbb{1}_{n+1} \rangle \cdot \mathbb{1}_{n+1}/(n+1) = x - S(x) \cdot \mathbb{1}_{n+1}/(n+1) \in \mathcal{D}$ denote the orthogonal projection onto $\mathcal{D}$. Here $\langle \cdot, \cdot \rangle$ denotes the scalar product in $\mathbb{R}^{n+1}$, and $\mathbb{1}_{n+1} \in \mathbb{R}^{n+1}$ is the all-ones vector. F' is defined as follows:

$$F'(x) = \left(1 - \frac{|S(x)|}{n+1}\right) \cdot H(\Pi(x)) + C \cdot \frac{S(x)}{n+1} \cdot \mathbb{1}_{n-1}$$

where $C = 1 + 2(n+1)^4 N\varepsilon$. Note that $F'(x) \in [-1,1]^{n-1}$ for all $x \in [-1,1]^{n+1}$. It is easy to check that F' is an odd function by using the fact that $S(-x) = -S(x)$, $\Pi(-x) = -\Pi(x)$ and $h(-x) = -h(x)$.

It is easy to see that F' is continuous. Let us determine an upper bound on its Lipschitz parameter. For any $x, y \in [-1,1]^{n+1}$ we have

$$\begin{aligned}
\|F'(x) - F'(y)\|_\infty &\leq \frac{||S(x)| - |S(y)||}{n+1} \|H(\Pi(x))\|_\infty + (1 - |S(y)|/(n+1)) \|H(\Pi(x)) - H(\Pi(y))\|_\infty \\
&\quad + C|S(x) - S(y)|/(n+1) \\
&\leq \|x - y\|_\infty + 2(n+1)^3 N\varepsilon \sqrt{n+1} \|x - y\|_\infty + C\|x - y\|_\infty \\
&\leq (2(n+1)^4 N\varepsilon + C + 1)\|x - y\|_\infty
\end{aligned}$$

where we used $|S(x) - S(y)| \leq (n+1)\|x - y\|_\infty$ and $\|\Pi(x) - \Pi(y)\|_\infty \leq \|x - y\|_2 \leq \sqrt{n+1}\|x - y\|_\infty$. Thus, F' is Lipschitz-continuous with Lipschitz parameter $2(n+1)^4 N\varepsilon + C + 1 = 4(n+1)^4 N\varepsilon + 2$.

Let us now show that F' is monotone. For this, it is enough to show that $F'(x) \leq F'(y)$ for all $x, y \in [-1,1]^{n+1}$ with $x \leq y$ and $S(x) \geq 0$, $S(y) \geq 0$. Indeed, since F' is odd, this implies that the statement also holds if $S(x) \leq 0$ and $S(y) \leq 0$. Finally, if $S(x) \leq 0$ and $S(y) \geq 0$, then there exists z with $x \leq z \leq y$ and $S(z) = 0$, which implies that $F'(x) \leq F'(z) \leq F'(y)$.

Let $x \in [-1,1]^{n+1}$ be such that $S(x) \geq 0$. For any $j \in [n+1]$, $i \in [n-1]$ and $t \geq 0$ with $x_j + t \leq 1$, it holds that

$$\begin{aligned}
&F'_i(x + t \cdot e_j) - F'_i(x) \\
&= -\frac{t}{n+1} H_i(\Pi(x + t \cdot e_j)) + (1 - S(x)/(n+1))(H_i(\Pi(x + t \cdot e_j)) - H_i(\Pi(x))) + \frac{tC}{n+1} \\
&\geq -\frac{t}{n+1} - 2(n+1)^3 N\varepsilon \|\Pi(x + t \cdot e_j) - \Pi(x)\|_\infty + \frac{tC}{n+1} \\
&\geq (C - 1 - 2(n+1)^4 N\varepsilon) \cdot \frac{t}{n+1} \geq 0
\end{aligned}$$

since $C = 1 + 2(n+1)^4 N\varepsilon$. Here we used the fact that $\|\Pi(x + t \cdot e_j) - \Pi(x)\|_\infty \leq \|\Pi(x + t \cdot e_j) - \Pi(x)\|_2 \leq \|x + t \cdot e_j - x\|_2 = t$. We obtain that F' is monotone, since for any x, y with $x \leq y$ and

$S(x) \geq 0$, $S(y) \geq 0$, we can decompose $y = (\ldots((x + (y_1 - x_1) \cdot e_1) + (y_2 - x_2) \cdot e_2) \ldots) + (y_{n+1} - x_{n+1}) \cdot e_{n+1}$.

Step 4: Extending to a function $[-1,1]^{n+1} \to [-1,1]^n$. Define $F'_n : [-1,1]^{n+1} \to \mathbb{R}$ by $F'_n(x) = C_n \cdot S(x)/(n+1)$, where $C_n := (4(n+1)^4 N\varepsilon + 2)/(\min\{2, 1/\varepsilon\} - 1)$. Clearly, F'_n is an odd function, monotone and C_n-Lipschitz-continuous. Finally, we define $F : [-1,1]^{n+1} \to [-1,1]^n$ by $F(x) = T_1(F'(x))$. It is easy to check that F remains monotone and odd, because T_1 and F are monotone and odd. Furthermore, since $C \geq 1$ and $C_n \geq 1$, we have that $F(1,1,\ldots,1) = (1,1,\ldots,1)$. Thus, F is a monotone normalised nD-Borsuk-Ulam function with Lipschitz parameter $\max\{4(n+1)^4 N\varepsilon + 2, C_n\} = (4(n+1)^4 N\varepsilon + 2)/(\min\{2, 1/\varepsilon\} - 1)$.

Consider any $x \in \partial([-1,1]^{n+1})$ with $\|F(x)\|_\infty \leq \varepsilon$. Since $|F_n(x)| \leq \varepsilon$, it follows that $|S(x)| \leq (n+1)\varepsilon/C_n$. Assume that there exists $i \in [n-1]$ such that $|H_i(\Pi(x))| \geq \delta$. Then, we have

$$|F_i(x)| \geq (1 - \varepsilon/C_n)\delta - C\varepsilon/C_n \geq \delta - (C+1)\varepsilon/C_n > \delta - (\min\{2, 1/\varepsilon\} - 1)\varepsilon \geq \delta - (\delta - \varepsilon) \geq \varepsilon$$

where we used $C_n > (C+1)/(\min\{2, 1/\varepsilon\} - 1)$. But this would mean that $\|F(x)\|_\infty > \varepsilon$, a contradiction. Thus, it must be that $\|H(\Pi(x))\|_\infty < \delta$.

In order to show that $\Pi(x)$ is a solution to H, it remains to prove that there exists $i \in [n]$ such that $|[\Pi(x)]_i| \geq 1/(n+1)$. Since $x \in \partial([-1,1]^{n+1})$, there exists $j \in [n+1]$ such that $|x_j| = 1$. As a result, $|[\Pi(x)]_j| = |x_j - S(x)/(n+1)| \geq 1 - \varepsilon/C_n$. If $j < n+1$, then let $i := j$. Otherwise, if $j = n+1$, then there exists $i \in [n]$ such that $|[\Pi(x)]_i| \geq 1/n - \varepsilon/(nC_n)$, since $S(\Pi(x)) = 0$. In both cases we have found $i \in [n]$ such that $|[\Pi(x)]_i| \geq \min\{1 - \varepsilon/C_n, 1/n - \varepsilon/(nC_n)\}$. Since $C_n \geq (n+1)\varepsilon$, it follows that $|[\Pi(x)]_i| \geq 1/(n+1)$.

Thus, from any $x \in \partial([-1,1]^{n+1})$ with $\|F(x)\|_\infty \leq \varepsilon$, we can obtain a solution to the original $(n-1)D$-Tucker instance. Note that the reduction is polynomial-time and we have only used operations allowed by the gates of the arithmetic circuit. In particular, we have only used division by constants, which can be performed by multiplying by the inverse of that constant.

The reduction is black-box and any query to F can be answered by at most $2n$ queries to the labelling function ℓ of the original $(n-1)D$-Tucker instance. Note that this is a constant, since n is constant.

White-box model. We have obtained a polynomial-time many-one reduction from $(n-1)D$-Tucker to monotone normalised nD-Borsuk-Ulam for all $n \geq 2$. Since $(n-1)D$-Tucker is PPA-hard for any $n \geq 3$, we immediately obtain the PPA-hardness of monotone normalised nD-Borsuk-Ulam, and Theorem 6.5 follows.

Black-box model. The reduction we have provided only requires black-box access to the labelling function ℓ of the $(n-1)D$-Tucker instance. In more detail, any query to F can be answered by making at most $2n$ queries to ℓ. Since n is a constant, the query lower bounds from $(n-1)D$-Tucker carry over to monotone normalised nD-Borsuk-Ulam, and Theorem 6.7 follows.

7 Relations to the Robertson-Webb Query Model

The black-box query model that we used in the previous sections is the standard model used in the related literature of query complexity, where the nature of the input functions depends on the specific problems at hand. For example, for nD-Borsuk-Ulam the function F inputs points on the domain and returns other points, whereas in nD-Tucker the function ℓ inputs points and outputs their labels.

At the same time, in the literature of the cake-cutting problem, the predominant query model is in fact a more expressive query model, known as the *Robertson-Webb model (RW)* [Robertson and Webb, 1998; Woeginger and Sgall, 2007]. The RW model has been defined only for the case of *additive valuations*, and consists of the following two types of queries:

- eval queries, where the agent is given an interval $[a, b]$ and she returns her value for that interval, and

- cut queries, where the agent is given a point $x \in [0, 1]$ and a real number α, and they designate the smallest interval $[x, y]$, for which their value is exactly α.

In fact, in the literature of *envy-free* cake-cutting, the query complexity in the RW model has been one of the most important open problems [Brams and Taylor, 1996; Procaccia, 2016], with breakthrough results coming from the literature of computer science fairly recently [Aziz and Mackenzie, 2016a,b]. Since ε-Consensus-Halving and ε*-fair* cake-cutting [Brânzei and Nisan, 2017] are conceptually closely related, it would make sense to consider the query complexity of the former problem in the RW model as well.[8]

A potential hurdle in this investigation is that the RW model has not been defined for valuation functions beyond the additive case. To this end, we propose the following generalisation of the RW model that we call *Generalised Robertson-Webb model (GRW))*, which is appropriate for monotone valuation functions that are not necessarily additive. Intuitively, in the GRW model the agent essentially is given *sets of intervals* A rather than single intervals, and the queries are defined accordingly (see also Figure 4).

Definition 6 (*Generalised Robertson-Webb (GRW) Query Model*)**.** In the GRW query model, there are two types of queries:

- eval queries, where agent i is given any Lebesgue-measurable subset A of $[0, 1]$ and she returns her value $v_i(A)$ for that set, and

- cut queries, where agent i is given two disjoint Lebesgue-measurable subsets A_1 and A_2 of $[0, 1]$, an interval $I = [a, b]$, disjoint from A_1 and A_2, and a real number $\gamma \geq 0$, and she designates some $x \in I$ such that $\frac{v_i(A_1 \cup [a,x])}{v_i(A_2 \cup ([x,b])} = \gamma$, if such a point exists.

Let us discuss why this model is the most appropriate generalisation of the RW model. First, the definition of eval queries is in fact the natural extension, as the agent needs to specify her value for sets of intervals; note that in the additive case, it suffices to elicit an agent's value for only single intervals, as her value for unions of intervals is then simply the sum of the elicited values. This is not the case in general for monotone valuations, and therefore we need a more expressive eval query. We also remark that the eval query is exactly the same as a query in the black-box

[8]The ε-Consensus-Halving halving problem has in fact recently been studied under this query model as well, but in a somewhat different direction, and for agents with additive valuations [Alon and Graur, 2020]. Note that the authors do not refer to their query model as the RW model, but the queries that they use are essentially RW queries.

Figure 4: Visualisation of CUT and EVAL queries. (a) The input A to an EVAL query is denoted by the green intervals. (b) The inputs A_1 and A_2 to the CUT query are denoted by the green and yellow intervals respectively, and the interval $I = [a, b]$ is denoted in blue. The agent places a cut (if possible) at a position $x \in I$ such that her value for $A_1 \cup [a, x]$ and her value for $A_2 \cup [x, b]$ are in a specified proportion.

model, as defined in Section 3, and therefore the GRW model is stronger than the black-box query model. Brânzei and Nisan [2017] in fact studied the restriction of the RW model (for the cake-cutting problem and for additive valuations), for which only EVAL queries are allowed, and they coined this the *RW$^-$ query model*. To put our results into context, we offer the following definition of the *GRW$^-$ query model*, which is, as discussed, equivalent to the black-box query model of Section 3. By the discussion above, all of our query complexity bounds in Section 5 and Section 6 apply verbatim to the GRW$^-$ query model.

Definition 7 (*Generalised Robertson-Webb$^-$ (GRW$^-$) query model*)**.** In the GRW$^-$ query model, only EVAL queries are allowed; there agent i is given a Lebesgue-measurable subset A of $[0, 1]$ and she returns her value $v_i(A)$ for that set.

While the extension of EVAL queries from the RW model to the GRW model is relatively straightforward, the generalisation of CUT queries is somewhat more intricate. Upon closer inspection of a CUT query in the (standard) RW model for additive valuations, it is clear that one can equivalently define this query as

Given an interval $I = [a, b]$ and a real number γ, place a cut at $x \in [a, b]$ such that $\frac{v_i([a,x])}{v_i([x,b])} = \gamma$.

This is because one can easily find the value of the agent for $[a, b]$ with one EVAL query, and then for any value of α used in the standard definition of a cut query, there is an appropriate value of γ in the modified definition above, which will result in exactly the same position x of the cut and vice-versa.

The simplicity of the CUT queries in the RW model is enabled by the fact that for additive valuations, the value of any agent i for an interval I does not depend on how the remainder of the interval $[0, 1]$ has been cut. This is no longer the case for monotone valuations, as now the agent needs to specify a different value for sets of intervals. We believe that our definition of the CUT query in the GRW model is the appropriate generalisation, which captures the essence of the original cut queries in RW, but also allows for enough expressiveness to make this type of query useful for monotone valuations beyond the additive case.

Finally, we remark that for general valuations (beyond monotone), any sensible definition of cut queries seems to be too strong, in the sense that it conveys unrealistically too much information (in contrast to the RW and GRW models, where the CUT queries are intuitively "shortcuts" for binary search). For example, assume that the agent is asked to place a cut at some point x in an interval $[a, b]$, for which (a) if the cut is placed at a the agent "sees" an excess of "+" and

(b) if the cut is placed at b, the agent still "sees" an excess of "+". By the boundary conditions of the interval, there is no guarantee that a cut that "satisfies" the agent exists within that interval, and we would need to exhaustively search through the whole interval to find such a cut position, if it exists, meaning that binary search does not help us here. On the other hand, a single CUT query would either find the position or return that there is no such x within the interval.

We are now ready to state our results for the section, which we summarise in the following theorem. Qualitatively, we prove that ε-CONSENSUS-HALVING with three normalised monotone agents still has an exponential query complexity in the GRW model (with logarithmic savings compared to the black-box model), whereas for two normalised monotone agents, the problem becomes "easier" by a logarithmic factor.

Theorem 7.1. *In the Generalised Robertson-Webb model:*

- ε-CONSENSUS-HALVING *with* $n \geq 3$ *normalised monotone agents requires* $\Omega\left(\left(\frac{(L/\varepsilon)^{n-2}}{\log(L/\varepsilon)}\right)\right.$ *queries.*
- ε-CONSENSUS-HALVING *with* $n = 2$ *monotone agents can be solved with* $O(\log(L/\varepsilon))$ *queries.*

Proof. The upper bound for $n = 2$ can be obtained relatively easily, by observing that in the proof of Theorem 6.1, Step 5 and Step 15 of Algorithm 1 were obtained via binary search, using $O(\log(L/\varepsilon))$ queries, which resulted in a query complexity of $O(\log^2(L/\varepsilon))$, since these steps were executed $O(\log(L/\varepsilon))$ times. In the GRW model, we can simply replace each of those binary searches by a single CUT query (as these only apply to the monotone agents) and obtain a query complexity of $O(\log(L/\varepsilon))$. For example, Step 5 can be simulated by a CUT query where $\gamma = 1$, $A_1 = \emptyset$, $A_2 = [0, \frac{\ell+r}{2}]$, and $I = [\frac{\ell+r}{2}, 1]$.

For the lower bound when $n \geq 3$, we will show how to construct an instance of ε-CONSENSUS-HALVING with n normalised monotone agents, such that the $\Omega((L/\varepsilon)^{n-2})$ lower bound for EVAL queries still holds, but we can additionally answer any CUT query by performing at most $O(\log(L/\varepsilon))$ EVAL queries. We will again use our reduction from normalised monotone nD-BORSUK-ULAM to ε-CONSENSUS-HALVING with n normalised monotone agents, which we developed in Section 4.1.

Given a CUT query $(A_1, A_2, I = [a,b], \gamma)$, we define $\phi : [a,b] \to \mathbb{R}_{\geq 0}$ by $\phi(t) = \frac{v_i(A_1 \cup [a,t])}{v_i(A_2 \cup [t,b])}$. Note that we are looking for $t^* \in I$ such that $\phi(t^*) = \gamma$ and that ϕ can be evaluated by using two EVAL queries. Furthermore, since v_i is monotone, ϕ is non-decreasing. We begin by checking that $\frac{v_i(A_1)}{v_i(A_2 \cup [a,b])} \leq \gamma$ and $\frac{v_i(A_1 \cup [a,b])}{v_i(A_2)} \geq \gamma$. If one of these two conditions does not hold, then we can immediately answer that there is no $t \in I$ that satisfies the query. In what follows, we assume that these two conditions hold. In that case, we can query $\phi(t)$ for some $t \in I$ to determine whether the solution t^* lies in $[a,t]$ or in $[t,b]$. We denote by $J \subseteq I$ the current interval for which we know that $t^* \in J$. At the beginning, we have $J := I$. Using at most $\lceil \log_2(n+1) \rceil + 2$ queries to ϕ we can shrink J such that $J \subseteq R_j$ for some $j \in [n+1]$. Recall that $[x(A)]_i = 2(n+1) \cdot \lambda(A \cap R_i) - 1$ for any $i \in [n+1]$ and $A \in \Lambda([0,1])$. It follows that for any $t \in J$, $[x(A_1 \cup [a,t])]_i$ and $[x(A_2 \cup [t,b])]_i$ are fixed for all $i \in [n+1] \setminus \{j\}$. Furthermore, with an additional $\lceil \log_2 m \rceil + 2$ queries, we can ensure that for all $t \in J$, $[x(A_1 \cup [a,t])]_j \in [k/m, (k+1)/m]$ and $[x(A_2 \cup [t,b])]_j \in [\ell/m, (\ell+1)/m]$ for some $k, \ell \in \mathbb{Z}$. Next, with an additional $2\lceil \log_2(n+1) \rceil$ queries, we can shrink J, so that for all $t \in J$, $x(A_1 \cup [a,t])$ and $x(A_2 \cup [t,b])$ each lie in some fixed simplex of Kuhn's triangulation of the domain K_m^{n+1} (defined below). In that case, by our construction below, it will hold that

$v_i(x(A_1 \cup [a,t]))$ and $v_i(x(A_2 \cup [t,b]))$ can be expressed as an affine function of $t \in J$ and thus we can exactly determine the value of t^*. In order for this to hold, we will ensure that our normalised monotone nD-BORSUK-ULAM function is piecewise linear. Furthermore, we will pick $m = \lceil 2nL/\varepsilon \rceil$, and thus we have used $2(3\lceil \log_2(n+1) \rceil + 2 + \lceil \log_2(\lceil 2nL/\varepsilon \rceil) \rceil + 2)$ EVAL queries to answer one CUT query. Note that this expression is $O(\log(L/\varepsilon))$, since n is constant.

Consider a normalised monotone nD-BORSUK-ULAM function $F : [-1,1]^{n+1} \to [-1,1]^n$ with Lipschitz parameter $L \geq 3$ and some $\varepsilon \in (0,1)$. We first discretize the domain to be $K_m^{n+1} := \{-1, -(m-1)/m, \dots, -1/m, 0, 1/m, 2/m, \dots, (m-1)/m, 1\}^{n+1}$ where $m = \lceil 2nL/\varepsilon \rceil$. We let $f : K_m^{n+1} \to [-1,1]^n$ be defined by $f(x) = F(x)$. Note that f is antipodally anti-symmetric ($f(-x) = -f(x)$ for all $x \in K_m^{n+1}$), monotone ($f(x) \leq f(y)$ whenever $x \leq y$) and $f(1,1,\dots,1) = (1,1,\dots,1)$. Furthermore, any $x \in \partial(K_m^{n+1})$ with $\|f(x)\|_\infty \leq \varepsilon$ yields a solution to the original instance F. We extend f back to a function $\widehat{f} : [-1,1]^{n+1} \to [-1,1]^n$ by using Kuhn's triangulation on the grid K_m^{n+1} and interpolating (see Appendix C for a description of the triangulation and interpolation). By the arguments presented in Appendix C, it holds that $\widehat{f}$ is a continuous, monotone, odd function, and $\widehat{f}(1,1,\dots,1) = (1,1,\dots,1)$. Furthermore, if x_i is fixed for all $i \in [n+1] \setminus \{j\}$ and x_j is constrained such that x lies in some fixed simplex σ of the Kuhn's triangulation, then $\widehat{f}(x)$ can be expressed as a linear affine function of x_j.

Let us now determine the Lipschitz parameter of $\widehat{f}$. Consider any simplex $\sigma = \{y^0, y^1, \dots, y^{n+1}\}$ of the Kuhn triangulation of K_m^{n+1}. Consider any $x \in [0,1]^{n+1}$ that lies in σ and any $j \in [n+1]$ and $t \in [-1/m, 1/m]$ such that $x + t \cdot e_j$ also lies in σ. Then, the interpolation (as defined in Appendix C) yields

$$\begin{aligned}
\|\widehat{f}(x) - \widehat{f}(x + t \cdot e_j)\|_\infty = \|t \cdot m \cdot f(y^j) - t \cdot m \cdot f(y^{j-1})\|_\infty &\leq |t| \cdot m \cdot \|F(y^j) - F(y^{j-1})\|_\infty \\
&\leq |t| \cdot m \cdot L \cdot \|y^j - y^{j-1}\|_\infty \\
&\leq L \cdot |t|.
\end{aligned}$$

It is easy to check that this implies that $\widehat{f}$ is $(n+1)L$-Lipschitz-continuous on the simplex σ. By a simple argument, it follows that $\widehat{f}$ is $(n+1)L$-Lipschitz-continuous on $[-1,1]^{n+1}$ (see e.g., the proof in Section 5.1).

Now consider any simplex $\sigma = \{y^0, y^1, \dots, y^{n+1}\}$ such that there exists $i \in [n+1] \cup \{0\}$ with $\|f(y^i)\|_\infty > \varepsilon$. Since $\widehat{f}(y^i) = f(y^i)$, and $\widehat{f}$ is $(n+1)L$-Lipschitz-continuous, it follows that for any $x \in [-1,1]^{n+1}$ that lies in σ we have

$$\|\widehat{f}(x)\|_\infty \geq \|\widehat{f}(y^i)\|_\infty - nL\|x - y^0\|_\infty > \varepsilon - nL/m \geq \varepsilon/2$$

where we used $m \geq 2nL/\varepsilon$. Thus, for any $x \in \partial([-1,1]^{n+1})$ with $\|\widehat{f}(x)\|_\infty \leq \varepsilon/2$, it must hold that both $\|f(y^0)\|_\infty \leq \varepsilon$ and $\|f(y^{n+1})\|_\infty \leq \varepsilon$, where $\sigma = \{y^0, y^1, \dots, y^{n+1}\}$ is the Kuhn simplex containing x. However, since $x \in \partial([-1,1]^{n+1})$, it follows that y^0 or y^{n+1} lies on the boundary of K_m^{n+1}. This means that we have obtained a solution to the original nD-BORSUK-ULAM function F.

Since the parameters for $\widehat{f}$ are $L' = nL$ and $\varepsilon' = \varepsilon/2$, and n is constant, the query lower bound for F carries over to $\widehat{f}$. $\square$

8 Conclusion and Future Directions

In this paper, we managed to completely settle the computational complexity of the ε-Consensus-Halving problem for a constant number of agents with either general or monotone valuation functions. We also studied the query complexity of the problem and we provided exponential lower bounds corresponding to our hardness results, and polynomial upper bounds corresponding to our polynomial-time algorithms. We also defined an appropriate generalisation of the Robertson-Webb query model for monotone valuations and we showed that our bounds are qualitatively robust to the added expressiveness of this model. The main open problem associated with our work is the following.

> What is the computational complexity and the query complexity of ε-Consensus-Halving with a *constant* number of agents and *additive valuations*?

Our approach in this paper prescribes a formula for answering this question: One can construct a black-box reduction to this version of ε-Consensus-Halving from a computationally-hard problem like *nD*-Tucker, for which we also know query complexity lower bounds, and obtain answers to both questions at the same time. Alternatively, one might be able to construct polynomial-time algorithms for solving this problem; concretely, attempting to do that for three agents with additive valuations would be the natural starting step, as this is the first case for which the problem becomes computationally hard for agents with monotone valuations.

Another line of work would be to study the query complexity of the related fair cake-cutting problem using the GRW model that we propose. In fact, while the fundamental existence results for the problem (e.g., see [Su, 1999]) actually apply to quite general valuation functions, most of the work in computer science has restricted its attention to the case of additive valuations only, with a few notable exceptions [Caragiannis et al., 2011; Deng et al., 2012]. We believe that our work can therefore spark some interest in the study of *the query complexity of fair cake-cutting with valuations beyond the additive case*.

Acknowledgements

Alexandros Hollender is supported by an EPSRC doctoral studentship (Reference 1892947).

References

James Aisenberg, Maria Luisa Bonet, and Sam Buss. 2-D Tucker is PPA complete. *Journal of Computer and System Sciences*, 108:92–103, 2020. doi:10.1016/j.jcss.2019.09.002.

Noga Alon. Splitting necklaces. *Advances in Mathematics*, 63(3):247–253, 1987. doi:10.1016/0001-8708(87)90055-7.

Noga Alon and Andrei Graur. Efficient splitting of measures and necklaces. *arXiv preprint*, 2020. URL https://arxiv.org/abs/2006.16613.

Noga Alon and Douglas B. West. The Borsuk-Ulam Theorem and Bisection of Necklaces. *Proceedings of the American Mathematical Society*, 98(4):623–628, 1986. doi:10.2307/2045739.

Georgios Amanatidis, George Christodoulou, John Fearnley, Evangelos Markakis, Christos-Alexandros Psomas, and Eftychia Vakaliou. An improved envy-free cake cutting protocol for four agents. In *Proceedings of the 11th International Symposium on Algorithmic Game Theory (SAGT)*, pages 87–99, 2018. doi:10.1007/978-3-319-99660-8_9.

A. K. Austin. Sharing a cake. *The Mathematical Gazette*, 66(437):212–215, 1982. doi:10.2307/3616548.

Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for any number of agents. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 416–427, 2016a. doi:10.1109/FOCS.2016.52.

Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for four agents. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, pages 454–464, 2016b. doi:10.1145/2897518.2897522.

Siddharth Barman and Nidhi Rathi. Fair cake division under monotone likelihood ratios. In *Proceedings of the 21st ACM Conference on Economics and Computation (EC)*, pages 401–437, 2020. doi:10.1145/3391403.3399512.

Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The Relative Complexity of NP Search Problems. *Journal of Computer and System Sciences*, 57(1):3–19, 1998. doi:10.1145/225058.225147.

Karol Borsuk. Drei Sätze über die n-dimensionale euklidische Sphäre. *Fundamenta Mathematicae*, 20:177–190, 1933. doi:10.4064/fm-20-1-177-190.

Steven J. Brams and Alan D. Taylor. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996. doi:10.1017/CBO9780511598975.

Simina Brânzei and Noam Nisan. The query complexity of cake cutting. *arXiv preprint*, 2017. URL `https://arxiv.org/abs/arXiv:1705.02946`.

Simina Brânzei and Noam Nisan. Communication complexity of cake cutting. In *Proceedings of the 20th ACM Conference on Economics and Computation (EC)*, pages 525–525, 2019. doi:10.1145/3328526.3329644.

Ioannis Caragiannis, John K. Lai, and Ariel D. Procaccia. Towards more expressive cake cutting. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 127–132, 2011. doi:10.5591/978-1-57735-516-8/IJCAI11-033.

Xi Chen and Xiaotie Deng. Matching algorithmic bounds for finding a Brouwer fixed point. *Journal of the ACM*, 55(3):13:1–13:26, 2008. doi:10.1145/1379759.1379761.

Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM*, 56(3):14:1–14:57, 2009. doi:10.1145/1516512.1516516.

Constantinos Daskalakis and Christos H. Papadimitriou. Continuous local search. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 790–804, 2011. ISBN 9780898719932. doi:10.1137/1.9781611973082.62.

Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009. doi:10.1137/070699652.

Argyrios Deligkas, John Fearnley, Themistoklis Melissourgos, and Paul G. Spirakis. Computing exact solutions of consensus halving and the borsuk-ulam theorem. *J. Comput. Syst. Sci.*, 117:75–98, 2021. doi:10.1016/j.jcss.2020.10.006. URL `https://doi.org/10.1016/j.jcss.2020.10.006`.

Xiaotie Deng, Qi Qi, Amin Saberi, and Jie Zhang. Discrete fixed points: Models, complexities, and applications. *Mathematics of Operations Research*, 36(4):636–652, 2011. doi:10.1287/moor.1110.0511.

Xiaotie Deng, Qi Qi, and Amin Saberi. Algorithmic solutions for envy-free cake cutting. *Operations Research*, 60(6):1461–1476, 2012. doi:10.1287/opre.1120.1116.

Kousha Etessami and Mihalis Yannakakis. On the complexity of Nash equilibria and other fixed points. *SIAM Journal on Computing*, 39(6):2531–2597, 2010. doi:10.1137/080720826.

Aris Filos-Ratsikas and Paul W. Goldberg. Consensus Halving is PPA-complete. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 51–64, 2018. doi:10.1145/3188745.3188880.

Aris Filos-Ratsikas and Paul W. Goldberg. The Complexity of Splitting Necklaces and Bisecting Ham Sandwiches. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 638–649, 2019. doi:10.1145/3313276.3316334.

Aris Filos-Ratsikas, Søren Kristoffer Still Frederiksen, Paul W. Goldberg, and Jie Zhang. Hardness Results for Consensus-Halving. In *Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 24:1–24:16, 2018. doi:10.4230/LIPIcs.MFCS.2018.24.

Aris Filos-Ratsikas, Alexandros Hollender, Katerina Sotiraki, and Manolis Zampetakis. Consensus-Halving: Does it Ever Get Easier? In *Proceedings of the 21st ACM Conference on Economics and Computation (EC)*, pages 381–399, 2020. doi:10.1145/3391403.3399527.

Aris Filos-Ratsikas, Alexandros Hollender, Katerina Sotiraki, and Manolis Zampetakis. A topological characterization of modulo-p arguments and implications for necklace splitting. In *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2615–2634, 2021. doi:10.1137/1.9781611976465.155.

Charles H. Goldberg and Douglas B. West. Bisection of Circle Colorings. *SIAM Journal on Algebraic Discrete Methods*, 6(1):93–106, 1985. doi:10.1137/0606010.

Michelangelo Grigni. A Sperner lemma complete for PPA. *Information Processing Letters*, 77(5–6): 255–259, 2001. doi:10.1016/S0020-0190(00)00152-6.

Charles R. Hobby and John R. Rice. A moment problem in L1 approximation. *Proceedings of the American Mathematical Society*, 16(4):665–670, 1965. doi:10.2307/2033900.

Ilan Komargodski, Moni Naor, and Eylon Yogev. White-box vs. black-box complexity of search problems: Ramsey and graph property testing. *Journal of the ACM*, 66(5):1–28, 2019. doi:10.1109/FOCS.2017.63.

Jiří Matoušek. *Using the Borsuk-Ulam theorem: lectures on topological methods in combinatorics and geometry*. Springer Science & Business Media, 2008. doi:10.1007/978-3-540-76649-0.

Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991. doi:10.1016/0304-3975(91)90200-L.

Frédéric Meunier. Simplotopal maps and necklace splitting. *Discrete Mathematics*, 323(28):14–26, 2014. doi:10.1016/j.disc.2014.01.008.

Jerzy Neyman. Un théorème d'existence. *CR Acad. Sci. Paris*, 222:843–845, 1946.

Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994. doi:10.1016/S0022-0000(05)80063-7.

Ariel D. Procaccia. Cake cutting algorithms. In *Handbook of Computational Social Choice*, pages 311–330. Cambridge University Press, 2016. ISBN 978-1-107-06043-2.

Jack M. Robertson and William A. Webb. Approximating fair division with a limited number of cuts. *Journal of Combinatorial Theory, Series A*, 72(2):340–344, 1995. doi:10.1016/0097-3165(95)90073-X.

Jack M. Robertson and William A. Webb. *Cake-cutting algorithms: Be fair if you can*. CRC Press, 1998. ISBN 9781568810768.

Forest W. Simmons and Francis E. Su. Consensus-halving via theorems of Borsuk-Ulam and Tucker. *Mathematical social sciences*, 45(1):15–25, 2003. doi:10.1016/S0165-4896(02)00087-2.

Francis E. Su. Rental harmony: Sperner's lemma in fair division. *The American Mathematical Monthly*, 106(10):930–942, 1999. doi:10.1080/00029890.1999.12005142.

Albert W. Tucker. Some topological properties of disk and sphere. In *Proceedings of the First Canadian Math. Congress, Montreal*, pages 286–309. University of Toronto Press, 1945.

Gerhard J. Woeginger and Jiří Sgall. On the complexity of cake cutting. *Discrete Optimization*, 4(2): 213–220, 2007. doi:10.1016/j.disopt.2006.07.003.

APPENDIX

A ε-Consensus-Halving for piece-wise constant valuations

As we mentioned in the introduction, the valuation functions studied in previous work [Filos-Ratsikas and Goldberg, 2018, 2019] are piece-wise constant valuations (i.e., step functions) which are given explicitly as part of the input, with each agent specifying the end-points and the height of each step. For a constant number of agents n, this case is solvable in polynomial-time, as illustrated in Figure 1.

Lemma A.1. ε-Consensus-Halving *with a constant number n of agents and piece-wise constant valuations (given explicitly as part of the input) is solvable in polynomial time.*

Proof. The proof follows closely that of Lemma 15 in [Filos-Ratsikas et al., 2020]; here we provide the main proof idea, and we refer the reader to that paper for a fully detailed proof.

First, we partition the interval $[0, 1]$ into *regions*, via a set of points $t_1, \ldots, t_m$ such that the density of the valuation function of each agent is constant within each region. Let $T = [t_j, t_{j+1}]$ denote an arbitrary region, and let $v_i(T)$ denote the constant value of the density of agent i in region T (i.e., the height of the step). Since the valuations are provided explicitly (or equivalently, are polynomial in the other input parameters), this process will result in a polynomial number of such regions.

We will be interested in the regions that will contain cuts, and then we will find the appropriate positions of those cuts using linear programming. First, it is not hard to see that a solution in which some region contains more than 1 cut can be transformed into a solution in which every region contains at most 1 cut, by appropriately "merging" and "shifting" sets of cuts within the region. Therefore, we can assume that each region either contains a cut or it doesn't. For any $k = 1, \ldots, n$, we can consider all the possible ways of distributing the k cuts to the regions, such that no region receives more than one cut; since n is a constant, this can be done in polynomial time.

Let x_T be the position of the cut in region $T = [t_i, t_{i+1}]$. This means that for agent i, the "left sub-region" $[t_i, x_T]$ receives one label (say "$+$"), whereas the "right sub-region" $[x_T, t_{i+1}]$ receives the other label (say "$-$"), resulting in values $v_i(T) \cdot [t_i, x_T]$ and $v_i(T) \cdot [x_T, t_{i+1}]$ for the agent respectively. If there is not cut in region T, then the whole interval receives the same label. Now, we can consider the set of cuts at positions x_T in each of the regions that contain cuts, and the corresponding partitioning of $[0, 1]$ into intervals, labelled "$+$" and "$-$"; without loss of generality we can assume that this labelling is alternating. From there, we can devise a linear program for each value of k, where we aim to minimise z, subject to $|v_i(\mathcal{I}^+) - v_i(\mathcal{I}^-)| \leq z$ (which can be transformed into a set of linear constraints), plus additional linear constraints for the positions of the cuts. Since a solution always exists, and since we consider all values of $k \in [1, n]$, one of the linear programs will terminate with $z = 0$, and therefore we can find an *exact* solution to the problem. □

B Formal definition of the input model

In Section 3, we mentioned that in the white-box model, the agents' valuations v_i are accessed via *polynomial-time algorithms*, which are given explicitly as part of the input. We provide the precise definition of the input model in this section. Formally, we assume that valuations are computed by *Turing machines*. The input to the Turing machine is a list of intervals, and the Turing machine outputs the value of the union of these intervals.

Definition 8 (ε-CONSENSUS-HALVING *(white-box model), formal definition*)**.** For any constant $n \geq 1$, and polynomial p, the problem ε-CONSENSUS-HALVING *with n agents* is defined as follows:

- **Input:** $\varepsilon > 0$, the Lipschitz parameter L, Turing machines $v_1, \ldots, v_n$
- **Output:**
  - A partition of $[0,1]$ into two sets of intervals $\mathcal{I}^+$ and $\mathcal{I}^-$ such that for each agent i, it holds that $|v_i(\mathcal{I}^+) - v_i(\mathcal{I}^-)| \leq \varepsilon$, using at most n cuts.
  - A *violation* of Lipschitz-continuity for one of the valuations.
  - An input X on which one of the Turing machines does not terminate in at most $p(|X|)$ steps.
  - (Optional, for monotone valuations only). A *violation* of monotonicity for one of the valuations.

Note that for all solution types except the first one, a solution can be a union of an arbitrary number of intervals, i.e., not obtained using at most n cuts. Note that the violation solutions are there to ensure that the problem is contained in TFNP. They are irrelevant for our hardness results, which also hold for the promise version of the problem where we are promised that the input does not violate any of the conditions.

C The Kuhn Triangulation

Let $D_m := \{0, 1/m, 2/m, \ldots, m/m\}$. Kuhn's triangulation is a standard way to triangulate a grid D_m^n. Every $x \in (D_m \setminus \{1\})^n$ is the base of the cube containing all vertices $\{y : y_i \in \{x_i, x_i + 1/m\}\}$. Every such cube is subdivided into $n!$ n-dimensional simplices as follows: for every permutation π of $[n]$, $\sigma = \{y^0, y^1, \ldots, y^n\}$ is a simplex, where $y^0 = x$ and $y^i = y^{i-1} + \frac{1}{m} e_{\pi(i)}$ for all $i \in [n]$ (where e_i is the ith unit vector).

It is easy to see that Kuhn's triangulation has the following properties:

- For any simplex $\sigma = \{z^1, \ldots, z^k\}$ it holds that $\|z^i - z^j\|_\infty \leq 1/m$ for all i, j, and there exists a permutation π of $[k]$ such that $z^{\pi(1)} \leq \cdots \leq z^{\pi(k)}$ (component-wise).
- Given a point $x \in [0,1]^n$, we can efficiently determine the simplex that contains it as follows. First find the base y of a cube of D_m^n that contains x. Next, find a permutation π such that $x_{\pi(1)} - y_{\pi(1)} \geq \cdots \geq x_{\pi(n)} - y_{\pi(n)}$. Then, it follows that (y, π) is the simplex containing x.
- The triangulation is antipodally anti-symmetric: if $\sigma = \{y^0, y^1, \ldots, y^n\}$ is a Kuhn simplex of D_m^n, then $\overline{\sigma} = \{\overline{y^0}, \ldots, \overline{y^n}\}$ is also a simplex, where $\overline{y^i}_j = 1 - y^i_j$ for all i, j.

Using Kuhn's triangulation, a function $f : D_m^n \to [-M, M]$ can be extended to a Lipschitz-continuous function $\widehat{f} : [0,1]^n \to [-M, M]$. $\widehat{f}$ is constructed in each Kuhn simplex $\sigma =$

$\{y^0, y^1, \ldots, y^n\}$ by interpolating over the values $\{f(y^0), f(y^1), \ldots, f(y^n)\}$. In more detail, for any $x \in [0,1]^n$ that lies in simplex $\sigma = \{y^0, y^1, \ldots, y^n\}$, we let $z := m \cdot (x - y^0)$. Let π denote the permutation used to obtain σ. Note that since x lies in σ, it must be that $z_{\pi(1)} \geq z_{\pi(2)} \geq \cdots \geq z_{\pi(n)}$. Then, it holds that $x = \sum_{i=0}^{n} \alpha_i y^i$, where $\alpha_0 = 1 - z_{\pi(1)}$, $\alpha_n = z_{\pi(n)}$, and $\alpha_i = z_{\pi(i)} - z_{\pi(i+1)}$ for $i \in [n-1]$. We define

$$\widehat{f}(x) := \sum_{i=0}^{n} \alpha_i f(y^i).$$

It is easy to check that the function $\widehat{f} : [0,1]^n \to [-M, M]$ thus obtained is continuous. Indeed, if x lies on a common face of two Kuhn simplices, then the value $\widehat{f}(x)$ obtained by interpolating in either simplex is the same. It can be shown that $\widehat{f}$ is Lipschitz-continuous with Lipschitz parameter $2M \cdot n \cdot m$ with respect to the ℓ_∞-norm.

Furthermore, if f is antipodally anti-symmetric, i.e., $f(\overline{x}) = -f(x)$ for all $x \in D_m^n$, then so is $\widehat{f}$, i.e., $\widehat{f}(\overline{x}) = -\widehat{f}(x)$ for all $x \in [0,1]^n$.

Finally, if f is monotone, i.e., $f(x) \leq f(y)$ for all $x, y \in D_m^n$ with $x \leq y$, then so is $\widehat{f}$, i.e., $\widehat{f}(x) \leq \widehat{f}(y)$ for all $x, y \in [0,1]^n$ with $x \leq y$. Consider any point $x \in [0,1]^n$ that lies in some simplex $\sigma = \{y^0, y^1, \ldots, y^n\}$. Then for any $j \in [n]$ and $t \geq 0$ such that $x + t \cdot e_j$ lies in the simplex σ, we have

$$\widehat{f}(x + t \cdot e_j) - \widehat{f}(x) = tmf(y^j) - tmf(y^{j-1}) \geq 0$$

since $y^j \geq y^{j-1}$ and f is monotone. Using this it is easy to show that $\widehat{f}$ is monotone within any simplex σ, since for any $x \leq y$ in σ we can construct a path that goes from x to y (and lies in σ) that only uses the positive cardinal directions. Since monotonicity holds for any segment of the path, it also holds for x and y. Finally, for any $x \leq y$ that lie in different simplices, we can just use the straight path that goes from x to y, and the fact that $\widehat{f}$ is monotone in each simplex that we traverse.