Structural Results for Total Search Complexity Classes with Applications to Game Theory and Optimisation

Alexandros Hollender

St Anne's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2021

Acknowledgements

It is hard for me to imagine what this thesis would have looked like if I had embarked on this journey without my supervisor Paul Goldberg. It was Paul who first introduced me to the wonderful world of TFNP and who accompanied my (slow and clumsy) first steps exploring it. His careful guidance – letting me discover the topic at my own pace to develop my intuition, while also being available to meet as often and for as long as needed – made the research presented in this thesis possible and allowed me to grow into a confident researcher. Besides Paul, I also want to express my gratitude to my co-supervisor Elias Koutsoupias, who – although we did not end up working together during my thesis (due to my obsession with the topic of TFNP) – was always available for advice and support.

I could not possibly write these acknowledgements without mentioning my third – unofficial – mentor Aris Filos-Ratsikas. Over the years, Aris, in addition to his friendship, has also continuously offered me advice on all matters related to academia and research. He has introduced me to most of my co-authors and it is safe to say that at least half of this thesis would not exist without these collaborations. This is also a good place to thank all of my collaborators for the very enjoyable meetings: Alexandros, Argyris, Aris, Ayumi, Diogo, Francisco, Giorgos, Giorgos, Jiarui, John, Kasper, Katerina, Kristoffer, Manolis, Ninad, Pasin, Paul, Philip, Rahul, Warut, and Yiannis.

I am grateful to my examiners Rahul Santhanam and Kousha Etessami for carefully reading the thesis and providing comments and suggestions that helped improve its presentation and content, and for making the viva a very enjoyable experience. I also thank Standa Živný who, along Rahul, served as an examiner for the internal "DPhil milestones" of the department. For proofreading various parts of this thesis and offering feedback, I thank Ninad, Aris, and Chetan. I would also like to express my appreciation for the staff at the department and at my college, who were always very helpful with various administrative tasks. I received generous financial support from the EPSRC, the Computer Science department, and St Anne's college.

For making my time in Oxford enjoyable, I am deeply grateful to Ninad, Francisco, Rhea, Aris, Philip, Giorgos, Andrius, Jacob, Varun, Alex, Jiarui, Edwin, Igor, Jan, Matthew, Paolo, and all those that I forgot to mention by name here. For their friendship over multiple decades, I thank Giorgos and Kostas. Finally, none of this would have been possible without the love and support from my parents and my brother.

Abstract

While the celebrated theory of NP-completeness has been very successful in explaining the intractability of many interesting problems, there still remain many natural and seemingly hard problems that are not known to be NP-hard. Several of these problems lie in the class of total NP search problems (TFNP), namely the class of NP search problems that always have a solution. Importantly, these problems cannot be NP-hard unless NP = co-NP. Notable examples of TFNP problems include Factoring (given a natural number, compute its prime factorisation) and Nash (given a game, compute a mixed Nash equilibrium). In order to shed light on the complexity of these problems, researchers have attempted to classify them in various subclasses of TFNP such as PPAD, PPA, PPP, PLS, and CLS. A celebrated result in this line of research is the PPAD-completeness of Nash, yielding strong evidence that the problem is not polynomial-time solvable.

In this thesis we provide new structural results for TFNP subclasses and show how they can be used to classify natural problems arising from application areas such as game theory and optimisation. In the first part of this work, we construct more powerful tools for proving membership in PPAD, as well as PPAD-hardness. We directly apply these tools to show that the Hairy Ball theorem from topology ("you can't comb a hairy ball flat without creating a cowlick"), as well as the equilibrium computation problem in First Price Auctions with subjective priors, are both PPAD-complete.

In the second part of this thesis, we present our main result: the collapse CLS = PPAD ∩ PLS. We prove this surprising collapse by exhibiting the first non-artificial PPAD ∩ PLS-complete problem – a problem arising naturally from the famous gradient descent algorithm. Our result puts PPAD ∩ PLS on the map as a TFNP subclass that captures the complexity of natural problems.

In the third and final part, we provide various structural results for the classes PPA-k (corresponding to arguments modulo k), including the first topological characterisations of these classes. As a direct application, we prove that Necklace-Splitting with k thieves – a notorious problem in combinatorics and fair division – lies in PPA-k under Turing reductions.

Contents

List of Figures

Chapter 1

Introduction

1.1 Total Search Problems

Some of the most celebrated milestones in Game Theory and Economics are mathematical existence results, such as:

- Nash's theorem [Nas50]: every finite game in strategic form has a mixed Nash equilibrium.
- The Arrow-Debreu theorem [AD54]: a market equilibrium always exists in the Arrow-Debreu market model.
- Envy-free cake-cutting [Str80]: it is always possible to cut a cake fairly.

There is something fascinating about these existential results, which essentially assert that a desired solution is mathematically guaranteed to exist.

However, in most situations *existence is not enough*. Instead, we would like to be able to *find* this desired solution. This is perhaps most apparent in the envy-free cake-cutting example: clearly we want to *construct* a fair division of the cake. The Nash and market equilibrium examples are a bit different in that sense. One could argue that existence here is interesting by itself, since it shows that every game or market admits a stationary state in which it should (in theory) lie. But if one accepts that these solution concepts indeed correctly represent the expected outcome in these settings, then being able to *predict* the outcome becomes a goal in itself with major economic and even societal ramifications.

This point becomes even more apparent when one considers more fundamental existential results, such as:

- every natural number has a prime factorisation.
- every continuous function defined on a compact domain has a minimum.

Of course, these existence results are mathematically interesting. But it is the ability to *find* a prime factorisation that would threaten the security of various cryptographic systems used in practice. It is the ability to *approach* a minimum of a function through gradient descent that underlies many of the advances on neural networks in Machine Learning.

Computer Science, and more specifically the Theory of Computation, provides a framework to study these questions in a rigorous mathematical way by replacing the words *find/construct/predict* by the word *compute*. Within this formal framework, one can study questions such as "can this task be solved by a computer?" and then "how fast can this task be solved by a computer?". One of the greatest success stories of this approach has been the notion of NP-completeness, introduced by Cook, Levin and Karp around 50 years ago. By proving that a problem is NP-complete, one can provide strong evidence that the problem cannot be solved efficiently, namely that it is not polynomial-time solvable. Over the years, thousands of problems have been shown to be NP-complete, thus establishing that they are inherently hard to solve.

The motivating examples we mentioned above naturally give rise to corresponding computational problems. FACTORING is the problem: given a natural number, compute its prime factorisation. NASH is the problem: given a normal-form game, compute a Nash equilibrium. Despite intense effort, no polynomial-time algorithms are known for these problems. It is natural therefore to try to provide evidence that the problems are indeed intractable.

Unfortunately, the theory of NP-completeness cannot be used in this case. To be more precise, it seems very unlikely that any of these problems could be NP-hard. The fundamental reason for this is that they lie in the class of total NP search problems (TFNP). TFNP problems are search problems with efficiently-checkable solutions that are also *total*: for any instance, there always exists a solution. So, why are TFNP problems very unlikely to be NP-hard? The answer is that this would imply an extremely surprising result that most complexity theorists believe is highly unlikely to hold. Formally: if some TFNP problem is NP-hard, then by a simple argument [MP91], NP = co-NP. While this statement is weaker than P = NP, it is equally *not* expected to hold. This basic fact – that hard TFNP problems are in a very strong sense "NP-intermediate" – provides TFNP's strong theoretical appeal.

What this means for our problems of interest is that the theory of NP-hardness is of no help in our quest to show that they are computationally hard. Furthermore, TFNP-hardness is also not believed to be an option, because it seems unlikely that a single problem could capture all the proofs of totality. Thus, a different approach is needed in order to provide evidence that these very natural problems are indeed hard to solve.

The computational landscape inside TFNP

Almost 30 years ago, Papadimitriou [Pap94] initiated an investigation of the computational landscape *inside* TFNP. In particular, he proposed the following approach: classify TFNP problems into syntactic subclasses depending on their proof of totality and then attempt to prove that these problems are complete for these subclasses. Some of the main subclasses that have been studied to this day include:

- PPAD: this subclass contains all TFNP problems for which totality can be proved by a directed path-following argument, or equivalently by Brouwer's fixed point theorem. Since Nash's theorem can be proved by using this topological theorem, it follows that Nash lies in PPAD.

- PPA: problems where totality follows from a parity argument, such as the Handshaking lemma. Equivalently, PPA contains all problems where totality can be proved by using another famous topological result: the Borsuk-Ulam theorem.

- PPP: problems where totality follows from the pigeonhole principle. Various problems related to Cryptography lie in PPP.

- PLS: problems where totality follows from a local search argument or equivalently a potential argument. Many local search optimisation problems lie in PLS.

- CLS: problems where totality follows from a *continuous* local search argument. This class was introduced by Daskalakis and Papadimitriou [DP11] as a more natural counterpart to PPAD∩PLS. In particular, this class contains all problems that can be solved by gradient descent.

The known relationships between these classes are shown in Figure 1.1, which also includes the classes PPA-k corresponding to arguments modulo k, where PPA-2 = PPA.

Perhaps the most celebrated result in this line of research has been the PPAD-completeness of Nash [DGP09; CDT09]. This result tells us that computing a Nash equilibrium is as hard as solving any other PPAD problem, i.e., essentially any problem whose totality can be proved by using Brouwer's fixed point theorem. Thus, just like NP-hardness (albeit in a weaker sense), this provides evidence that the problem is not polynomial-time solvable. Since this first PPAD-completeness result, many problems related to Game Theory and Economics have been shown to be PPAD-complete, including problems related to Arrow-Debreu market equilibria [CDDT09] and some versions of envy-free cake-cutting [DQS12]. PLS has also been very successful in capturing the complexity of various local optimisation problems [JPY88; Kre89; Sch91]. Recently, problems from Fair Division and Cryptography

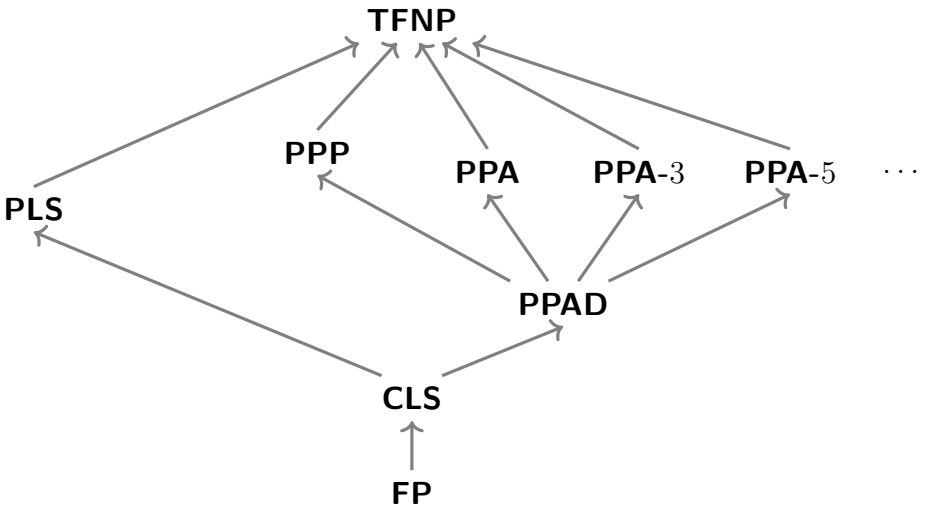


Figure 1.1: Some of the main subclasses of TFNP. Arrows are used to denote containment. For example, CLS is contained in PLS and in PPAD.

have been shown to be the first natural[1] complete problems for PPA [FG18] and PPP [SZZ18] respectively. However, the complexity of many natural TFNP problems remains open. For example, FACTORING has partially been related to PPP and PPA [Jeř16], but its exact classification in the TFNP landscape is still unknown.

In this thesis, we provide various structural results for TFNP subclasses and present applications of these results to problems from Game Theory and Optimisation. Our main contribution is the surprising collapse $\mathsf{CLS} = \mathsf{PPAD} \cap \mathsf{PLS}$ and a characterisation of this class as the set of all TFNP problems that can be solved by gradient descent.

1.2 Outline of the Thesis and Main Contributions

Chapter 2 presents notation and definitions that are used throughout the thesis, including the formal definitions of the TFNP subclasses that we study. The rest of the thesis is then divided into three parts, where each part focuses on different subclasses of TFNP. Finally, Chapter 12 presents a short list of some of the main open problems that emerge from this thesis.

Part I: PPAD

In Chapter 3 we study the canonical PPAD-complete problem END-OF-LINE: given a source in a (succinctly represented) directed graph with indegree and outdegree at most 1, find another end of a line, i.e., a sink or another source. We show that variants of

[1]Here "natural" refers to the fact that these problems do not contain a computational device in the input, such as a circuit or a Turing machine.

this problem where we are given multiple sources – instead of just one – remain PPAD-complete. Surprisingly, the proof is not at all trivial. Our results for multiple-source End-of-Line provide new tools for proving *membership* in PPAD and as a direct application we show that PPAD is also characterised by the problem Imbalance: given an unbalanced vertex in a (succinctly represented) directed graph, find another unbalanced vertex. This corrects an incomplete proof given by Beame et al. [BCE+98].

In Chapter 4 we study the complexity of the computational problem associated with the Hairy Ball theorem. The Hairy Ball theorem is a well-known topological theorem due to Poincaré [Poi85] and Brouwer [Bro11]. It states that there is no non-vanishing continuous tangent vector field on an even-dimensional k-sphere. Informally, it can be interpreted as saying that "you can't comb a hairy ball flat without creating a cowlick," or "there is a point on the surface of the earth with zero horizontal wind velocity." We prove that the corresponding computational problem – namely finding a point where the vector field (almost) vanishes – is PPAD-complete, where the membership heavily relies on the tools developed in Chapter 3. Our result also shows that the Hairy Ball theorem is in a certain sense equivalent to Brouwer's fixed point theorem, which was not previously known.

In Chapter 5 we study the *Generalised Circuit* problem which is the most widely used tool to prove PPAD-hardness. It was in particular used for the seminal PPAD-completeness results for Nash [DGP09; CDT09; Rub18]. We show that even a significantly simplified version of Generalised Circuit remains PPAD-complete. This provides perhaps the simplest possible tool for proving PPAD-*hardness*.

Finally, in Chapter 6 we study the First Price Auction, namely the most basic auction format where the highest bidder wins the item and pays their bid. Athey [Ath01] has shown that an equilibrium is guaranteed to exist in this setting. We prove that computing an approximate equilibrium when the bidders have subjective priors about each other is PPAD-complete. The hardness result is obtained by a reduction from the simplified Generalised Circuit problem introduced in Chapter 5. The membership in PPAD is proved by providing a new proof of existence of equilibria based on Brouwer's fixed point theorem, instead of Kakutani's fixed point theorem, which was used by Athey [Ath01]. We also show that the problem of computing an *exact* equilibrium is complete for the class FIXP, introduced by Etessami and Yannakakis [EY10] to capture the complexity of computing an exact Brouwer fixed point.

Part II: CLS = PPAD ∩ PLS

In this part of the thesis we study a TFNP problem corresponding to Gradient Descent: given a smooth function on a bounded domain, find any point where Gradient Descent terminates. This problem has the following nice interpretation:

if some problem reduces to the Gradient Descent problem, then it can be solved by performing Gradient Descent (though not necessarily in a polynomial number of iterations). Since Gradient Descent is a special case of continuous local search, this Gradient Descent problem lies in the class CLS.

In Chapters 7 and 8 we prove that the Gradient Descent problem is not only CLS-complete, but in fact even PPAD ∩ PLS-complete. In particular, this implies the surprising collapse CLS = PPAD ∩ PLS. As a result, the class PPAD ∩ PLS, which is obtained by combining PPAD and PLS in a completely artificial way, turns out to have a very natural characterisation:

PPAD ∩ PLS *is the class of all problems that can be solved by performing Gradient Descent on a bounded domain.*

Most importantly, we establish that PPAD ∩ PLS is an important class that captures the complexity of interesting problems. The various consequences of our result are presented in detail in Chapter 7.

The main technical contribution underpinning this result is proved in Chapter 8. We show that computing an approximate Karush-Kuhn-Tucker (KKT) point of a continuously differentiable function over the domain $[0, 1]^2$ is PPAD ∩ PLS-hard. Since this problem can be seen to be equivalent to the Gradient Descent problem, it implies the collapse mentioned above. This KKT problem is the first non-artificial problem to be proved PPAD ∩ PLS-complete. Previously, the only known PPAD ∩ PLS-complete problems were of the form: given an instance I_A of a PPAD-complete problem A and an instance I_B of a PLS-complete problem B, output a solution to I_A or a solution to I_B. Our new technique shows how to reduce such problems to the problem of finding a KKT point.

Part III: PPA-k

In Chapter 9 we study the classes PPA-k, $k \geq 2$, defined by Papadimitriou [Pap94] to capture the complexity of arguments modulo k. In particular, PPA-2 = PPA corresponds to parity arguments such as the Handshaking lemma. Very little is known about these classes for $k \geq 3$, but they have recently emerged as candidates to capture the complexity of some fair division problems. In this chapter, we present structural results which provide a solid foundation for the further study of these classes. Namely, we investigate the classes PPA-k in terms of (i) equivalent definitions, (ii) inner structure, (iii) relationship to each other and to other TFNP classes, and (iv) closure under Turing reductions. In particular, we prove that, when $k = p^r$ is a prime power, PPA-k = PPA-p and PPA-p is closed under Turing reductions.

In Chapter 10 we present the first topological characterisations for the classes PPA-p for prime $p \geq 3$. First, we show that a simple two-dimensional computational problem

associated with a generalisation of Tucker's Lemma, termed p-POLYGON-TUCKER, is PPA-p-complete. Then, we show that a computational version of the Bárány-Shlosman-Szücs (BSS) theorem [BSS81] is also PPA-p-complete. The BSS theorem has various applications in combinatorics, including the proof of the Necklace Splitting theorem by Alon [Alo87], introduced below. All of our membership results are obtained through a new combinatorial proof for $\mathbb{Z}_p$-versions of Tucker's lemma that is a generalisation of the classic combinatorial proof of Tucker's lemma by Freund and Todd [FT81]. Topological characterisations have been pivotal in obtaining completeness results for PPAD and PPA, such as the PPAD-completeness of NASH [DGP09; CDT09] and the PPA-completeness of the Necklace Splitting problem with two thieves [FG19].

Finally, in Chapter 11 we study the Necklace Splitting problem. This is a classical problem in combinatorics and fair division: k thieves are aiming to split an open necklace containing n beads of t different colours (with exactly $k \cdot a_i$ beads of colour i, for some $a_i \in \mathbb{N}$), such that each thief receives exactly a_i beads of colour i. Alon [Alo87] has shown that $(k-1)t$ cuts are always enough to obtain such a division. The corresponding computational problem was studied by Filos-Ratsikas and Goldberg [FG19] who proved that it is PPA-complete for *two thieves*. In this chapter, we prove that the problem with k thieves lies in PPA-k under Turing reductions. In particular, when $k = p^r$ is a prime power, the problem lies in PPA-p (under standard many-one reductions). The result is obtained by reducing a continuous version of the problem (CONSENSUS-1/p-DIVISION) to a new generalisation of Tucker's lemma that we call $\mathbb{Z}_p$-STAR-TUCKER and which we prove to be PPA-p-complete in Chapter 10. This also provides a new proof of the Necklace Splitting theorem, which is fully combinatorial, unlike the original proof by Alon [Alo87].

1.3 List of Papers

The results presented in this thesis have appeared in the following papers.

Chapters 3 and 4 are based on:

- Paul W. Goldberg, Alexandros Hollender
  The Hairy Ball Problem is PPAD-complete [GH21]
  Journal of Computer and System Sciences (JCSS)
  Preliminary version at ICALP 2019

Chapters 5 and 6 are based on:

- Aris Filos-Ratsikas, Yiannis Giannakopoulos, Alexandros Hollender, Philip Lazos, Diogo Poças
  On the Complexity of Equilibrium Computation in First-Price Auctions [FGH+21]
  22nd ACM Conference on Economics and Computation (EC 2021)

Chapters 7 and 8 are based on:

- John Fearnley, Paul W. Goldberg, Alexandros Hollender, Rahul Savani
  The Complexity of Gradient Descent: CLS = PPAD ∩ PLS [FGHS21]
  53rd ACM Symposium on Theory of Computing (STOC 2021)
  (best paper award)

Chapter 9 is based on:

- Alexandros Hollender
  The Classes PPA-k: Existence from Arguments Modulo k [Hol21]
  Theoretical Computer Science (TCS)
  Preliminary version at WINE 2019

Finally, Chapters 10 and 11 are based on:

- Aris Filos-Ratsikas, Alexandros Hollender, Katerina Sotiraki, Manolis Zampetakis
  A Topological Characterization of Modulo-p Arguments and Implications for Necklace Splitting [FHSZ21]
  32nd ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)

The following papers were also co-authored during the author's doctoral studies, but were not included in this thesis:

- Aris Filos-Ratsikas, Kristoffer Arnsfelt Hansen, Kasper Høgh, Alexandros Hollender
  FIXP-membership via Convex Optimization: Games, Cakes, and Markets
  62nd IEEE Symposium on Foundations of Computer Science (FOCS 2021)

- Argyrios Deligkas, Aris Filos-Ratsikas, Alexandros Hollender
  Two's Company, Three's a Crowd: Consensus-Halving for a Constant Number of Agents [DFH21]
  22nd ACM Conference on Economics and Computation (EC 2021)

- Aris Filos-Ratsikas, Alexandros Hollender, Katerina Sotiraki, Manolis Zampetakis
  Consensus-Halving: Does It Ever Get Easier? [FHSZ20]
  21st ACM Conference on Economics and Computation (EC 2020)

- Paul W. Goldberg, Alexandros Hollender, Ayumi Igarashi, Pasin Manurangsi, Warut Suksompong
  Consensus Halving for Sets of Items [GHI+20]
  16th International Conference on Web and Internet Economics (WINE 2020)

- Paul W. Goldberg, Alexandros Hollender, Warut Suksompong
  Contiguous Cake Cutting: Hardness Results and Approximation Algorithms [GHS20]
  Journal of Artificial Intelligence Research (JAIR)
  Preliminary version at AAAI 2020

- Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, Alexandros Hollender, Alexandros A. Voudouris
  Maximum Nash Welfare and Other Stories About EFX [ABF+21]
  Theoretical Computer Science (TCS)
  Preliminary version at IJCAI 2020

- Georgios Birmpas, Jiarui Gan, Alexandros Hollender, Francisco J. Marmolejo-Cossío, Ninad Rajgopal, Alexandros A. Voudouris
  Optimally Deceiving a Learning Leader in Stackelberg Games [BGH+21]
  Journal of Artificial Intelligence Research (JAIR)
  Preliminary version at NeurIPS 2020

Chapter 2

Preliminaries

We work in the standard Turing machine model. For an introduction see, e.g., the textbooks [Sip06; AB09]. Let $\{0,1\}^*$ denote the set of all finite-length bit-strings, and for any $w \in \{0,1\}^*$, let $|w|$ denote its length. For $k \in \mathbb{N}$, let $[k] := \{1, 2, \ldots, k\}$.

2.1 The Class TFNP

Decision. A *decision problem* is given by a subset $\mathcal{L} \subseteq \{0,1\}^*$, interpreted as the following computational problem: given an instance $I \in \{0,1\}^*$, output "YES" if $I \in \mathcal{L}$, and "NO" if $I \notin \mathcal{L}$. In other words, the task is to *decide* whether $I \in \mathcal{L}$ or not. A decision problem $\mathcal{L}$ is also called a *language*. The class P contains all decision problems $\mathcal{L}$ that are polynomial-time decidable, i.e., such that there exists a polynomial-time Turing machine M that correctly solves $\mathcal{L}$, namely $I \in \mathcal{L} \Leftrightarrow M(I)$ accepts. The class NP contains all decision problems $\mathcal{L}$ that are polynomial-time verifiable, i.e., such that there exist a polynomial-time Turing machine V (*a Verifier*) and a polynomial p that satisfy $I \in \mathcal{L} \Leftrightarrow \exists c \in \{0,1\}^{p(|I|)} : V(I,c)$ accepts. This definition of NP is equivalent to its original definition as the set of all decision problems that can be solved by a polynomial-time *nondeterministic* Turing machine.

Search. A *search problem* is given by a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, interpreted as the following computational problem: given an instance $I \in \{0,1\}^*$, output $s \in \{0,1\}^*$ with $(I,s) \in R$ (i.e., find a solution), or output "NO" if no such s exists. The class FNP contains all NP search problems, namely all search problems R such that the relation R is polynomial-time decidable (i.e., $(I,s) \in R$ can be decided in polynomial time in $|I| + |s|$) and polynomially balanced (i.e., there exists some polynomial p such that $(I,s) \in R \implies |s| \leq p(|I|)$). The class FP contains all NP search problems R that can be solved in polynomial time, i.e., there exists a polynomial-time Turing machine M such that $M(I) \in \{0,1\}^* \cup \{\text{"NO"}\}$ is a correct output for R for any instance I.

Search vs Decision. The question of whether $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ is equivalent to $\mathsf{FP} \stackrel{?}{=} \mathsf{FNP}$. To see this, note that the decision and search versions of the satisfiability problem, namely SAT and FSAT, are NP- and FNP-complete, respectively. Furthermore, by the self reducibility of SAT, it follows that SAT and FSAT are polynomially (Turing) equivalent, in the sense that if one problem can be solved in polynomial time, then so can the other. Thus, it immediately follows that $\mathsf{P} = \mathsf{NP} \Leftrightarrow \mathsf{FP} = \mathsf{FNP}$.

More generally, every NP search problem R induces a corresponding NP decision problem

$$\mathcal{L}_R = \{\, I \in \{0,1\}^* \mid \exists s \in \{0,1\}^* : (I, s) \in R \,\}.$$

Clearly, every NP decision problem is induced by some NP search problem, but there might be *multiple*, very different, NP search problems that induce the *same* NP decision problem. A natural question is then: *How does the complexity of a decision problem relate to that of its corresponding search problems?* It is easy to see that the decision problem is always at least as easy as any of its corresponding search problems; in other words, decision reduces to search. For NP-complete decision problems, it turns out that the opposite also holds: search reduces to decision. This essentially follows from the fact that search reduces to decision for satisfiability, as mentioned above. Does search reduce to decision for all NP problems? Under a complexity theoretic assumption ($\mathsf{EE} \neq \mathsf{NEE}$), Bellare and Goldwasser [BG94] show that this is not the case in a strong sense: there exists an NP decision problem such that *none* of its corresponding search problems reduce to it.

Total NP search problems. The class TFNP contains all *total* NP search problems. A search problem is total, if every instance always admits at least one solution, i.e., "NO" is never a correct output. Formally, TFNP consists of all problems R in FNP that satisfy: for every $I \in \{0,1\}^*$, there exists $s \in \{0,1\}^*$ such that $(I, s) \in R$. One can also think of TFNP as being the set of all NP search problems that induce the (trivial) NP decision problem $\mathcal{L} = \{0,1\}^*$, as defined above. The set of search problems corresponding to this trivial decision problem turns out to be extremely rich and contains many natural problems.

The study of TFNP is also related to the question $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$, since, in a certain sense, TFNP lies between P and NP, or rather between FP and FNP. Clearly, $\mathsf{TFNP} \subseteq \mathsf{FNP}$, but with a slight abuse of notation we can also say that $\mathsf{FP} \subseteq \mathsf{TFNP}$. Indeed, any problem R in FP can be turned into a TFNP problem by including a pair (I, NO) in R for each instance I that does not have a solution, where NO is some dedicated bit-string. Note that despite this minor modification, the search problem remains essentially the same.

One of the reasons that make TFNP so interesting is that it is generally believed to lie *strictly* between FP and FNP. Indeed, on the one hand, TFNP contains many problems that we do not expect to be polynomial-time solvable. On the other hand, by a simple argument [MP91], it can be shown that if some TFNP problem is FNP-hard, then NP = co-NP, which is not expected to hold.

Reductions between TFNP problems. Let R and S be two TFNP problems. We say that R *(many-one) reduces* to S if there exist polynomial-time computable functions $f : \{0,1\}^* \to \{0,1\}^*$ and $g : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that, for all $I, s \in \{0,1\}^*$,

$$(f(I), s) \in S \implies (I, g(I,s)) \in R.$$

Intuitively, this says that for any instance I of R, if we can find a solution s to instance $f(I)$ of S, then $g(I,s)$ gives us a solution to instance I of R.

We say that R *Turing-reduces* to S if there exists a polynomial-time oracle Turing machine that solves R given access to an oracle for S. Note that in the context of TFNP a Turing reduction that only performs a single query is equivalent to a many-one reduction.

Promise problems. When defining a computational problem it is often convenient and, in fact, quite natural to restrict the instance space to some subset $\mathcal{X} \subset \{0,1\}^*$. In that case, an algorithm solving the problem is only required to work correctly on instances $I \in \mathcal{X}$ and can behave in an arbitrary way otherwise. In a certain sense, the algorithm is *promised* that the instance lies in $\mathcal{X}$. Accordingly, such problems are usually called *promise problems*. Clearly, promise problems are a strict generalisation of the standard computational problems where $\mathcal{X} = \{0,1\}^*$.

Even though TFNP does not, strictly speaking, contain promise problems with $\mathcal{X} \neq \{0,1\}^*$, it can still capture various settings where the instance space is restricted. Indeed, consider the case where we are interested in studying some FNP problem R on a restricted set of instances $\mathcal{X} \subset \{0,1\}^*$ and R is total on $\mathcal{X}$. First of all, if $\mathcal{X}$ is polynomial-time decidable, i.e., $\mathcal{X} \in \mathsf{P}$, then we can easily transform R into an equivalent TFNP problem by adding $(I, *)$ to R for all $I \notin \mathcal{X}$, where $*$ is some dedicated bit-string. Furthermore, even when we do not know whether $\mathcal{X} \in \mathsf{P}$, the restriction on the instance space can often still be enforced *syntactically*. Namely, there exists some polynomial-time computable function F such that for all instances $I \in \{0,1\}^*$ it holds that $F(I) \in \mathcal{X}$, and, additionally, whenever $I \in \mathcal{X}$ it holds that $(F(I), s) \in R \implies (I, s) \in R$. These techniques allow us to define a TFNP problem that captures the complexity of R on inputs in $\mathcal{X}$, without having to use promise problems.

In some cases, however, it is not clear whether one of the two "tricks" above can be used. In such cases it is still possible to obtain a related TFNP problem by modifying the original problem so as to allow additional types of solutions. These types of solutions are usually called *violation solutions*, as they attest that some promise on the instance has been violated. For example, we will be interested in studying some problems where the instance contains an L-Lipschitz-continuous function represented as an arithmetic circuit. Since there is no known way of syntactically enforcing that an arithmetic circuit be L-Lipschitz-continuous, such problems are turned into TFNP problems by introducing the following violation solutions: any pair of points (x, y) witnessing a violation of L-Lipschitz-continuity.

Of course, when adding violation solutions to some promise problem $\tilde{S}$ in order to obtain a TFNP problem S, there is no guarantee that S is not *strictly* harder to solve than $\tilde{S}$. In particular, providing a reduction from some TFNP problem R to S would not necessarily imply anything about $\tilde{S}$. In order to address this issue, a stronger notion of reduction, called *promise-preserving* reduction, has been proposed by Fearnley et al. [FGMS20]. A reduction (f, g) from problem R to problem S is promise-preserving, if for any instance I of R, for any violation solution s of instance $f(I)$ of S, it holds that $g(I, s)$ is a violation solution of instance I of R. Informally: any violation solution of S is mapped back to a violation solution of R. As a result, such a reduction also yields a reduction from $\tilde{R}$ to $\tilde{S}$, namely the original promise versions of the problems. In particular, if $R = \tilde{R}$, i.e., no violation solutions were added to R, then a promise-preserving reduction from R to S immediately yields a reduction from R to $\tilde{S}$, i.e., even the promise version of S is as hard to solve as R.

Alternatively, one could consider extending the definition of TFNP to include promise problems. However, most of these promise TFNP problems would no longer be syntactically total, thus missing out on the property which makes TFNP special. Indeed, it is easy to define a promise TFNP problem that is FNP-hard (given a satisfiable 3-SAT formula, find a satisfying assignment), whereas this would immediately imply NP = co-NP for a standard TFNP problem. For this reason, promise TFNP problems are generally avoided, even though there does not seem to be a reason to avoid them altogether, as long as one is aware of the issue mentioned above.

2.2 Representation of Functions

Many of the problems studied in this thesis are of the following form: given some function $f : U_1 \to U_2$, find a point $x \in U$ such that $(x, f(x))$ satisfies some property. In order to formally define such a computational problem we must specify how exactly f is represented. We distinguish between two cases:

- when U_1 and U_2 are sets of bounded-length bit-strings, f is represented using a Boolean circuit,

- when U_1 and U_2 are subsets of $\mathbb{R}^d$, f is represented using an arithmetic circuit.

In both cases, polynomial-time Turing machines could be used instead of circuits. However, using Turing machines would be more cumbersome, because there is no clear connection between the length of the description of a Turing machine and its running time. As a result, one would have to fix some polynomial upper bound on the running time and introduce violation solutions for the case where the Turing machine does not terminate within the bound on some input. In contrast, circuits have the advantage that they can be evaluated in time polynomial in their description length. Thus, using circuits to represent functions ensures that there is a direct connection between the length of the representation of the function and the time it takes to evaluate it. Another option is to use the notion of polynomially computable and continuous function classes, which we briefly present in Section 2.2.3.

2.2.1 Boolean circuits

A Boolean circuit $C : \{0,1\}^n \to \{0,1\}^m$, with n inputs and m outputs, is allowed to use the logic gates $\wedge$ (AND), $\vee$ (OR) and $\neg$ (NOT), where the $\wedge$ and $\vee$ gates have fan-in 2, and the $\neg$ gate has fan-in 1. We let $\mathrm{size}(C)$ denote the size of circuit C, i.e., the number of bits needed to represent it. In some cases, we will identify $\{0,1\}^n$ with $[2^n]$ or with $\{0,1,\ldots,2^n-1\}$.

Encoding of Sets. Many of the computational problems we consider in this thesis involve Boolean circuits whose output is interpreted as a set. For example, it will often be the case that a circuit C takes as input a bit-string in $\{0,1\}^n$ and outputs a set of at most m bit-strings in $\{0,1\}^n$. We will denote this by $C : \{0,1\}^n \to \mathsf{Set}_{\leq m}(\{0,1\}^n)$. Of course, a Boolean circuit has a fixed number of output bits and so the circuit will in fact be of the form $C : \{0,1\}^n \to \{0,1\}^t$, for some t that is sufficiently large so that there are enough bits to encode any set of size at most m. It is easy to see that taking $t = mn + 1$ is enough. Indeed, we can for example use the following encoding: the set $\{x_1,\ldots,x_\ell\} \subseteq \{0,1\}^n$, $\ell \leq m$, is represented by the bit-string $x_1 \cdots x_\ell 10^{(m-\ell)n} \in \{0,1\}^{mn+1}$. Clearly, we can efficiently check whether a bit-string in $\{0,1\}^{mn+1}$ is a valid representation of a set, and if not, we can just interpret it as the empty set $\emptyset \subset \{0,1\}^n$.

2.2.2 Arithmetic circuits

Let n be a positive integer. For $x \in \mathbb{R}^n$, $\|x\|_2$, $\|x\|_1$ and $\|x\|_\infty$ denote the standard ℓ_2-norm (Euclidean), ℓ_1-norm (Manhattan) and ℓ_∞-norm (Maximum) respectively. Unless stated otherwise, $\|\cdot\|$ refers to the Euclidean norm. For $x, y \in \mathbb{R}^n$, $\langle x, y\rangle := \sum_{i=1}^n x_i y_i$ denotes the inner product.

We let $B^n = \{x \in \mathbb{R}^n : \|x\|_2 \leq 1\}$ denote the n-dimensional unit ball and $S^{n-1} = \partial B^n = \{x \in \mathbb{R}^n : \|x\|_2 = 1\}$ the corresponding unit sphere. For any non-empty closed convex set $D \subseteq \mathbb{R}^n$, we let $\Pi_D : \mathbb{R}^n \to D$ denote the projection onto D with respect to the Euclidean norm. Formally, for any $x \in \mathbb{R}^n$, $\Pi_D(x)$ is the unique point $y \in D$ that minimises $\|x - y\|$.

Rational numbers are represented as irreducible fractions, with the numerator and denominator of the irreducible fraction given in binary. Note that given any fraction, it can be made irreducible in polynomial time using the Euclidean algorithm. For a rational number x, we let size(x) denote the number of bits needed to represent x, i.e., the number of bits needed to write down the numerator and denominator (in binary) of the irreducible fraction for x. We also use this notation for vectors and matrices with rational entries.

Arithmetic circuits. An arithmetic circuit representing a function $f : \mathbb{R}^n \to \mathbb{R}^m$, is a circuit with n inputs and m outputs, which uses gates with fan-in 2 that perform operations such as addition or multiplication, and gates with fan-in 0 that represent rational constants. In this thesis, unless stated otherwise, an arithmetic circuit is allowed to use the operations $+, -, \times, \max, \min, >$. The comparison gate $>$, on input $a, b \in \mathbb{R}$, outputs 1 if $a > b$, and 0 otherwise. For an arithmetic circuit f, we let size(f) denote the size of the circuit, i.e., the number of bits needed to describe the circuit, including the rational constants used therein.

Well-behaved arithmetic circuits. The definition of arithmetic circuits that we have introduced above is very natural. However, these circuits suffer from a subtle issue that seems to have been overlooked in some prior works. Using the multiplication gate, such an arithmetic circuit can perform repeated squaring to construct numbers that have exponential representation size with respect to the size of the circuit and the input to the circuit. In other words, the circuit can construct numbers that are *double* exponential (or the inverse thereof). Thus, in some cases, it might not be possible to evaluate the circuit on some input efficiently, i.e., in time polynomial in the size of the circuit and the given input.

In this thesis, we resolve this issue as follows. We restrict our attention to what we call *well-behaved* arithmetic circuits.[1] An arithmetic circuit f is well-behaved if, on any directed path that leads to an output, there are at most $\log(\text{size}(f))$ *true* multiplication gates. A true multiplication gate is one where both inputs are non-constant nodes of the circuit. In particular, note that we allow our circuits to perform multiplication by a constant as often as needed without any restriction. Indeed, these operations cannot be used to do repeated squaring.

It is easy to see that given an arithmetic circuit f, we can check in polynomial time whether f is well-behaved. Thus, the restriction to well-behaved circuits can be enforced syntactically, without the need for violation solutions. Furthermore, these circuits can always be efficiently evaluated.

Lemma 2.1. *Let f be a well-behaved arithmetic circuit with n inputs. Then, for any rational $x \in \mathbb{R}^n$, $f(x)$ can be computed in time* $\text{poly}(\text{size}(f), \text{size}(x))$.

We provide a proof of this Lemma in Appendix A.

Remark 2.1. Our definition of well-behaved circuits is robust in the following sense. For any $k \in \mathbb{N}$, say that a circuit f is k-well-behaved if, on any path that leads to an output, there are at most $k \cdot \log(\text{size}(f))$ true multiplication gates. In particular, a circuit is well-behaved if it is 1-well-behaved. It is easy to see that for any fixed $k \in \mathbb{N}$, if we are given a circuit f that is k-well-behaved, we can construct in time $\text{poly}(\text{size}(f))$ a circuit f' that is well-behaved and computes the same function as f. This can be achieved by adding $(\text{size}(f))^k$ dummy gates to the circuit f, i.e., gates that do not alter the output of the circuit. For example, we can add gates that repeatedly add 0 to the output of the circuit.

Lipschitz-continuity. Note that even well-behaved arithmetic circuits might not yield continuous functions, because of the comparison gate. Some of our problems require continuity of the function, and a very convenient type of continuity for computational purposes is Lipschitz-continuity. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is Lipschitz-continuous (w.r.t. the norm $\|\cdot\|$) on the domain $D \subseteq \mathbb{R}^n$ with Lipschitz-constant L, if for all $x, y \in D$

$$\|f(x) - f(y)\| \leq L \cdot \|x - y\|.$$

There is no known way of syntactically enforcing that an arithmetic circuit be Lipschitz-continuous. Thus, to ensure that the problems in question indeed lie in TFNP, we allow

[1] We use well-behaved arithmetic circuits in this thesis because (a) they can always be efficiently evaluated, and (b) they are powerful enough to express the functions that we construct in our reductions. Our results continue to hold if one considers other restricted classes of circuits, as long as they satisfy (a) and (b). In particular, this holds for any class of arithmetic circuits that is at least as expressive as well-behaved circuits, while remaining efficiently evaluatable.

any well-behaved circuit in the input, together with a purported Lipschitz-constant L, and also accept a pair (x, y) witnessing a violation of L-Lipschitz-continuity as a solution. This is the usual "trick" that was also used by Daskalakis and Papadimitriou [DP11] for the definition of CLS.

Linear arithmetic circuits. Linear arithmetic circuits are only allowed to use the operations $+, -, \times\zeta, \max, \min$ and rational constants. The operation $\times\zeta$ denotes multiplication by a constant (which is part of the description of the circuit). In particular, such circuits cannot use general multiplication gates or comparison gates. Note that even though these circuits are called *linear*, they in fact correspond to *piecewise linear* functions.

Every linear arithmetic circuit is a well-behaved arithmetic circuit, and so, by Lemma 2.1, can be evaluated in polynomial time. Furthermore, every linear arithmetic circuit represents a Lipschitz-continuous function such that the Lipschitz constant has polynomial bit-size with respect to the size of the circuit.

Lemma 2.2. *Any linear arithmetic circuit $f : \mathbb{R}^n \to \mathbb{R}^m$ is $2^{\text{size}(f)^2}$-Lipschitz-continuous (w.r.t. the ℓ_∞-norm) over $\mathbb{R}^n$.*

We provide a proof of this Lemma in Appendix A.

As a result, no violation solutions are needed to ensure the Lipschitzness of such circuits. Depending on the situation – namely if restricting the functions to be piecewise linear is not an issue – we will also use these circuits to define TFNP problems.

Interestingly, it turns out that Lipschitz-continuous well-behaved arithmetic circuits can be arbitrarily well approximated by linear circuits.

Theorem 2.1. *Given a well-behaved arithmetic circuit $f : [0,1]^n \to \mathbb{R}^d$, a purported Lipschitz constant $L > 0$, and a precision parameter $\varepsilon > 0$, in polynomial time in $\text{size}(f)$, $\log L$ and $\log(1/\varepsilon)$, we can construct a linear arithmetic circuit $F : [0,1]^n \to \mathbb{R}^d$ such that for any $x \in [0,1]^n$ it holds that:*

- $\|f(x) - F(x)\|_\infty \leq \varepsilon$, *or*
- *given x, we can efficiently compute $y \in [0,1]^n$ such that*

$$\|f(x) - f(y)\|_\infty > L\|x - y\|_\infty.$$

Here "efficiently" means in polynomial time in $\text{size}(x)$, $\text{size}(f)$, $\log L$ and $\log(1/\varepsilon)$.

The proof of this result is included in Appendix B, where we show an even more general result, namely that any *polynomially-approximately-computable* class of functions (defined in Appendix B) can be approximated by linear arithmetic circuits.

2.2.3 Polynomially Computable and Polynomially Continuous Function Classes

In this section, we briefly present the notion of a polynomially computable and polynomially continuous class of functions, which we will use for some of our membership results.

Consider a class of functions $\mathcal{F}$ with some representation scheme. Formally, any bit-string $w \in \{0,1\}^*$ represents some function $f_w \in \mathcal{F}$, $f_w : \mathbb{R}^n \to \mathbb{R}^m$, and for any function $f \in \mathcal{F}$ there exists $w \in \{0,1\}^*$ with $f_w = f$. With a slight abuse of notation, we let $\text{size}(f)$ denote the size of the representation of f, i.e., the length of the bit-string w representing it. For any $f \in \mathcal{F}$, $f : \mathbb{R}^n \to \mathbb{R}^m$, we assume that $\text{size}(f) \geq n + m$ (except perhaps for a finite number of $f \in \mathcal{F}$). An example is the class of functions represented by arithmetic circuits, where we let invalid representations map to some arbitrary fixed function. In this case, $\text{size}(f)$ is simply the size of the circuit, as defined earlier.

Definition 2.1 (Etessami and Yannakakis [EY10])**.** Let $\mathcal{F}$ be a class of functions.

- $\mathcal{F}$ is *polynomially computable*, if there exists some polynomial p such that for any $f \in \mathcal{F}$, $f : \mathbb{R}^n \to \mathbb{R}^m$, and any rational input $x \in \mathbb{R}^n$, $f(x)$ can be computed in time $p(\text{size}(f) + \text{size}(x))$.

- $\mathcal{F}$ is *polynomially continuous*, if there exists some polynomial q such that for any $f \in \mathcal{F}$, $f : \mathbb{R}^n \to \mathbb{R}^m$, and any rational $\varepsilon > 0$, there exists a rational $\delta > 0$ with $\text{size}(\delta) \leq q(\text{size}(f) + \text{size}(\varepsilon))$ such that for all $x, y \in \mathbb{R}^n$ we have $\|x - y\|_\infty \leq \delta \implies \|f(x) - f(y)\|_\infty \leq \varepsilon$.

By Lemmas 2.1 and 2.2 it follows that the class of functions represented by linear arithmetic circuits is polynomially computable and polynomially continuous. Note, however, that polynomial continuity fails if we consider well-behaved arithmetic circuits instead, because the comparison gate might make the function discontinuous.

2.3 Syntactic Subclasses of TFNP

As noted by Megiddo and Papadimitriou [MP91] and Papadimitriou [Pap94], the definition of TFNP is semantic and as such TFNP is unlikely to admit complete problems. Furthermore, Pudlák [Pud15] has provided an oracle with respect to which TFNP indeed does not have any complete problems.

As a result, various *syntactic* subclasses of TFNP have been defined to capture the complexity of natural problems. These classes are usually defined using very simple existence principles. The main subclasses and corresponding existence principles are:[2]

- PPAD : given a directed graph and an unbalanced vertex (i.e., outdegree $\neq$ indegree), there must exist another unbalanced vertex. [Pap94]
- PPADS : given a directed graph and a positively unbalanced vertex (i.e., outdegree $>$ indegree), there must exist a negatively unbalanced vertex (i.e., outdegree $<$ indegree). [Pap94; BCE+98]
- PPA : given an undirected graph and vertex with odd degree, there must exist another vertex with odd degree (Handshaking Lemma). [Pap94]
- PPA-p, prime $p \geq 2$: existence is guaranteed by an argument modulo p. PPA-2 = PPA. [Pap94]
- PPP : given a function mapping a finite set to a smaller set, there must exist a collision (Pigeonhole Principle). [Pap94]
- PLS : given a directed acyclic graph, there must exist a sink (where the acyclicity is enforced locally by a potential). [JPY88]
- CLS : existence is guaranteed by a *continuous* local search argument. [DP11]
- EOPL : given a directed acyclic graph and an unbalanced vertex, there must exist another unbalanced vertex (where the acyclicity is enforced locally by a potential). [FGMS20]
- UEOPL : same as EOPL, but the graph only consists of lines that never overlap in terms of the potential. [FGMS20]
- SOPL : given a directed acyclic graph and a positively unbalanced vertex, there must exist a negatively unbalanced vertex (where the acyclicity is enforced locally by a potential). [GKRS19]

The known relationships between these classes are shown in Figure 2.1. Goldberg and Papadimitriou [GP18] have also defined a class PTFNP and shown that it contains most of the known TFNP subclasses. PTFNP stands for *provable* TFNP and its complete problem is essentially: given a proof of a contradiction, find a mistake in the proof.[3] Another class that we have not mentioned is PWPP, a subclass of PPP that corresponds to a weaker version of the Pigeonhole Principle.

Before presenting the main classes in more detail, we briefly discuss some important related results.

[2]For some of these classes, the corresponding existence principles listed here are more general

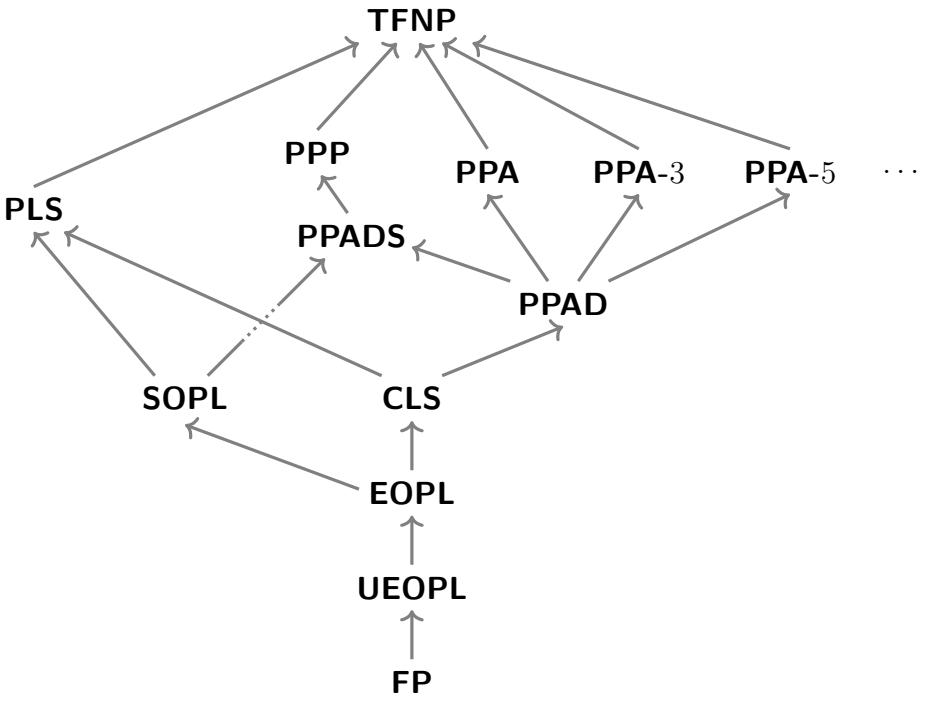


Figure 2.1: The main subclasses of TFNP, where arrows are used to denote containment.

Closure under Turing reductions. The subclasses of TFNP are formally defined using complete problems that embody the corresponding existence principles. For example, PPAD is formally defined as the set of all TFNP problems that (many-one) reduce to a problem called End-of-Line, which embodies the PPAD existence principle (see Section 2.3.1). As a result, these classes are trivially closed under many-one reductions. However, it is not immediately clear whether they are also closed under Turing reductions. Note that closure under Turing reductions can be useful for proving membership in a class, but even just on a purely structural level, it is a clear sign of robustness.

PPA, PPAD, PPADS and PLS have been shown to be closed under Turing reductions by Buss and Johnson [BJ12]. In Chapter 9 we show that this also holds for the classes PPA-p for prime p. The collapse CLS = PPAD ∩ PLS which we prove in Chapters 7 and 8 also implies that CLS is closed under Turing reductions. However, it is still not known whether PPP is closed under Turing reductions. For PWPP Jeřábek [Jeř16] has shown that it is closed under *nonadaptive* Turing reductions, i.e., where the oracle queries are not allowed to depend on responses to previous queries.

than the ones originally used for their definition, but have since been proved to be equivalent. For PPAD this is proved in this thesis (see Section 2.3.1). For PPADS it was proved by Beame et al. [BCE+98], and the same arguments also work for SOPL. For PPA this was already proved when the class was first defined [Pap94]. For EOPL this was proved by Ishizuka [Ish21].

[3] In order to formally instantiate this problem, one first has to fix a proof system. Goldberg and Papadimitriou [GP18] define PTFNP by picking a specific proof system in the definition of the problem. One can define different versions of this class by changing the proof system that is used.

Oracle separations. Unfortunately, we cannot hope to prove that classes in Figure 2.1 are distinct without first proving $\mathsf{P} \neq \mathsf{NP}$. Thus, for now, we have to rely on oracle separations as indications that these classes are distinct.

Indeed, there are extensive results on this and all of these results have a common general technique. First, one defines type-2 analogues of the TFNP subclasses, namely, the circuits in the input are replaced by black boxes. The next step is to prove that there is no reduction between complete problems for these classes. It is indeed possible to prove such separations *unconditionally* in this type-2 setting. Finally, this impossibility of a reduction yields a separation of the classes with respect to any generic oracle (see [BCE+98] for more details on this). Another equivalent – and perhaps cleaner – approach directly uses tools from proof complexity on TFNP problems formulated as unsatisfiable CNFs. In this way, it is possible to characterise TFNP subclasses in terms of proof systems; see [GKRS19] for more details.

Using this framework, Beame et al. [BCE+98] proved all possible separations between the classes PPA, PPAD, PPADS and PPP, even for Turing reductions. Subsequently, Morioka [Mor01] extended these results by proving that PPAD is not Turing reducible to PLS in the type-2 setting. This implies that none of PPA, PPAD, PPADS and PPP are contained in PLS under generic oracles. Buresh-Oppenheim and Morioka [BM04] further proved that there is no many-one reduction from PLS to PPA in the type-2 setting, and this was extended to Turing reductions by Buss and Johnson [BJ12]. Thus, an important open problem is whether PLS is contained in PPP or PPADS under generic oracles. To show that this is not the case, it would suffice to prove that there is no reduction from PLS to PPP in the type-2 setting.

Hardness from cryptographic assumptions. Hubácek, Naor and Yogev [HNY17] proved that average-case hardness of TFNP can be based on average-case hardness of NP, albeit under some strong derandomisation assumption. More recently, Pass and Venkitasubramaniam [PV20] showed that if NP is hard on average then either (a) one-way functions exist, or (b) TFNP is also hard on average. This result in particular implies that TFNP is hard on average in Pessiland, namely Impagliazzo's world [Imp95] where NP is hard on average but one-way functions do not exist.

It has been shown that subclasses of TFNP are hard if we assume some specific cryptographic assumptions. First of all, Papadimitriou [Pap94] already noted that if one-way permutations exist, then PPP is not polynomial-time solvable. Similarly, if collision-resistant hash functions exist, then PWPP (and thus PPP) is hard. Jeřábek [Jeř16] showed that the FACTORING problem lies in PPA and PWPP under *randomised* reductions. Thus, hardness of these classes can be based on the hardness of FACTORING. Komargodski, Naor and Yogev [KNY19] also show that RAMSEY, a problem based on Ramsey's theorem, is hard if collision-resistant hash functions exist.

For PPAD, Abbott, Kane and Valiant [AKV04] proved that hardness can be based on virtual black-box obfuscation, but this is an extremely strong cryptographic assumption that is unlikely to hold. Bitansky, Paneth and Rosen [BPR15] showed that (average-case) hardness of PPAD can be based on the existence of injective one-way functions and indistinguishability obfuscation. The latter assumption is quite strong, but it has recently been related to more standard assumptions [JLS21]. Their reduction uses a problem called Sink-of-Verifiable-Line as an intermediate step. Also using this problem, Garg, Pandey and Srinivasan [GPS16] proved that hardness of PPAD can be based on the existence of one-way permutations and a public key functional encryption assumption. This is also a non-standard cryptographic assumption. Hubácek and Yogev [HY20] showed that these hardness results also hold for CLS by showing that Sink-of-Verifiable-Line lies in CLS. In fact, their reduction essentially first shows that Sink-of-Verifiable-Line lies in EOPL and then that EOPL is a subset of CLS, so the hardness results also hold for EOPL. More recently, Choudhuri et al. [CHK+19] proved that hardness of EOPL can also be based on the soundness of the Fiat-Shamir heuristic applied to the sumcheck protocol and hardness of #SAT, the problem of counting satisfying assignments. Jawale et al. [JKKZ21] showed that these assumptions can be replaced by the assumption that Learning With Errors (LWE) is sub-exponentially hard.

The question of whether PPAD-hardness can be based on more standard cryptographic assumptions such as the existence of one-way functions is still open. Within the framework of black-box reductions, Rosen, Segev and Shahaf [RSS21] have shown that strong cryptographic assumptions are not essential for PPAD-hardness. Thus, there is hope that hardness can be based on simpler cryptographic assumptions. However, they have also proved that such black-box reductions cannot use the Sink-of-Verifiable-Line problem and would necessarily yield an End-of-Line instance with an exponential number of solutions.

2.3.1 The Classes PPAD and PPADS

PPAD. The class PPAD, which stands for "Polynomial Parity Argument on Directed graphs," was introduced by Papadimitriou [Pap94]. It is usually defined using a problem called End-of-Line. In this problem, the input is a succinct representation of an exponentially large directed graph, where every vertex has in- and out-degree at most 1. We are given a source and have to locate a sink or another source, which is guaranteed to exist by a simple argument.

Formally, PPAD is defined as the set of all TFNP problems that (many-one) reduce to End-of-Line [Pap94; DGP09].

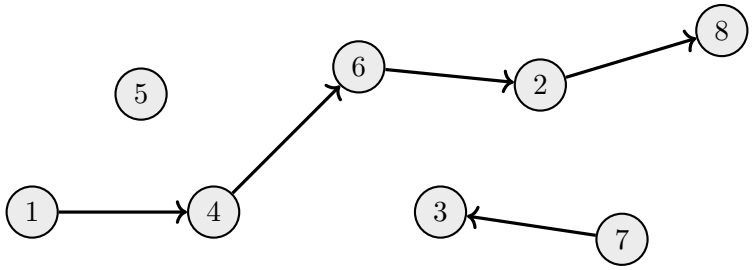


Figure 2.2: Example of an End-of-Line instance for $n = 3$, with vertex set $\{0,1\}^3$ interpreted as [8]. The vertices are represented by circular nodes and the directed edges by arrows. In this example, the End-of-Line solutions are the vertices 3, 7 and 8. In more detail, vertices 3 and 8 are sinks, while vertex 7 is a source. Note that the "trivial" source 1 is not a solution. Finally, the isolated vertex 5 is also not a solution. Although this example does not contain cycles, they are also allowed in End-of-Line and do not introduce solutions.

Definition 2.2. **End-of-Line**:

Input: Boolean circuits $S, P : \{0,1\}^n \to \{0,1\}^n$ with $P(0^n) = 0^n \neq S(0^n)$.
Output: $x \in \{0,1\}^n$ such that $P(S(x)) \neq x$ or $S(P(x)) \neq x \neq 0^n$.

Note that the graph is not provided explicitly in the input, but instead we are given Boolean circuits that efficiently compute the successor and predecessor of each vertex. In more detail, the circuits define a graph as follows. There is a directed edge from vertex x to y ($x \neq y$), if and only if $S(x) = y$ and $P(y) = x$. Note that any badly defined edge, i.e., $S(x) = y$ and $P(y) \neq x$, or $P(y) = x$ and $S(x) \neq y$, qualifies as a solution of End-of-Line as defined above (because $P(S(x)) \neq x$ or $S(P(x)) \neq x$ respectively). The vertex 0^n is a source of the graph, unless $P(S(0^n)) \neq 0^n$, in which case 0^n is a valid solution to the problem as stated above. The condition $P(0^n) = 0^n \neq S(0^n)$ can be enforced syntactically, so this is indeed a TFNP problem and not a promise problem. See Figure 2.2 for an example of an instance of End-of-Line.

The attentive reader might have noticed that End-of-Line is only a special case of the general principle "given a directed graph and an unbalanced vertex (i.e., outdegree $\neq$ indegree), there must exist another unbalanced vertex." In Chapter 3 we show that End-of-Line is equivalent to the problem Imbalance which embodies this more general principle. As a result, it is indeed correct to say that PPAD also corresponds to this general formulation of the principle.

In order to prove that a problem is PPAD-hard, one can reduce from End-of-Line or any other PPAD-complete problem. In fact, PPAD also has a useful topological characterisation in terms of Brouwer's fixed point theorem.

Theorem 2.2 (Brouwer's Fixed Point Theorem [Bro11])**.** *Let $D \subseteq \mathbb{R}^n$ be a nonempty, compact, and convex set. Then, for any continuous function $f : D \to D$ there exists $x \in D$ with $f(x) = x$.*

Namely, finding an approximate Brouwer fixed point is PPAD-complete, even in two dimensions [Pap94; CD09]. In Appendix C we present a general version of the computational problem that is useful for proving membership in PPAD. The discrete counterpart of Brouwer's fixed point theorem – Sperner's lemma [Spe28] – is also PPAD-complete.

Perhaps the most famous PPAD-complete problem is Nash [DGP09; CDT09]. The characterisation of PPAD using Brouwer's fixed point theorem – and more specifically the Generalised Circuit problem, which we study in more detail in Chapter 5 – was crucial for proving the PPAD-hardness of Nash. [Rub18] improved the hardness results for the Generalised Circuit problem and used this to show that computing a Nash equilbrium is PPAD-hard in polymatrix games even with *constant* approximation parameter. PPAD has generally been very successful in capturing the complexity of problems in Game Theory [Meh14; CDO15; DQS12], Economics [CSVY08; CDDT09; VY11; CPY17] and beyond [KPR+13].

PPADS. The class PPADS, which stands for "Polynomial Parity Argument on Directed graphs: Sink," is obtained by modifying End-of-Line so that now only sinks are solutions. Namely, we are not interested in finding other sources. Formally, PPADS is defined as the set of all TFNP problems that reduce to Sink [Pap90; BCE+98].

Definition 2.3. **Sink**:

Input: Boolean circuits $S, P : \{0,1\}^n \to \{0,1\}^n$, with $P(0^n) = 0^n \neq S(0^n)$.
Output: $x \in \{0,1\}^n$ such that $P(S(x)) \neq x$.

The interpretation of the circuits S and P is exactly the same as in End-of-Line. In Chapter 3 we study some variants of Sink and also explain why the problem indeed captures the full generality of the principle "given a directed graph and a positively unbalanced vertex, there must exist a negatively unbalanced vertex."

2.3.2 The Class PPA

The class PPA (Polynomial Parity Argument) is defined as the set of all TFNP problems that reduce to the problem Leaf [Pap94; BCE+98]: given an undirected graph with maximum degree 2 and a leaf (i.e., a vertex of degree 1), find another leaf. Formally, the problem is defined as follows, where we use the Set-notation introduced in Section 2.2.1.

Definition 2.4. **LEAF**:

Input: Boolean circuit $C : \{0,1\}^n \to \mathsf{Set}_{\leq 2}(\{0,1\}^n)$ with $|C(0^n)| = 1$.
Output: Either

- $x \neq 0^n$ such that $|C(x)| = 1$ (another leaf), or
- x, y such that $x \in C(y)$ but $y \notin C(x)$ (an inconsistent edge).

Here the vertex set is $\{0,1\}^n$ and the undirected graph is represented by a Boolean circuit $C : \{0,1\}^n \to \mathsf{Set}_{\leq 2}(\{0,1\}^n)$. By this we mean that for any $x \in \{0,1\}^n$, we interpret $C(x)$ as the set of potential neighbours of x, where we syntactically enforce that $x \notin C(x)$. We say that there is an edge between x and y if $x \in C(y)$ and $y \in C(x)$. Thus, every vertex has at most two neighbours. As before, note that the size of the graph can be exponential with respect to its description size.

It is easy to see that PPA contains PPAD, since ignoring the direction in an instance of END-OF-LINE yields an instance of LEAF. Papadimitriou [Pap94] proved that LEAF is equivalent to the seemingly more general problem ODD: given an undirected graph and an odd degree node, find another one. Beame et al. [BCE+98] proved that the following problem is equivalent to LEAF and thus also PPA-complete:

Definition 2.5. **LONELY**:

Input: Boolean circuit $C : \{0,1\}^n \to \{0,1\}^n$ with $C(0^n) = 0^n$.
Output: $x \neq 0^n$ such that $C(x) = x$ or $C(C(x)) \neq x$.

The first PPA-complete problems were all related to topological problems on non-orientable spaces [Gri01; FISV06; DEF+21]. Aisenberg, Bonet and Buss [ABB20] provided the first topological characterisation of PPA on standard orientable spaces by proving the PPA-completeness of the problems associated to the Borsuk-Ulam theorem and its discrete counterpart, Tucker's lemma. This corrected an incorrect claim made in [Pap94] about PPAD-membership of these problems. We include the formal statements of these theorems here, see the book by Matoušek [Mat08] for more details.

Theorem 2.3 (Borsuk-Ulam Theorem [Bor33])**.** *For every continuous odd function $f : S^n \to \mathbb{R}^n$ (i.e., such that $f(-x) = -f(x)$), there exists a point $x \in S^n$ with $f(x) = 0$.*

Theorem 2.4 (Tucker's Lemma [Tuc45])**.** *Let $m, d \geq 1$. Let $\lambda : ([-m, m] \cap \mathbb{N})^d \to \{\pm 1, \pm 2, \ldots, \pm d\}$ be any labelling that satisfies $\lambda(-x) = -\lambda(x)$ for all $x \in ([-m, m] \cap \mathbb{N})^d$ such that $\exists i$ with $|x_i| = m$. Then there exist two points $x, y \in ([-m, m] \cap \mathbb{N})^d$ with $\|x - y\|_\infty \leq 1$ that have opposite labels, i.e., $\lambda(x) = -\lambda(y)$.*

Subsequently, Deng, Feng and Kulkarni [DFK17] proved that a special version of Tucker's lemma, known as the Octahedral Tucker lemma, is also PPA-complete. The first PPA-complete problems not inspired from topological fixed point theorems were given by Belovs et al. [BIQ+17], who considered problems related to Chevalley's theorem and the Combinatorial Nullstellensatz. It is only relatively recently that Filos-Ratsikas and Goldberg [FG18; FG19] provided the first "natural" PPA-complete problems, in the sense that these problems do not have a computational device in their input (such as a circuit or Turing machine). They proved that the Consensus-Halving problem from fair division, the Necklace Splitting problem (with 2 thieves) from combinatorics, and the discrete Ham Sandwich problem from geometry are all PPA-complete.

In Part III of this thesis we study generalisations of PPA (which corresponds to arguments modulo 2) by considering the classes PPA-k corresponding to arguments modulo k. We provide various structural results, including the first topological characterisations for these classes. We also prove a membership result for the Necklace Splitting problem with k thieves.

2.3.3 The Class PLS

The class PLS (Polynomial Local Search) is defined as the set of all TFNP problems that reduce to the problem LOCALOPT [JPY88; DP11].

Definition 2.6. **LOCALOPT**:

Input: Boolean circuits $S, V : [2^n] \to [2^n]$.
Output: $v \in [2^n]$ such that $V(S(v)) \geq V(v)$.

This problem embodies local search over the node set $[2^n]$. The output of the circuit V represents a value and ideally we would like to find a node $v \in [2^n]$ that minimises $V(v)$. The circuit S helps us in this task by proposing a possibly improving node $S(v)$ for any v. We stop our search, when we find a v such that $V(S(v)) \geq V(v)$, i.e., S no longer helps us decrease the value of V. This is local search, because the circuit S represents the search for an improving node in some small (polynomial-size) neighbourhood.

In this thesis, we also make use of the following PLS-complete problem [Mor01].

Definition 2.7. **ITER**:

Input: Boolean circuit $C : [2^n] \to [2^n]$ with $C(1) > 1$.
Output: $v \in [2^n]$ such that either

- $C(v) < v$, or
- $C(v) > v$ and $C(C(v)) = C(v)$.

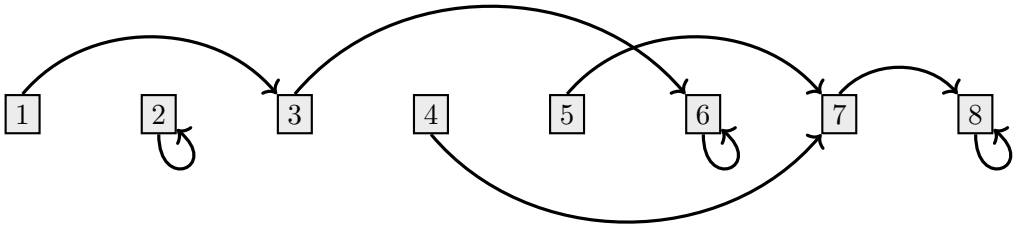


Figure 2.3: Example of an ITER instance C for $n = 3$. The 2^n $(= 8)$ nodes are represented by squares. The arrows indicate the mapping given by the circuit C. In this example, nodes 2, 6 and 8 are the fixed points of C. Any node that is mapped by C to a fixed point is a solution to the ITER instance. Thus, in this example, the solutions are nodes 3 and 7.

In this problem, it is convenient to think of the nodes in $[2^n]$ as lying on a line from left to right. Then, we are looking for any node v that is mapped to the left by C, or any node v that is mapped to the right and such that $C(v)$ is a fixed point of C. Since $C(1) > 1$, i.e., node 1 is mapped to the right, it is easy to see that such a solution must exist (apply C repeatedly on node 1). Note that the condition $C(1) > 1$ can be enforced syntactically, so this is indeed a TFNP problem and not a promise problem. See Figure 2.3 for an example of an ITER instance. Daskalakis and Papadimitriou [DP11] also provided a continuous problem that is PLS-complete. This problem is called REAL-LOCALOPT and we present a more general version of this problem in Appendix C.

The first natural problem to be proven PLS-complete is the graph-partitioning problem with the Kernighan-Lin heuristic [JPY88], and later with the swap heuristic [Sch91]. Subsequently, this was also proved for the traveling salesman problem with the Lin-Kernighan heuristic [Pap92] and also with the k-OPT neighbourhood [Kre89] (for some large constant k). The MAX-CUT problem with flip neighbourhood (i.e., move one vertex to the opposite side) is also PLS-complete [Sch91]. There are also PLS-completeness results for various local search problems related to SAT [Kre90; Kre89; Sch91; DM13]. A celebrated result is the PLS-completeness of computing a pure Nash equilibrium in a congestion game [FPT04], and various extensions and special cases [SV08; ARV08; BFH09].

2.3.4 The Class CLS

Daskalakis and Papadimitriou [DP11] noted that a lot of natural problems lie in both PPAD and PLS. Examples include finding a fixed point of a contraction map (CONTRACTION), solving the Linear Complementarity Problem for P-matrices (P-LCP), finding a stationary point of a polynomial, and computing a *mixed* Nash equilibrium in a congestion game.

In order to study the complexity of these problems, they defined a subclass of PPAD and PLS, which they called CLS for "Continuous Local Search." Unlike PPAD, PPA and PLS, CLS is defined using arithmetic circuits. Formally, the class CLS is defined as the set of all TFNP problems that reduce to the problem 3D-Continuous-Localopt.

Definition 2.8. **Continuous-Localopt**:

Input:

- precision/stopping parameter $\varepsilon > 0$,
- well-behaved arithmetic circuits $p : [0,1]^n \to [0,1]$ and $g : [0,1]^n \to [0,1]^n$,
- Lipschitz constant $L > 0$.

Output: An approximate local optimum of p with respect to g. Formally, find $x \in [0,1]^n$ such that

$$p(g(x)) \geq p(x) - \varepsilon.$$

Alternatively, we also accept one of the following violations as a solution:

- (p is not L-Lipschitz) $x, y \in [0,1]^n$ such that $|p(x) - p(y)| > L\|x - y\|$,
- (g is not L-Lipschitz) $x, y \in [0,1]^n$ such that $\|g(x) - g(y)\| > L\|x - y\|$.

For $k \in \mathbb{N}$, we let kD-Continuous-Localopt denote the problem Continuous-Localopt where n is fixed to be equal to k.

Continuous-Localopt is similar to Localopt (Definition 2.6), in the sense that we are looking for a minimum of p over the domain $[0,1]^n$ using the help of a function g. The membership of the problem in PLS and in PPAD is easy to show [DP11]. The membership in PPAD follows from the observation that g is a Brouwer function and that every (approximate) fixed point of g also yields a solution to the Continuous-Localopt instance.

Note that the original definition of Continuous-Localopt in [DP11] uses arithmetic circuits without the "well-behaved" restriction. As argued in Section 2.2.2, these circuits cannot always be evaluated efficiently, and so we instead use well-behaved arithmetic circuits, to ensure that the problem lies in TFNP. This subtle issue was recently also noticed by Daskalakis and Papadimitriou, who proposed a way to fix it in a corrigendum[4] to the definition of CLS. Their modification consists in having an additional input K (in unary) provided as part of the input such that the evaluation of the arithmetic circuit – purportedly – only involves numbers of bit-size at most $K \cdot \text{size}(x)$ on input x. Any point x where the arithmetic circuit fails to satisfy this property is accepted as a violation solution.

[4] http://people.csail.mit.edu/costis/CLS-corrigendum.pdf

Using well-behaved arithmetic circuits – instead of the solution proposed by Daskalakis and Papadimitriou – has the advantage that we do not need to add any additional inputs, or any additional violation solutions to our problems. Indeed, the restriction to well-behaved circuits can be enforced syntactically. Furthermore, we note that our problems defined with well-behaved circuits easily reduce to the versions using the solution proposed by Daskalakis and Papadimitriou (see Remark 2.2 below). Thus, this restriction only makes our hardness results stronger. In fact, for CLS we show that even restricting to linear arithmetic circuits (representing piecewise linear functions) does not make the class any weaker (see Section 7.4).

Remark 2.2. The proof of Lemma 2.1 (in Appendix A) shows that if we evaluate a well-behaved arithmetic circuit f on some input x, then, the value $v(g)$ at any gate g of the circuit will satisfy $\text{size}(v(g)) \leq 6 \cdot \text{size}(f)^3 \cdot \text{size}(x)$. As a result, it immediately follows that problems with well-behaved arithmetic circuits can be reduced to the versions of the problems with the modification proposed by Daskalakis and Papadimitriou in the corrigendum of their paper [DP11]. Indeed, it suffices to let $K = 6 \cdot \text{size}(f)^3$, which can be written down in unary. In particular, this holds for the definition of CLS.

In this thesis, we consider more general domains than just $[0,1]^n$ and so we also define a more general version of Continuous-Localopt.

Definition 2.9. **General-Continuous-Localopt**:

Input:

- precision/stopping parameter $\varepsilon > 0$,
- $(A, b) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m$ defining a bounded non-empty domain $D = \{x \in \mathbb{R}^n : Ax \leq b\}$,
- well-behaved arithmetic circuits $p : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^n$,
- Lipschitz constant $L > 0$.

Output: An approximate local optimum of p with respect to g on domain D. Formally, find $x \in D$ such that

$$p\big(\Pi_D(g(x))\big) \geq p(x) - \varepsilon.$$

Alternatively, we also accept one of the following violations as a solution:

- (p is not L-Lipschitz) $x, y \in D$ such that $|p(x) - p(y)| > L\|x - y\|$,
- (g is not L-Lipschitz) $x, y \in D$ such that $\|g(x) - g(y)\| > L\|x - y\|$.

Note that given $(A, b) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m$, it is easy to check whether the domain $D = \{x \in \mathbb{R}^n : Ax \leq b\}$ is bounded and non-empty by using linear programming.

We use the projection Π_D in this definition, because it is not clear whether there is some syntactic way of ensuring that $g(x) \in D$. Namely, it is unclear whether Π_D can be computed inside our arithmetic circuits. However, Π_D can be computed efficiently by a Turing machine, since it can be formulated as a convex quadratic program, known to be solvable in polynomial time [KTK80]. To be more precise, given a rational vector $x \in \mathbb{R}^n$, $\Pi_D(x)$ can be computed exactly in time $\text{poly}(\text{size}(x), \text{size}(A), \text{size}(b))$. Note that when $D = [0,1]^n$, the projection Π_D can easily be computed by arithmetic circuits, so Π_D is not needed in the definition of Continuous-Localopt. Indeed, when $D = [0,1]^n$, we have $[\Pi_D(x)]_i = \min\{1, \max\{0, x_i\}\}$ for all $i \in [n]$ and $x \in \mathbb{R}^n$.

The definition of CLS using 3D-Continuous-Localopt, instead of 2D-Continuous-Localopt, Continuous-Localopt, or General-Continuous-Localopt, leaves open various questions about whether all these different ways of defining it are equivalent. We prove that this is indeed the case. We discuss this, as well as the robustness of the definition of CLS with respect to other modifications in Section 7.4.

Following the definition of CLS by Daskalakis and Papadimitriou [DP11], two CLS-complete problems were identified: Banach [DTZ18] and MetametricContraction [FGMS17]. Banach is a computational presentation of Banach's fixed point theorem in which the metric is presented as part of the input (and could be complicated). Banach fixed points are unique, but CLS problems do not in general have unique solutions, and the problem Banach circumvents that obstacle by allowing certain violation solutions, such as a pair of points witnessing that f is not a contraction map. MetametricContraction is a generalisation of Banach, where the metric is replaced by a slightly relaxed notion called a meta-metric.

Some of the problems that motivated the original definition of CLS have been shown to lie in the class UEOPL defined by Fearnley et al. [FGMS20]. This holds in particular for the P-LCP problem and the Contraction problem with piecewise linear functions.

2.3.5 The Class PPAD ∩ PLS

The class PPAD ∩ PLS is the set of all TFNP problems that lie in both PPAD and PLS. A problem in PPAD ∩ PLS cannot be PPAD- or PLS-complete, unless PPAD ⊆ PLS or PLS ⊆ PPAD. Neither of these two containments is believed to hold, and, as mentioned above, this is supported by oracle separations between the classes. It is easy to construct "artificial" PPAD ∩ PLS-complete problems from PPAD- and PLS-complete problems. Let A and B be any TFNP problems.

Definition 2.10. **EITHER-SOLUTION**(A,B):

Input: An instance I_A of A and an instance I_B of B.
Output: A solution of I_A or a solution of I_B.

Then, it is quite easy to show that:

Proposition 2.1 (Daskalakis and Papadimitriou [DP11])**.** *If A is* PPAD*-complete and B is* PLS*-complete, then* EITHER-SOLUTION(A,B) *is* PPAD $\cap$ PLS*-complete.*

In fact, this holds more generally for any two TFNP classes. In particular, we obtain the following corollary, which we will use to show our main PPAD $\cap$ PLS-hardness result in Chapter 8.

Corollary 2.1. EITHER-SOLUTION(END-OF-LINE,ITER) *is* PPAD $\cap$ PLS*-complete.*

Prior to the work presented in this thesis, the problems EITHER-SOLUTION(A,B), where A is PPAD-complete and B is PLS-complete, were the only known PPAD $\cap$ PLS-complete problems.

2.4 The Class FIXP

Recall that the problem of computing an approximate Brouwer fixed point is PPAD-complete. An approximate fixed point of a function f is a point x such that $\|f(x) - x\| \leq \varepsilon$. However, note that such a point x might be far away from an actual exact fixed point.

In order to study *exact* fixed point computation, Etessami and Yannakakis [EY10] defined the class FIXP that captures the complexity of computing an exact fixed point of a function given by an arithmetic circuit and mapping the unit cube into itself. Since these exact fixed points can be irrational, it might seem at first that FIXP can only be used to study this problem in real-valued computational models. However, completeness results for this class usually use a special type of reduction, called an SL-reduction, which ensures that the reduction also holds for three types of problems that can be studied in the standard Turing machine model: the "strong approximation problem" (i.e., find a point close to an exact solution), the "partial computation problem" (i.e., compute the first n bits of an exact solution) and various decision problems. As a result, a FIXP-completeness result obtained through SL-reductions also implies that all these problems are hard in the standard Turing machine model.

Etessami and Yannakakis [EY10] proved that the problem of computing an exact Nash equilibrium of a 3-player game is FIXP-complete.[5] This means that computing a

[5]It is known that for 2 players a rational Nash equilibrium always exists and the problem of finding one is PPAD-complete [CDT09].

strong approximation for 3-player Nash is as hard as computing a strong approximation of a Brouwer function given by an arithmetic circuit.

Even though it is convenient to think of FIXP as the exact version of PPAD, it is perhaps more correct to think of FIXP as the exact version of a specific PPAD-complete problem (namely, the Brouwer fixed point problem). Indeed, it is possible that the exact version of some other PPAD-complete problem could give rise to a different class. For example, in Chapter 4 we show that the exact version of the PPAD-complete Hairy Ball problem is FIXP-hard, but it remains open whether it lies in FIXP – and it could very well turn out to be strictly harder than FIXP.

The exact versions of some problems complete for other TFNP classes have also been studied. The class BU capturing an exact version of the PPA-complete Borsuk-Ulam theorem was defined by Deligkas et al. [DFMS21] and further studied by Batziou, Hansen and Høgh [BHH21]. Daskalakis and Papadimitriou [DP11] also briefly discuss the exact versions of CLS- and PLS-complete problems.

FIXP. We now provide a formal definition of FIXP. A real-valued search problem Π associates to any input instance I (encoded as a bit-string) a search space $D_I \subseteq \mathbb{R}^{d_I}$ and a set of solutions $\mathrm{Sol}(I)$. We assume that given I, we can compute a description of D_I in polynomial time. In this thesis, we only consider total real-valued search problems, i.e., we assume that $\mathrm{Sol}(I) \neq \emptyset$.

We begin by defining basic FIXP problems that directly correspond to the problem of finding an exact Brouwer fixed point. The class FIXP is then obtained by taking the closure with respect to SL-reductions, also defined below.

In the context of FIXP, an arithmetic circuit is allowed to use gates in $\{+, -, \times, /, \max, \min\}$ as well as rational constants. Note that, in particular, such arithmetic circuits are always continuous. Etessami and Yannakakis [EY10] have shown that the definition of FIXP is robust to some changes to the gate set (e.g., the division gate $/$ is not actually needed, and we can also use gates $\sqrt[k]{}$ where k is given in unary).

Basic FIXP problems. A real-valued search problem Π is a basic FIXP problem if for every instance I:

- the domain D_I is a nonempty compact set described by linear inequalities with rational coefficients, and
- I contains the description of an arithmetic circuit $F_I : D_I \to D_I$ such that $\mathrm{Sol}(I) = \{x \in D_I : F_I(x) = x\}$.

There is an implicit promise here that F_I indeed maps any point from D_I to a point in D_I, and that the arithmetic circuit never divides by zero on any valid input from D_I.

SL-reductions. Let Π_1 and Π_2 be two real-valued search problems. An SL-reduction from Π_1 to Π_2 is a pair of polynomial-time computable functions (f, g) where f maps instances of Π_1 to instances of Π_2 and the following holds. For any instance I of Π_1: if $y \in \mathrm{Sol}_{\Pi_2}(f(I))$, then $g(I, y) \in \mathrm{Sol}_{\Pi_1}(I)$. Furthermore, we also require that g be separable linear. Namely, given I we can in polynomial time compute rational constants a_i, b_i, $i = 1, \ldots, d_I$, such that $x = g(I, y)$ is given by $x_i = a_i y_j + b_i$ (for some j) for all i.

In Chapter 4 we show that computing exact solutions of the Hairy Ball theorem is FIXP-hard. In Chapter 5 we provide an *exact* Generalised Circuit problem that is very convenient for proving FIXP-hardness results. We present an application of this tool in Chapter 6, where we show that computing exact equilibria in First Price auctions with subjective priors is FIXP-complete.

Part I

PPAD

Chapter 3

Multiple-Source End-of-Line: One Source to Rule Them All

The class PPAD is usually defined using the canonical complete problem End-of-Line, as explained in Section 2.3.1. Recall that in this problem we are given a (succinctly-represented) directed graph with indegree/outdegree at most 1, as well as a source of the graph, and the goal is to find any other end of line, i.e., a sink or another source. It is easy to see that a solution is guaranteed to exist by a simple combinatorial principle.

In this chapter, we consider various generalisations of this principle and investigate the complexity of the resulting total search problems. In particular, we study the following problems on graphs with indegree/outdegree at most 1:

- given k sources, find any other end of line
- given k sources, find k other ends of line
- given k sources, find k sinks or 100 sources.

We prove that these multiple-source variants of End-of-Line remain PPAD-complete, thus providing further evidence that the class is very robust. On the other hand, if we insist on only accepting sinks as solutions, then these problems become PPADS-complete.

Our techniques also allow us to provide the first full[1] proof of PPAD-completeness for the Imbalance problem, in which we are given a general directed graph and an unbalanced vertex (indegree different from outdegree), and have to find another unbalanced vertex. As a result, PPAD indeed embodies the general combinatorial principle: "every directed graph with an unbalanced node must have another."

From a more practical standpoint, these results yield new tools for proving membership in PPAD. An application of these tools is provided in the next chapter (Chapter 4), where a multiple-source variant of End-of-Line plays a crucial role in resolving the complexity of the Hairy Ball theorem.

[1] Although this result had previously been stated by Beame et al. [BCE+98], the proof was incomplete, since it could only handle the case where the given vertex had imbalance 1. See Section 3.2 for more details.

3.1 Multiple-Source End-of-Line

In the End-of-Line problem (Definition 2.2), we are given a directed graph where each vertex has in- and out-degree at most 1 and a known source of this graph, and we wish to find a sink or another source. But, what if, instead of just one, we already know *two* sources of an End-of-Line instance? We are still interested in finding any sink or any *other* source. Intuitively, the problem might seem easier, because the existence of two sources implies the existence of at least two sinks, hence more potential solutions. In fact, it is easy to see that this problem is actually *at least as hard* as End-of-Line: just duplicate the whole End-of-Line instance.

The other direction, however, is not trivial. Indeed, if we interpret our 2-source End-of-Line instance as a standard End-of-Line instance (and pick one of the two sources as the standard source), then the other known source is a valid solution to End-of-Line, but not a valid solution to our original problem. In other words, it is not clear how to solve this problem if we are given access to an oracle solving End-of-Line, because the oracle could just return the other known source. We consider the following more general problem, where we are given an End-of-Line graph and an explicit list of known sources.

Definition 3.1. Multiple-Source End-of-Line (MS-EoL):
Input: Boolean circuits $S, P : \{0,1\}^n \to \{0,1\}^n$ and $s_1, \ldots, s_k \in \{0,1\}^n$ such that $P(s_i) = s_i \neq S(s_i)$ for all i.
Output: $x \in \{0,1\}^n$ such that $P(S(x)) \neq x$ or $x \notin \{s_1, \ldots, s_k\}$ such that $S(P(x)) \neq x$.

In passing, let us note that in the undirected case this kind of generalisation is trivial. Recall that the undirected analogue of End-of-Line is Leaf (Definition 2.4): given an undirected graph where every vertex has degree at most 2 and given a vertex of degree 1, find another vertex of degree 1, i.e., another leaf. Assume that we know k leaves instead of just one. If k is even, then the problem is not in TFNP. If k is odd, then we can add edges between known leaves until exactly one is left. Thus, the problem is equivalent to Leaf. This kind of reduction does not work for the directed case. Nevertheless, we obtain[2]:

Theorem 3.1. Multiple-Source End-of-Line *is equivalent to* End-of-Line.

[2]This problem was discussed in an online thread (https://cstheory.stackexchange.com/q/37481). E. Jeřábek proved membership in PPADS and PPA-p for every prime p (but not membership in PPAD).

Remark (Multiple Known Sources and Sinks). A natural generalisation of Multiple-Source End-of-Line is the following problem: given an End-of-Line graph and a list of k known sources and m known sinks, find another source or sink. Note that for this problem to be in TFNP, we must require $k \neq m$. Using Theorem 3.1, it is easy to see that this problem is equivalent to End-of-Line. If $k > m$, then we add an edge from each of the m known sinks to some corresponding known source and obtain an instance with $k - m$ known sources and no known sinks. Similarly, if $k < m$, then we first reverse all directed edges and then apply the same trick.

We now give the proof of Theorem 3.1. The next two sections then present additional consequences of this result.

Proof of Theorem 3.1. The reduction from End-of-Line to MS-EoL is trivial. The challenging step is the reduction from MS-EoL to End-of-Line.

Let (S, P) be an instance of MS-EoL with a list of k known sources, where the vertex set is $\{0,1\}^n$, also interpreted as $\{0, \ldots, 2^n - 1\}$. Without loss of generality, we assume that the known sources are $0, 1, 2, \ldots, k-1$. This is easy to achieve by applying an efficient bijection on the vertex set.

For simplicity, we are going to assume that $P(S(z)) = z$ for all $z < k$. We also assume that for all x we have $P(S(x)) = x$ unless $S(x) = x$, and $S(P(x)) = x$ unless $P(x) = x$. The first assumption corresponds to requiring that the first k vertices indeed be sources. Note that we can check this using $O(k)$ evaluations of the circuits, i.e., in polynomial time in $\text{size}(S) + \text{size}(P) + kn$, and any "false" source is a solution to the MS-EoL instance. Note that $\text{size}(S) + \text{size}(P) + kn$ essentially corresponds to the size of the input, namely the sum of the sizes of the two circuits and the length of the list of known sources given in the input. The second assumption corresponds to requiring that the graph is in some sense well-defined. This requirement can be enforced by a slight modification of the circuits that can be done in time polynomial in $\text{size}(S) + \text{size}(P) + kn$. Any solution of the modified instance is a solution of the original instance.

Let V^s, V^t, V^p be the following subsets of $\{0, \ldots, 2^n - 1\}$:

- $V^s = \{x : P(x) = x, S(x) \neq x\}$. This corresponds to all the sources of the graph.
- $V^t = \{x : S(x) = x, P(x) \neq x\}$. This corresponds to all the sinks of the graph.
- $V^p = \{x : P(x) \neq x, S(x) \neq x\}$. This corresponds to all the vertices that are not isolated, but are neither sources nor sinks. We call those *path vertices*.

Clearly, members of all those subsets are recognisable in polynomial time in size(S) + size$(P) + kn$. Let $V = V_P \cup V_S \cup V_I$. Note that isolated vertices are not contained in V.

Let $G = (V, E)$ be the graph represented by circuits S, P (without isolated vertices). Below we will give an inductive construction of the graph $G_k = (V_k, E_k)$ that will have the following properties:

- The sources of G_k are all the sets of the form $\{s_1, \ldots, s_k\}$, where $s_1, \ldots, s_k \in V^s$ are distinct sources of the original graph.
- The sinks of G_k are all the sets of the form $\{t_1, \ldots, t_k\}$, where $t_1, \ldots, t_k \in V^t$ are distinct sinks of the original graph.
- Every vertex of G_k can be represented using at most $kn + k^2$ bits. There exists a polynomial algorithm that for each such bit-string decides whether it represents a vertex of G_k or not.
- The successor and predecessor circuits S_k, P_k have polynomial size with respect to size(S) + size$(P) + kn$ and can be constructed in polynomial time.

Note that the only known source of G_k is $\{0, 1, \ldots, k-1\}$. Any other source or any sink of G_k contains at least one unknown source or sink of the original graph. Thus, if we can construct (in polynomial time in size(S) + size$(P) + kn$) circuits P_k, S_k that represent this graph, then we have a polynomial reduction from MS-EoL to End-of-Line.

We now give a formal inductive construction of G_ℓ. For $\ell = 1$, G itself already satisfies these properties (if we interpret any vertex x as $\{x\}$). Let $\ell \geq 2$. Assume that we know how to construct G_i for all $1 \leq i \leq \ell - 1$. We then construct G_ℓ as follows. For any set X and any $j \in \mathbb{N}$, let Subsets$(X, j) := \{A \subseteq X : |A| = j\}$, i.e., the set of all subsets of X with cardinality exactly j. The set of vertices of G_ℓ is defined as

$$V_\ell = \text{Subsets}(V, \ell) \cup \bigcup_{i=1}^{\ell-1} \left(\text{Subsets}(V^p, i) \times V_{\ell-i}\right).$$

Let us investigate the number of bits needed to represent an element in V_ℓ. We need $n \cdot i$ bits to represent an element in Subsets(V^p, i). By induction hypothesis, $(\ell-i)n+(\ell-i)^2$ bits suffice to represent an element in $V_{\ell-i}$. Thus, for any $i \in \{1, \ldots, \ell-1\}$, at most $n + (\ell-1)n + (\ell-1)^2$ bits suffice to represent an element in $V_{\ell-i}$. We add another $\lceil \log_2(\ell-1) \rceil \leq \ell - 1$ bits to explicitly store the value of i. Thus, we can represent any element in $\bigcup_{i=1}^{\ell-1} (\text{Subsets}(V^p, i) \times V_{\ell-i})$ using at most $\ell n + (\ell-1)^2 + \ell - 1$ bits. We add one more bit to decide whether the element is in Subsets(V, ℓ) or not. We only

need ℓn bits to represent an element in Subsets(V, ℓ). Putting everything together, we get an upper bound on the number of bits needed to represent an element in V_ℓ

$$1 + \max\{\ell n, \ell n + (\ell - 1)^2 + \ell - 1\} = \ell n + (\ell - 1)^2 + \ell \leq \ell n + \ell^2.$$

Furthermore, given a bit-string, it is easy to "decode" it and decide whether it is a vertex of G_ℓ or not (and if it is not, then treat it as an isolated vertex).

We now give the construction of the predecessor and successor circuits. First, consider the case $\{x_1, \ldots, x_\ell\} \in$ Subsets(V, ℓ). Assume that we have reordered the elements in the set such that for some $i, j \in \{0, \ldots, \ell\}$ we have $x_1, \ldots, x_i \in V^p$, $x_{i+1}, \ldots, x_j \in V^s$ and $x_{j+1}, \ldots, x_\ell \in V^t$.

- If $j = \ell$ (i.e., only sources and path vertices), then we define

  $$S_\ell(\{x_1, \ldots, x_\ell\}) = \{S(x_1), \ldots, S(x_\ell)\}$$

  Furthermore, if we also have $1 \leq i \leq \ell - 1$ (i.e., at least one path vertex and source), then we define

  $$P_\ell(\{x_1, \ldots, x_\ell\}) = (\{x_1, \ldots, x_i\}, \{x_{i+1}, \ldots, x_\ell\}) \in \text{Subsets}(V^p, i) \times V_{\ell-i}$$

- If $i = j$ and $j < \ell$ (i.e., only path vertices and sinks), then we define

  $$P_\ell(\{x_1, \ldots, x_\ell\}) = \{P(x_1), \ldots, P(x_\ell)\}$$

  Furthermore, if we also have $i \geq 1$ (i.e., at least one path vertex and sink), then we define

  $$S_\ell(\{x_1, \ldots, x_\ell\}) = (\{x_1, \ldots, x_j\}, \{x_{j+1}, \ldots, x_\ell\}) \in \text{Subsets}(V^p, j) \times V_{\ell-j}$$

  (recall that we are in the case $i = j$).

- If $i < j$ and $j < \ell$ (i.e., both sources and sinks, as well as path vertices potentially), then $\{x_1, \ldots, x_\ell\}$ is an isolated vertex.

Now consider the case $(\{x_1, \ldots, x_i\}, z) \in$ Subsets$(V^p, i) \times V_{\ell-i}$ for some $i \in \{1, \ldots, \ell - 1\}$.

- We define $P_\ell((\{x_1, \ldots, x_i\}, z)) = (\{x_1, \ldots, x_i\}, S_{\ell-i}(z))$, except if $S_{\ell-i}(z) = z$ and $P_{\ell-i}(z) \neq z$, in which case $z \in$ Subsets$(V^t, \ell - i)$ (by induction hypothesis) and we then define $P_\ell((\{x_1, \ldots, x_i\}, z)) = \{x_1, \ldots, x_i\} \cup z \in$ Subsets(V, ℓ). Note that the two sets in this union are disjoint, because $x_1, \ldots, x_i$ are path vertices of G, whereas z only contains sinks of G.

- We define $S_\ell((\{x_1, \ldots, x_i\}, z)) = (\{x_1, \ldots, x_i\}, P_{\ell-i}(z))$, except if $P_{\ell-i}(z) = z$ and $S_{\ell-i}(z) \neq z$, in which case $z \in \text{Subsets}(V^s, \ell - i)$ (by induction hypothesis) and we then define $S_\ell((\{x_1, \ldots, x_i\}, z)) = \{x_1, \ldots, x_i\} \cup z \in \text{Subsets}(V, \ell)$.

It is straightforward to check that in the graph represented by S_ℓ, P_ℓ every vertex has in- and out-degree at most 1. Furthermore, by construction we also get that the sources of G_ℓ are exactly the vertices in $\text{Subsets}(V^s, \ell)$ and the sinks of G_ℓ are exactly the vertices in $\text{Subsets}(V^t, \ell)$. By induction it follows that we can construct (in polynomial time in $\text{size}(S) + \text{size}(P) + kn$) circuits S_k, P_k that represent G_k. □

3.2 The Imbalance Problem

Up to this point, we have only considered graphs where every vertex has in- and out-degree at most 1. However, the principle that guarantees the existence of a solution in an End-of-Line graph can be generalised to higher degree graphs. If we are given a directed graph and an *unbalanced* vertex, i.e. a vertex with in-degree $\neq$ out-degree, then there must exist another unbalanced vertex.

Beame et al. [BCE+98] defined the corresponding problem Imbalance, which is seemingly more general than End-of-Line. In this problem, every vertex is not constrained to have in- and out-degree at most 1. Instead, in- and out-degree are bounded by some polynomial of the input length.[3] We are given a vertex that is unbalanced and have to find another unbalanced vertex (which is guaranteed to exist).

Beame et al. [BCE+98] claim that Imbalance reduces to End-of-Line, using the same argument as for the corresponding problems on undirected graphs. However, if the graph is directed, a complication arises (that is not an issue in the undirected case). Indeed, their proof idea is incomplete, because they overlook the fact that their reduction yields an End-of-Line instance with *multiple* known sources. Using Theorem 3.1 we can provide a full proof of their claim.

Theorem 3.2. Imbalance *is* PPAD*-complete.*

Let us now define the problem formally. Similarly to End-of-Line, the graph is provided implicitly through circuits $S, P : \{0,1\}^n \to \mathsf{Set}_{\leq d}(\{0,1\}^n)$ computing successors and predecessors respectively. Note, however, that in this case we need the circuits to output *sets* of successors or predecessors and so we use the Set-notation introduced in Section 2.2.1. We syntactically enforce that $x \notin S(x)$ and $x \notin P(x)$. We say that a directed edge (x, y) exists, if and only if $y \in S(x)$ and $x \in P(y)$.

[3] Note that this trivially holds, if the input consists of circuits that explicitly output the predecessor and successor list.

Definition 3.2. IMBALANCE:

Input: Boolean circuits $S, P : \{0,1\}^n \to \mathsf{Set}_{\leq d}(\{0,1\}^n)$ and a vertex $z \in \{0,1\}^n$ with $|S(z)| \neq |P(z)|$.
Output: Either

1. $x \in \{0,1\}^n \setminus \{z\}$ such that $|S(x)| \neq |P(x)|$, or
2. $x, y \in \{0,1\}^n$ such that $y \in S(x) \wedge x \notin P(y)$ or $y \notin S(x) \wedge x \in P(y)$.

Note that the number of incoming and outgoing edges at a vertex is bounded by a polynomial in the size of the input, because d is bounded by the size of the circuits. Beame et al. [BCE+98] defined this problem in a black box model, namely the predecessor and successor functions are oracles. Since our reductions do not make use of the white box aspect of the input circuits (they don't look *inside* the circuit, but just use it as a black box), the results also hold in their model.

Proof of Theorem 3.2. PPAD-hardness is trivial, since IMBALANCE generalises END-OF-LINE. To show membership in PPAD we follow the proof idea given by Beame et al. [BCE+98].

Recall that the undirected analogue of END-OF-LINE is LEAF (Definition 2.4): given an undirected graph where every vertex has degree at most 2 and given a vertex of degree 1, find another vertex of degree 1, i.e. another leaf. The generalisation of LEAF is called ODD: given an undirected graph and a vertex of odd degree, find another vertex of odd degree. It is quite straightforward to show that ODD is equivalent to LEAF. Clearly, LEAF trivially reduces to ODD. To show that ODD reduces to LEAF, Papadimitriou [Pap90; Pap94] and Beame et al. [BCE+98] use the *chessplayer algorithm* (which is based on an Euler tour argument). Intuitively, the idea is to separate vertices into multiple copies so that every copy has degree at most two.

As noted by Papadimitriou [Pap90] and Beame et al. [BCE+98], the chessplayer algorithm can also be applied to the directed case. Let (S, P, z) be an instance of IMBALANCE, where $S, P : \{0,1\}^n \to \mathsf{Set}_{\leq d}(\{0,1\}^n)$. First of all, note that we can assume wlog that all edges are well-defined, i.e., we have $y \in S(x) \Leftrightarrow x \in P(y)$. This can be achieved by a simple modification of the circuits: the modified circuit S' outputs $\{y \in S(x) : x \in P(y)\}$ instead of $S(x)$. Along with the analogous modification for P, this yields an instance where the solutions can only be of the first type, and any such solution yields a solution (of the first or second type) of the original instance.

Note that for any $x \in \{0,1\}^n$, the successor and predecessor lists can be ordered lexicographically. For any $x \in \{0,1\}^n$ and $i \geq 1$, let $S_i(x) \in \{0,1\}^n$ denote the ith successor of x, if $S(x)$ is ordered lexicographically. If x has less than i successors,

then let $S_i(x) = x$. Finally, let $\#_x^S(y)$ correspond to the index of $y \in \{0,1\}^n$ in the successor list of $x \in \{0,1\}^n$, i.e., $S_{\#_x^S(y)}(x) = y$. Define $P_i(x)$ and $\#_x^P(y)$ analogously.

Let $\ell = \lceil \log_2 d \rceil$. We construct an End-of-Line instance on the vertex set $\{0,1\}^{n+\ell}$. For convenience, we use the notation $(x,i) \in \{0,1\}^n \times [2^\ell]$ to denote elements in $\{0,1\}^{n+\ell}$. The End-of-Line circuits $\widehat{S}, \widehat{P} : \{0,1\}^{n+\ell} \to \{0,1\}^{n+\ell}$ are constructed as follows. On input (x,i) the circuit $\widehat{S}$ first computes $y = S_i(x)$. If $y = x$, then it outputs (x,i). Otherwise, it outputs $(y, \#_y^P(x))$. Similarly, on input (x,i), $\widehat{P}$ computes $y = P_i(x)$ and outputs $(y, \#_y^S(x))$ if $y \neq x$, and (x,i) otherwise. Note that $\widehat{S}, \widehat{P}$ can be constructed in polynomial time in the size of S and P.

It is easy to see that for any balanced vertex x in the graph given by (S,P), all of its versions $(x,\cdot)$ in the graph given by $(\widehat{S},\widehat{P})$ will either have in- and out-degree one, or be isolated. Thus, if (x,i) is a source or sink, then x is unbalanced in the graph given by (S,P). Furthermore, all edges are well-defined.

Beame et al. [BCE+98] claim that this reduction is sufficient to prove that Imbalance reduces to End-of-Line (which they call SOURCE-OR-SINK). However, consider the case where the imbalance in the known unbalanced vertex z is strictly greater than one, i.e., $||S(z)| - |P(z)|| \geq 2$. Then, there exist $i \neq j$ such that (z,i) and (z,j) are both sources or both sinks. For example, if $|S(z)| = 2$ and $|P(z)| = 0$, then $(z,1)$ and $(z,2)$ are sources, and all the other (z,i) are isolated vertices. If we consider this as an End-of-Line instance and pick $(z,1)$ as the known source, then $(z,2)$ is a valid solution. However, note that it does not yield a solution to the Imbalance instance.

Using our results on multiple-source End-of-Line we can complete the proof. First of all, we can make sure that z is in deficiency and not in excess, i.e., $|S(z)| - |P(z)| \geq 1$, simply by inverting the role of S and P. Then, the reduction described above yields an End-of-Line instance with known sources $(z, i+1), (z, i+2), \ldots, (z, i+j)$, where $i = |P(z)|$ and $j = |S(z)| - |P(z)| \geq 1$. Note that all other $(z,\cdot)$ are neither sources, nor sinks. Since we can efficiently produce an explicit list of all the known sources, we obtain an instance of MS-EoL, which lies in PPAD by Theorem 3.1. □

3.3 Looking for Multiple Solutions

If we are given an End-of-Line instance with k known sources, then we can ask for k sinks or k unknown sources. The problem is total, because at least k sinks are guaranteed to exist. For any $k \in \mathbb{N}$ we can define the following problem, where we interpret $\{0,1\}^n$ as $\{0,1,\ldots,2^n-1\}$.

Definition 3.3. ***k*-Ends-of-Line (*k*-EoL)**:

Input: Boolean circuits $S, P : \{0,1\}^n \to \{0,1\}^n$ such that $P(z) = z \neq S(z)$ for all $z < k$.

Output: Distinct $x_1, \ldots, x_k$ such that $P(S(x_i)) \neq x_i$ for all i or $S(P(x_i)) \neq x_i \geq k$ for all i.

Intuitively, this problem seems harder than End-of-Line or MS-EoL, because we are now looking for more than one solution. However, using Theorem 3.1 we can show:

Theorem 3.3. *For any $k \in \mathbb{N}$, k-Ends-of-Line is* PPAD*-complete.*

Proof. The non-trivial direction is the reduction to End-of-Line. Buss and Johnson [BJ12] have shown that PPAD, PPADS, PPA and PLS are closed under Turing reductions, by providing a way to transform a Turing reduction (to a complete problem of each of those classes) into a many-one reduction. Thus, it suffices to provide a Turing reduction from k-Ends-of-Line to End-of-Line. By Theorem 3.1 we can efficiently simulate an oracle for MS-EoL, using an oracle for End-of-Line. We can solve an instance of k-EoL by making repeated calls to MS-EoL oracles, with a list of all the currently known sources. If the oracle call returns a new source, then we add it to our list. If the oracle returns a sink, then we add an edge from this sink to one of the known sources, and remove that source from the list. It is easy to see that after at most $2k$ oracle calls we will have obtained at least k sinks or at least k new sources. Note that the list of known sources will have length at most $2k$. □

Remark. Note that the proof still works if we are given an explicit list of the known sources in the input (as in the definition of MS-EoL), i.e., k does not have to be fixed. Furthermore, the same proof also yields PPAD-completeness for the following problem. Fix some polynomial p. The problem is: given k sources, find k sinks or $p(k)$ sources. This seems quite surprising, as one might have expected this problem to be closer to PPADS.

We close this section by giving some analogous results for the class PPADS and its canonical complete problem Sink (Definition 2.3). Recall that Sink is identical to End-of-Line, except that we only accept a sink as a solution and are not interested in other sources.

In this case, the results are easier to obtain, because there is no need for an analogue of Theorem 3.1. Indeed, unlike for End-of-Line, here it is easy to prove that multiple source Sink is equivalent to Sink. Consider the problem MS-Sink, where we are given a graph and a list of known sources and are looking to find a sink. It is easy to see that this problem is equivalent to Sink. A reduction from Sink to MS-Sink is given by simply taking k copies of the original Sink instance graph. The

reduction in the other direction is even more trivial. Indeed, we can just ignore the extra $k-1$ sources we know, because we are only interested in sinks.

For any $k \in \mathbb{N}$, we can define the analogous problem to k-EoL, where we look for multiple sinks.

Definition 3.4. ***k*-Sinks**:

Input: Boolean circuits $S, P : \{0,1\}^n \to \{0,1\}^n$ such that $P(z) = z \neq S(z)$ for all $z < k$.

Output: Distinct $x_1, \ldots, x_k$ such that $P(S(x_i)) \neq x_i$ for all i.

Theorem 3.4. *For any $k \in \mathbb{N}$, k-Sinks is* PPADS*-complete.*

Proof. Sink easily reduces to k-Sinks by taking k copies of the graph. The other direction can again be proved by using the result by Buss and Johnson [BJ12]. A Turing reduction from k-Sinks to Sink is obtained by doing the following: given an instance of k-Sinks, use the oracle to solve the Sink problem on this instance, then add an edge from the sink we just obtained to one of the known sources. Doing this k times yields k distinct sinks of the original instance. □

Remark. Just like Theorem 3.3, this result also holds if there is a polynomial number of sources, in particular if they are given explicitly in the input.

3.4 Conclusion and Future Directions

Our results on multiple-source variants of End-of-Line yield stronger tools for showing membership of PPAD, which can be used to put further problems in this class. In the next chapter, we make use of these tools to resolve the complexity of the Hairy Ball theorem.

Furthermore, our results proving that End-of-Line variants remain PPAD-complete show that the definition of PPAD based on End-of-Line is very robust. This robustness of PPAD is perhaps also one of the reasons why this class captures the complexity of so many problems. In particular, we note the perhaps surprising result that if we are given one source and seek one sink or n^{100} other sources, then the problem is still in PPAD.

Our results on PPAD and PPADS also provide further evidence suggesting that the number of (most likely) non-trivial interesting subclasses of TFNP is rather small. Indeed, despite extensive investigation of problems in TFNP in the last 20 years, most of them seem to end up being complete for one of the already known subclasses. Building on the proof techniques developed in this chapter, Ishizuka [Ish21] recently provided similar robustness results for the class EOPL.

A natural direction for future work would be to look at End-of-Line variants where we have a super-polynomial number of given sources. In this case the sources would have to be provided implicitly, e.g., by saying that all vertices smaller than some super-polynomial value $k(n) \in [2^n]$ are sources. To solve the problem one would have to provide a new source, a sink, or a vertex in $[k]$ that is actually not a source. If the number of sources is a polynomial fraction of the total number of vertices, then the problem can be solved efficiently using a straightforward randomised algorithm. However, there remains a gap between those two extreme cases, where the complexity of the problem is not known.

Chapter 4

The Complexity of the Hairy Ball Theorem

The Hairy Ball Theorem (HBT) is a well-known topological theorem stating that there is no non-vanishing continuous tangent vector field on an even-dimensional k-sphere. It has various informal statements such as "you can't comb a hairy ball flat without creating a cowlick,"[1] or "there is a point on the surface of the earth with zero horizontal wind velocity."

The HBT was first proved in 1885 by Poincaré [Poi85] for the case $k = 2$. The theorem as stated for all even k was proved in 1911 by Brouwer [Bro11]. Accordingly, this result is sometimes also called the Poincaré-Brouwer theorem. In fact, the result proved by Poincaré [Poi85] is stronger than stated above. It follows from it that for any (sufficiently well-behaved) 2-dimensional manifold with genus $g \neq 1$, any continuous tangent vector field must have a zero. In particular, this means that the HBT also holds for the torus of genus g for $g \geq 2$, i.e., the 2-dimensional torus with g holes. It is easy to see that it does not hold for the standard single-hole torus.

Over the years, various papers in the *American Mathematical Monthly* have presented alternative proofs of the Hairy Ball Theorem and variants, for example [JT04; Mil78; Boo71; EG79; McG16; Cur18].

Given the long-standing interest in the Hairy Ball Theorem, it is natural to study the corresponding computational search problem. In this chapter, we prove that computing an approximate zero of a Hairy Ball vector field is PPAD-complete. Recall that the computational problems associated to Brouwer's fixed point theorem and the Borsuk-Ulam theorem are PPAD- and PPA-complete respectively (Sections 2.3.1 and 2.3.2). Thus, this complexity-theoretic analysis of topological search problems provides a well-defined sense in which the HBT is "Brouwer-like" rather than "Borsuk-Ulam-like." It has previously been noted that the HBT may be used to prove Brouwer's fixed point theorem [Mil78], but not the other way around. Indeed, the

[1] https://en.wikipedia.org/wiki/Hairy_ball_theorem

existing proof of HBT using Sperner's Lemma [JT04] actually uses a generalisation of Sperner's Lemma, which was not known to be equivalent to Brouwer's fixed point theorem prior to our work.

While many PPAD-completeness results already exist, a noteworthy novelty of our results is that we find that computing HBT solutions corresponds with *multiple-source* variants of the End-of-Line problem, as studied in Chapter 3. This is in contrast with previous PPAD-complete problems that naturally reduce to standard single-source End-of-Line. The importance of the multiple-source aspect becomes even more apparent in the study of the HBT on the torus of genus g for $g \geq 2$, for which we have also proved PPAD-completeness in [GH21].

The generalisation of Poincaré's result to higher dimensions is called the Poincaré-Hopf theorem (see, e.g., [GP74]). This theorem relates the number and types of zeros of a vector field on a manifold with its Euler characteristic, a topological invariant. In particular, if the Euler characteristic of a manifold is not 0, then any continuous tangent vector field on the surface must have a zero. The Euler characteristic of even-dimensional spheres is 2, while it is $2(1-g)$ for 2-dimensional toruses of genus $g \geq 2$. For odd-dimensional spheres it is 0.

We believe that the reduction to multiple-source End-of-Line is not an artefact of our techniques, but instead intrinsically related to the Euler characteristic of the domain. Indeed, the reduction from the HBT problem on even-dimensional spheres to End-of-Line yields 2 sources (Section 4.2). On the other hand, if one considers the HBT problem on the 2-dimensional torus of genus $g \geq 2$, then $2(g-1)$ sources are obtained [GH21]. As a sanity check, if one attempts to carry out the reduction for the problem on odd-dimensional spheres (where the HBT does *not* hold), then it yields an End-of-Line instance with one known source and one known sink. This is not a valid instance, since it is not guaranteed to have a solution, i.e., another source or sink.

Finally, we note that PPAD-hardness is obtained by constructing a HBT vector field from multiple copies of a Brouwer fixed point problem. The usage of multiple copies is a new conceptual feature, closely related to the multi-source aspect. Using the same high-level idea, we also provide a FIXP-hardness result for the problem of computing an *exact* solution (Section 4.3.1).

4.1 The Hairy-Ball Problem

The k-dimensional unit sphere in $\mathbb{R}^{k+1}$ (or k-sphere) is denoted $S^k = \{x \in \mathbb{R}^{k+1} : \|x\|_2 = 1\}$. A continuous tangent vector field on S^k is a continuous function $f : S^k \to \mathbb{R}^{k+1}$ such that for all $x \in S^k$ we have $\langle f(x), x\rangle = 0$, where $\langle \cdot, \cdot \rangle$ denotes the inner product. The Hairy Ball Theorem can be stated as follows:

Theorem 4.1 (Poincaré [Poi85]–Brouwer [Bro11])**.** *If $k \geq 2$ is even, then for any continuous tangent vector field $f : S^k \to \mathbb{R}^{k+1}$, there exists $x \in S^k$ such that $f(x) = 0$.*

4.1.1 The kD-Hairy-Ball problem

The Hairy Ball Theorem naturally yields a corresponding computational problem. We are given a continuous tangent vector field f on the unit sphere and have to find a point where it is zero. In trying to formalise this, some issues need to be addressed.

The first issue is that the vector field might not have a rational zero. Indeed, consider the following example: at $x \in S^2$ the vector field is simply the vector $(1, 1, 1)$ projected onto the tangent space of S^2 at x. In this case, the only solutions are $\pm(1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3})$. Thus, we cannot hope to always output an exact solution. We bypass this problem by asking for an *approximate* solution instead, i.e., a point $x \in S^2$ such that $\|f(x)\|_\infty \leq \varepsilon$ for some $\varepsilon > 0$ provided in the input. This notion of approximation is the standard one used when studying topological existence theorems in the context of TFNP (e.g., Brouwer's fixed point theorem or the Borsuk-Ulam theorem).

The second issue is that we have to decide how our vector field will be represented in the input. Here we make the choice of using linear arithmetic circuits (Section 2.2.2). Namely, a circuit F with $k+1$ inputs and outputs represents a function $F : S^k \to \mathbb{R}^{k+1}$. These circuits yields continuous piecewise linear functions. Furthermore, they can be evaluated efficiently (Lemma 2.1) and are Lipschitz-continuous with a Lipschitz-constant of polynomial representation size (Lemma 2.2). One potential issue is that the vector field given by a linear arithmetic circuit F might not be *tangent* to the sphere, but this is easy to fix by simply considering the vector field given by the projection of F onto the corresponding tangent space to the sphere. Thus, for any even $k \geq 2$, we define the computational problem as follows:

Definition 4.1. **kD-Hairy-Ball**:

Input: Linear arithmetic circuit $F : S^k \to \mathbb{R}^{k+1}$ and $\varepsilon > 0$.
Output: $x \in S^k$ such that $\|P_x[F(x)]\|_\infty \leq \varepsilon$.

Here $P_x[\cdot]$ denotes the projection onto the tangent space to the sphere S^k at $x \in S^k$. Note that for any $v \in \mathbb{R}^{k+1}$, we have $P_x[v] = v - \langle v, x\rangle x$, because $\|x\|_2 = 1$. Thus, the projection of any rational vector v onto the tangent space at rational $x \in S^k$ can be computed exactly in polynomial time in size(v) and size(x). Even though we have defined the problem with the ℓ_∞-norm, we could also have used the ℓ_2- or ℓ_1-norm, since all these versions are computationally equivalent.

Our main result is Theorem 4.2. Membership in PPAD, which turns out to be the most challenging part of this result, is presented in Section 4.2 (using the multiple-source End-of-Line results of Chapter 3). PPAD-hardness is presented in Section 4.3 together with the FIXP-hardness result for the exact version.

Theorem 4.2. *For all even $k \geq 2$, kD-Hairy-Ball is PPAD-complete.*

4.2 The Hairy-Ball Problem is in PPAD

In this section we present our main result: the problem of computing an approximate Hairy Ball solution reduces to End-of-Line, the canonical PPAD-complete problem.

From a purely mathematical standpoint, our proof can be used to provide a (fairly cumbersome) proof of the Hairy Ball theorem by using Brouwer's fixed point theorem. Indeed, it is known [Pap94] that End-of-Line reduces to Brouwer (in fact, even 2D-Brouwer [CD09]). Thus, given a Hairy Ball function f, using our reduction and Brouwer's fixed point theorem, one can prove the existence of a point x_k such that $\|f(x_k)\| \leq 1/2^k$ for any k (using the fact that f must be uniformly continuous since the sphere is compact). Then, since any sequence in a compact set must have a converging subsequence it follows that there must exist x such that $f(x) = 0$. Finding a more direct way to deduce the Hairy Ball theorem from Brouwer's fixed point theorem is an interesting open question.

4.2.1 A general version of the Hairy-Ball problem

Our goal is to prove that the Hairy Ball problem lies in PPAD in a setting that is as general and encompassing as possible. The way the function is represented, as a circuit or otherwise, should not play a role. Thus, we are only going to make two assumptions about the tangent vector field: that it can be evaluated in polynomial time and that it is polynomially continuous in the sense defined by Etessami and Yannakakis [EY10] (see Section 2.2.3). The first assumption is very natural: if we are given a Hairy Ball function, we expect to be able to evaluate it efficiently. The motivation for the second assumption is that if we omit it, then there is no guarantee that there will exist an approximate solution with representation size that is polynomial in the input size.

Formally, let $\mathcal{F}$ be a polynomially computable and polynomially continuous class of Hairy Ball functions $f : S^k \to \mathbb{R}^{k+1}$ (i.e., continuous tangent vector fields) with $k \geq 2$ even. Note that here k is not fixed for all $f \in \mathcal{F}$, but we assume that $k \leq \text{size}(f)$. Note that this setting also captures kD-Hairy-Ball, where k is fixed and $\mathcal{F}$ is the class of all such functions represented using linear arithmetic circuits (with the projection onto the tangent space at the end). By Lemmas 2.1 and 2.2 this class is indeed polynomially computable and polynomially continuous.

For any class $\mathcal{F}$ of Hairy Ball functions we define the following computational problem.

Definition 4.2. **Hairy-Ball($\mathcal{F}$):**

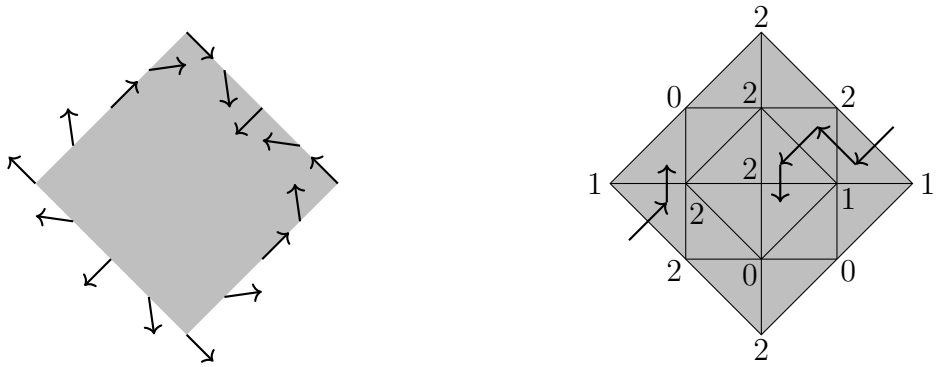
Input: $f \in \mathcal{F}$ and $\varepsilon > 0$.
Output: $x \in S^k$ such that $\|f(x)\|_\infty \leq \varepsilon$.

4.2.2 The general problem lies in PPAD

Theorem 4.3. *Let $\mathcal{F}$ be a class of Hairy Ball functions that is polynomially computable and polynomially continuous. Then,* Hairy-Ball($\mathcal{F}$) *lies in* PPAD.

Corollary 4.1. *For all even $k \geq 2$,* kD-Hairy-Ball *lies in* PPAD.

Proof overview for Theorem 4.3. The proof uses a Sperner argument to reduce Hairy-Ball($\mathcal{F}$) to 2-source End-of-Line, which lies in PPAD by Theorem 3.1, presented in Chapter 3. The inspiration for this reduction comes from a proof of the 2-dimensional Hairy Ball Theorem via a generalisation of Sperner's Lemma, given by Jarvis and Tanton [JT04]. Our contribution here is two-fold: we extend their proof to any higher (even) dimension and we turn it into a polynomial-time reduction. Even though Sperner's Lemma is known to be PPAD-complete [Pap94], this result does not apply to the general version of Sperner's Lemma that is used here. Indeed, by using standard techniques, this general version can only be reduced to a *2-source* instance of End-of-Line. This is why we then require our technical results about multiple-source End-of-Line from Chapter 3.



(a) Boundary conditions after "unfolding" ... **(b)** ... yielding a Sperner instance with two sources

Figure 4.1: An example for the proof of Theorem 4.3 in the case $k = 2$. The region C is represented in grey.

In order to turn the ideas of Jarvis and Tanton [JT04] into a polynomial-time reduction, instead of working directly on the sphere, we use a stereographic projection to "unfold" the sphere S^k ($\subset \mathbb{R}^{k+1}$) into the space $\mathbb{R}^k$, along with the vector field. Then, we consider a sufficiently large cross-polytope C of $\mathbb{R}^k$ and prove that the "unfolded" vector field satisfies certain boundary conditions. In the case $k = 2$, this corresponds to the vector field making two full rotations when we move along the boundary of C (see Figure 4.1a). Next, we pick an efficient triangulation of C and a suitable colouring of its nodes. The last step then requires us to prove that this colouring yields exactly two starting points (on the boundary) for Sperner paths (see

Figure 4.1b) that lead to *panchromatic* simplices. Using standard Sperner-arguments this yields a 2-source End-of-Line instance. Note that, as expected, the proof does not work if k is odd. Indeed, the construction then yields a starting point and an ending point on the boundary, instead of two starting points.

The full proof is presented in the next section.

4.2.3 Proof of Theorem 4.3

Let (f, ε) be an instance of Hairy-Ball($\mathcal{F}$) with $f : S^k \to \mathbb{R}^{k+1}$.

Stereographic projection. Consider the unit sphere S^k in $\mathbb{R}^{k+1}$. For convenience, we let the coordinate index start at 0 in $\mathbb{R}^{k+1}$, i.e., $x = (x_0, x_1, \ldots, x_k)$. Let $p = (1, 0, \ldots, 0) \in S^k$. The stereographic projection with respect to the pole p is defined as

$$SP: \quad S^k \setminus \{p\} \to \mathbb{R}^k, \quad (x_0, x_1, \ldots, x_k) \mapsto \frac{1}{1 - x_0}(x_1, \ldots, x_k)$$

and its inverse is

$$SP^{-1}: \quad \mathbb{R}^k \to S^k \setminus \{p\}, \quad (z_1, \ldots, z_k) \mapsto p + \frac{2}{1 + \sum_{i=1}^k z_i^2}(-1, z_1, \ldots, z_k).$$

Note that the stereographic projection and its inverse can be computed exactly in polynomial time in the representation size of the rational input point. In particular, rational points are always mapped to rational points. Furthermore, it is easy to check that the inverse stereographic projection SP^{-1} is $4\sqrt{k}$-Lipschitz continuous (w.r.t. ℓ_∞-norm).

Unfolding: changing the domain and range of f. We want to "transform" the function $f : S^k \to \mathbb{R}^{k+1}$ into a function $g : \mathbb{R}^k \to \mathbb{R}^k$, which is more convenient. Changing the domain of f is easy: the stereographic projection "unfolds" $S^k \setminus \{p\}$ into $\mathbb{R}^k$. To change the range of f we would like to also "unfold" the tangent vector field so that it now outputs a vector in $\mathbb{R}^k$ instead of $\mathbb{R}^{k+1}$. One way to achieve this is to find continuous vector fields $b_i : S^k \setminus \{p\} \to \mathbb{R}^{k+1}$, $i = 1, \ldots, k$, such that for every $x \in S^k \setminus \{p\}$, $b_1(x), \ldots, b_k(x)$ is a basis of the tangent space of S^k at x. Expressing $f(x)$ in this local basis then yields an element in $\mathbb{R}^k$, as desired. We can explicitly construct such vector fields by using SP and SP^{-1} to map the standard basis of $\mathbb{R}^k$ into the tangent space at each $x \in S^k \setminus \{p\}$. We obtain

$$[b_i(x)]_j = \begin{cases} x_i & \text{if } j = 0 \\ 1 - x_i^2/(1 - x_0) & \text{if } j = i \\ -x_j x_i/(1 - x_0) & \text{otherwise} \end{cases} \tag{4.1}$$

where $[\cdot]_j$ indicates the jth coordinate for $j \in \{0, 1, \ldots, k\}$. It is straightforward to check that $b_1, \ldots, b_k$ are continuous tangent vector fields that yield an orthonormal basis of the tangent space of S^k at every $x \in S^k \setminus \{p\}$.

"Unfolding" f yields $g : \mathbb{R}^k \to \mathbb{R}^k$ which is defined as follows for $i = 1, \ldots, k$

$$[g(z)]_i = \langle f(x(z)), b_i(x(z)) \rangle$$

where we define $x(z) := SP^{-1}(z)$ for convenience. Intuitively, g corresponds to the function that first maps $z \in \mathbb{R}^k$ to a point x on the sphere using the inverse stereographic projection, computes $f(x)$ and then expresses $f(x)$ in the local basis $(b_1(x), \ldots, b_k(x))$. Note that $g(z)$ can be computed in polynomial time in size(f) and size(z).

Since the b_i always form an orthonormal basis, it follows that we always have $\|g(z)\|_2 = \|f(SP^{-1}(z))\|_2$. Thus, in order to find some $x \in S^k$ with $\|f(x)\|_\infty \leq \varepsilon$, it suffices to find some $z \in \mathbb{R}^k$ with $\|g(z)\|_\infty \leq \varepsilon/\sqrt{k}$.

Continuity of g. Clearly, g is continuous. Moreover, since $\mathcal{F}$ is polynomially continuous, we can extend this to g in the following sense.

Claim 4.1. *There exists a polynomial r (that only depends on $\mathcal{F}$) such that for any $\widehat{\varepsilon} > 0$, there exists $\widehat{\delta}$ with* size$(\widehat{\delta}) \leq r($size$(f) +$ size$(\widehat{\varepsilon}))$ *such that for any $z, z' \in \mathbb{R}^k$ we have $\|z - z'\|_\infty \leq \widehat{\delta} \implies \|g(z) - g(z')\|_\infty \leq \widehat{\varepsilon}$.*

Proof. First, let us prove a bound on $\|f(x)\|_\infty$ for all $x \in S^k$. Let $\bar{\delta} > 0$ be such that $\|f(x) - f(y)\|_\infty \leq 1$ if $\|x - y\|_\infty \leq \bar{\delta}$. Recall that size$(\bar{\delta})$ is polynomially bounded in size(f). Since f is continuous on the sphere S^k, there exists some $x^* \in S^k$ such that $f(x^*) = 0$. The arc distance between x and x^* is at most π and we pick points $y^{(0)}, y^{(1)}, \ldots, y^{(m)} \in S^k$ along the arc such that $y^{(0)} = x, y^{(m)} = x^*$ and $\|y^{(i)} - y^{(i+1)}\|_\infty \leq \|y^{(i)} - y^{(i+1)}\|_2 \leq \bar{\delta}$. We can take $m \leq \lceil \pi/\bar{\delta} \rceil \leq 4/\bar{\delta}$ (assuming $\bar{\delta} \leq 1/2$). Then, we have

$$\|f(x)\|_\infty = \|f(x) - f(x^*)\|_\infty \leq \sum_{i=0}^{m-1} \|f(y^{(i)}) - f(y^{(i+1)})\|_\infty \leq \frac{4}{\bar{\delta}}.$$

For any $z, z' \in \mathbb{R}^k$ we have for all i

$$\begin{aligned}
|[g(z)]_i - [g(z')]_i| &= |\langle f(x(z)), b_i(x(z)) \rangle - \langle f(x(z')), b_i(x(z')) \rangle| \\
&\leq |\langle f(x(z)) - f(x(z')), b_i(x(z)) \rangle| + |\langle f(x(z')), b_i(x(z)) - b_i(x(z')) \rangle| \\
&\leq \|f(x(z)) - f(x(z'))\|_2 + \|f(x(z'))\|_2 \|b_i(x(z)) - b_i(x(z'))\|_2
\end{aligned}$$

Note that for all i, $z \mapsto b_i(x(z))$ is Lipschitz-continuous with a Lipschitz constant of the form $m\sqrt{k}$ for some constant m ($m = 20$ is enough). This can be checked by direct computation. Recalling that $x(z)$ is short for $SP^{-1}(z)$, the fact that SP^{-1} is $4\sqrt{k}$-Lipschitz and f is polynomially continuous, the claim then follows. □

The colouring. The function g induces a colouring on $\mathbb{R}^k$. Every $z \in \mathbb{R}^k$ is assigned a colour as follows. First, compute $u := g(z) \in \mathbb{R}^k$. If $u_i \geq 0$ for all i, then assign colour 0. Otherwise assign colour $j = \operatorname{argmin}_i u_i$ (break ties by picking the smallest such index).

Pick $\delta > 0$ so that $\|z - z'\|_\infty \leq \delta$ implies

$$\|g(z) - g(z')\|_\infty \leq \frac{\varepsilon}{8\sqrt{k}} \tag{4.2}$$

By Claim 4.1, it is possible to pick such δ with $\text{size}(\delta) \leq \text{poly}(\text{size}(f), \text{size}(\varepsilon))$.

A panchromatic δ-fine k-simplex in $\mathbb{R}^k$ is a k-simplex $z^{(0)}, \ldots, z^{(k)}$ in $\mathbb{R}^k$, such that $z^{(i)}$ has colour i and $\|z^{(i)} - z^{(j)}\|_\infty \leq \delta$ for all i, j.

Claim 4.2. *Any panchromatic k-simplex in $\mathbb{R}^k$ yields a solution.*

Proof. Let the δ-fine k-simplex $z^{(0)}, \ldots, z^{(k)}$ be panchromatic. Since $[g(z^{(0)})]_j \geq 0$ for all j, there exists i such that $[g(z^{(0)})]_i = \|g(z^{(0)})\|_\infty$. However, it must hold that $[g(z^{(i)})]_i < 0$ and thus $\|g(z^{(0)})\|_\infty \leq |[g(z^{(0)})]_i - [g(z^{(i)})]_i|$. Since $\|z^{(0)} - z^{(i)}\|_\infty \leq \delta$, it follows by (4.2) that $\|g(z^{(0)})\|_\infty \leq \varepsilon/8\sqrt{k} \leq \varepsilon/\sqrt{k}$, i.e., $z^{(0)}$ yields a solution. □

Sperner. The next step is to show that we can locate a panchromatic k-simplex by using a Sperner-like argument. We explain the intuition in the case $k = 2$. If $\|f(p)\|_\infty \leq \varepsilon$, then we have found a solution. If this is not the case, then we consider a sufficiently large region C in $\mathbb{R}^k$ centred at 0. We can show that on the boundary of C, g (approximately) behaves as in Figure 4.1a, which yields a colouring similar to Figure 4.1b. The colouring on the boundary of the domain in Figure 4.1b has a very special structure. First of all, it is antipodally symmetric, namely, points lying on opposite sides have the same colour. Furthermore, if we start from any point on the boundary and "move" along the boundary in anti-clockwise direction, then we will see colour 1 change into colour 2 exactly twice. However, we will never see colour 2 change into colour 1; there will always be colour 0 in-between.

Intuitively, we want to pick a δ-fine triangulation of the ball and construct a directed graph on the triangles. There is a directed edge between two triangles if they share a facet coloured $1, 2$ and the edge is directed such that it crosses the facet with the colour 1 on its left-hand side and the colour 2 is on its right-hand side. Then, because of the structure of the colouring on the boundary, we obtain two known sources and any sink or other source of this directed graph corresponds to a panchromatic simplex, i.e., a solution.

This intuition is formalised in the following lemma, which is proved below.

Lemma 4.1. *The problem of finding a δ-fine panchromatic k-simplex in $\mathbb{R}^k$ reduces to 2-source* End-of-Line.

By Theorem 3.1 in Chapter 3 it then follows that the problem lies in PPAD, which concludes the proof of Theorem 4.3.

4.2.4 Proof of Lemma 4.1

If $\|f(p)\|_\infty \leq \varepsilon$, then we have found a solution. Thus, assume that this is not the case. To simplify the analysis we assume that the standard coordinate system is such that $f(p) = (0, v, v, \ldots, v)/\sqrt{k}$, $v > \varepsilon$. If this is not the case then we perform a rotation such that this holds (at least approximately; a small error can be tolerated). Let $e^{(i)}$ be the ith unit vector in $\mathbb{R}^k$.

In order to investigate the colouring, we first prove some properties of an *ideal* colouring that we define below and then show that the *actual* colouring is "close" enough to the ideal colouring and also satisfies these properties.

The ideal colouring. The ideal colouring is a function $w : \mathbb{R}^k \setminus \{0\} \to \mathbb{R}^k$. For $z \in S^{k-1}$ it is given by $w_i(z) = (1 - 2z_i \sum_{j=1}^k z_j)/\sqrt{k}$ for $1 \leq i \leq k$. For $z \in \mathbb{R}^k \setminus \{0\}$, we set $w(z) := w(z/\|z\|_2)$. The ideal colour of z is the colour corresponding to $w(z)$ (using the same procedure as for the actual colouring), instead of $g(z)$.

Claim 4.3. *The following properties of w are easy to check by direct computation.*

1. *w is antipodally symmetric, i.e., $w(-z) = w(z)$ for all $z \in \mathbb{R}^k \setminus \{0\}$.*
2. *For all $z \in \mathbb{R}^k \setminus \{0\}$, $\|w(z)\|_2 = 1$.*
3. *If $w(z) \leq 0$, then $z \geq 0$ or $z \leq 0$.*
4. *For any subset of the indices $\emptyset \neq I \subseteq \{1, \ldots, k\}$, if $z \in S^{k-1}$ is such that $z_i \geq 0$ for $i \in I$ and $z_i = 0$ otherwise, then $w_j(z) = 0$ for all $j \notin I$ and there exists $i \in I$ such that $w_i(z) \leq -1/\sqrt{k}$. In particular, any colour-direction $u \in \mathbb{R}^k$ with $\|u - w(z)\|_\infty < 1/2\sqrt{k}$ yields a colour in I.*

The ideal colouring is "close" to the actual colouring. We now show that if $\|z\|_2$ is sufficiently large, then the ideal colouring is very close to the actual colouring. Namely, we will show that $w(z)$ is almost equal to $\widehat{g}(z)$, a "normalisation" of g defined by $\widehat{g}(z) = c(z) \cdot g(z)$, where $c(z) = \frac{1+\|z\|_2^2}{v\|z\|_2^2}$. Note that $\widehat{g}$ and g yield the exact same colour for any $z \neq 0$.

Claim 4.4. *There exists $t > 0$ with $\mathrm{size}(t) = \mathrm{poly}(\mathrm{size}(f), \mathrm{size}(\varepsilon))$ such that for all $z \in \mathbb{R}^k$ with $\|z\|_2 \geq t$*

$$\|\widehat{g}(z) - w(z)\|_\infty \leq \frac{1}{4\sqrt{k}}.$$

Proof. Since f is polynomially continuous, there exists $\widehat{\delta}$ such that $\|f(x) - f(p)\|_\infty \leq \varepsilon/16\sqrt{k(k+1)}$ with $\text{size}(\widehat{\delta}) = \text{poly}(\text{size}(f), \text{size}(\varepsilon))$. Using the definition of SP^{-1} it is easy to check that $\|x(z) - p\|_2 \leq 2/\sqrt{1 + \|z\|_2^2}$. Thus, we can construct $t \geq 4$ with $\text{size}(t) = \text{poly}(\text{size}(f), \text{size}(\varepsilon))$ such that $\|f(x(z)) - f(p)\|_\infty \leq \varepsilon/16\sqrt{k(k+1)}$ for all z with $\|z\|_2 \geq t$.

Let $\bar{g}(z)$ denote the colour-direction obtained at z if we use $f(p)$, instead of $f(x(z))$, i.e., $\bar{g}_i(z) = \langle f(p), b_i(x(z))\rangle$. Then, for all z with $\|z\|_2 \geq t$, we have

$$\|g(z) - \bar{g}(z)\|_\infty \leq \sqrt{k+1}\|f(x(z)) - f(p)\|_\infty \leq \varepsilon/16\sqrt{k}. \tag{4.3}$$

Recalling that $f(p) = (0, v, v, \ldots, v)/\sqrt{k}$ and using equation (4.1) and the definition of SP^{-1}, we can write for all i

$$\begin{aligned}
\bar{g}_i(z) = \langle f(p), b_i(x(z))\rangle &= \frac{v}{\sqrt{k}} \sum_{j=1}^{k} [b_i(x(z))]_j \\
&= \frac{v}{\sqrt{k}} \left(1 - \frac{x_i(z)}{1 - x_0(z)} \sum_{j=1}^{k} x_j(z)\right) \\
&= \frac{v}{\sqrt{k}} \left(1 - \frac{\frac{2z_i}{1+\|z\|_2^2}}{\frac{2}{1+\|z\|_2^2}} \sum_{j=1}^{k} \frac{2z_j}{1 + \|z\|_2^2}\right) \\
&= \frac{v}{\sqrt{k}} \left(1 - \frac{2z_i}{1 + \|z\|_2^2} \sum_{j=1}^{k} z_j\right) \\
&= \frac{v}{\sqrt{k}} \cdot \frac{\|z\|_2^2}{1 + \|z\|_2^2} \left(\frac{1 + \|z\|_2^2}{\|z\|_2^2} - \frac{2z_i}{\|z\|_2^2} \sum_{j=1}^{k} z_j\right) \\
&= \frac{v\|z\|_2^2}{1 + \|z\|_2^2} \left(\frac{1 - 2\frac{z_i}{\|z\|_2} \sum_{j=1}^{k} \frac{z_j}{\|z\|_2}}{\sqrt{k}} + \frac{1}{\sqrt{k}\|z\|_2^2}\right) \\
&= \frac{1}{c(z)} \left(w_i(z) + \frac{1}{\sqrt{k}\|z\|_2^2}\right).
\end{aligned}$$

We obtain that for any z with $\|z\|_2 \geq t$

$$\begin{aligned}
\|\widehat{g}(z) - w(z)\|_\infty \leq c(z)\|g(z) - \bar{g}(z)\|_\infty + \|c(z) \cdot \bar{g}(z) - w(z)\|_\infty &\leq \frac{c(z)\varepsilon}{16\sqrt{k}} + \frac{1}{\sqrt{k}\|z\|_2^2} \\
&\leq \frac{1 + \|z\|_2^2}{16\sqrt{k}\|z\|_2^2} + \frac{1}{\sqrt{k}t^2} \\
&\leq 1/4\sqrt{k}
\end{aligned}$$

where we used (4.3), $v > \varepsilon$ and $t \geq 4$. □

The cross-polytope C_m. Instead of working on a ball in $\mathbb{R}^k$, we define our Sperner instance on a different region that is easier to triangulate efficiently while still behaving

nicely with respect to the colouring. Let $m = \lceil t\sqrt{k} \rceil$. The *cross-polytope* of radius m is defined as $C_m = \{z \in \mathbb{R}^k : \|z\|_1 \leq m\}$.

The unit cross-polytope can be triangulated efficiently by using a standard efficient triangulation of the k-simplex $0, e^{(1)}, \ldots, e^{(k)}$ (see, e.g., [Tod76]) and then extending it to the rest of the cross-polytope by mirroring along each axis. Thus, using a δ/m-fine efficient triangulation of the simplex, extending it to the cross-polytope and then scaling to C_m, yields an efficient δ-fine triangulation of C_m. We call this triangulation T. In particular, given a simplex of T and one of its facets, we can compute the other simplex of T that shares this facet in polynomial time (or decide that it does not exist, if the facet lies on the boundary of C_m). Note that size(δ/m) is polynomial in the size of the input.

The boundary of C_m. The boundary of C_m is denoted $\partial C_m = \{z \in \mathbb{R}^k : \|z\|_1 = m\}$. Note that for any $z \in \partial C_m$ we have $\|z\|_2 \geq t$. Let $\partial C_m^+ := \{z \geq 0 : \|z\|_1 = m\}$ and $\partial C_m^- := \{z \leq 0 : \|z\|_1 = m\}$ denote the intersection of ∂C_m with the all-positive and all-negative orthant respectively. Note that ∂C_m^+ and ∂C_m^- are the $(k-1)$-simplices $m \cdot e^{(1)}, \ldots, m \cdot e^{(k)}$ and $-m \cdot e^{(1)}, \ldots, -m \cdot e^{(k)}$. Furthermore, by construction, T induces a triangulation on ∂C_m^+ and ∂C_m^-. We now prove two key properties of the actual colouring on ∂C_m.

Claim 4.5. *∂C_m^+ satisfies the standard Sperner boundary conditions. Namely, any face $m \cdot e^{(i_1)}, \ldots, m \cdot e^{(i_\ell)}$ $(1 \leq i_1 < \cdots < i_\ell \leq k,\ 1 \leq \ell \leq k)$ of ∂C_m^+ only contains colours in $\{i_1, \ldots, i_\ell\}$ in the actual colouring. The same also holds for ∂C_m^-.*

Proof. By Claim 4.3 (Item 4), it suffices to show that $\|\widehat{g}(z) - w(z)\|_\infty < 1/2\sqrt{k}$ for all z on the face $m \cdot e^{(i_1)}, \ldots, m \cdot e^{(i_\ell)}$, which holds by Claim 4.4 and the choice of m. □

Claim 4.6. *The restriction of the triangulation T to $\partial C_m \setminus (\partial C_m^+ \cup \partial C_m^-)$ does not contain any $(k-1)$-simplex coloured $1, 2, \ldots, k$ (in the actual colouring).*

Proof. Let $y^{(1)}, \ldots, y^{(k)}$ be a δ-fine $(k-1)$-simplex on ∂C_m such that $y^{(i)}$ has colour i. By Claim 4.3 (Item 3) it suffices to show that $w(y^{(i)}) \leq 0$ for all i. Note that since $\|y^{(i)} - y^{(j)}\|_\infty \leq \delta$, it follows by (4.2)

$$\|\widehat{g}(y^{(i)}) - \widehat{g}(y^{(j)})\|_\infty \leq \frac{1}{8\sqrt{k}} \frac{\varepsilon}{v} \frac{1 + \|z\|_2^2}{\|z\|_2^2} \leq 1/4\sqrt{k} \tag{4.4}$$

since $v > \varepsilon$ and $\|z\|_2 \geq t \geq 1$.

Now assume that there exists ℓ such that $w_\ell(y^{(i)}) > 0$. (4.4) and Claim 4.4 imply that $\widehat{g}_\ell(y^{(\ell)}) > -1/2\sqrt{k}$ and thus also $\widehat{g}_j(y^{(\ell)}) > -1/2\sqrt{k}$ for all j, since $y^{(\ell)}$ has colour ℓ in the actual colouring. On the other hand, since $\widehat{g}_j(y^{(j)}) < 0$ it follows that $\widehat{g}_j(y^{(\ell)}) < 1/4\sqrt{k}$ using (4.4). Thus, we get $\|\widehat{g}(y^{(\ell)})\|_\infty < 1/2\sqrt{k}$. Using Claim 4.4 again, it follows that $\|w(y^{(\ell)})\|_\infty < 1/\sqrt{k}$, which implies $\|w(y^{(\ell)})\|_2 < 1$, a contradiction to Claim 4.3. □

Orientation of ∂C_m^+ and ∂C_m^-. The following simple observation is crucial.

Claim 4.7. *The $(k-1)$-simplices ∂C_m^+ and ∂C_m^- have the same orientation with respect to the Sperner boundary conditions.*

Proof. Consider the linear function $T : \mathbb{R}^k \to \mathbb{R}^k$, $T(y) = -y$ and note that T maps ∂C_m^+ to ∂C_m^- while preserving the Sperner boundary conditions (Claim 4.5). Furthermore, the determinant of T is $(-1)^k = 1$, since k is even. It follows that ∂C_m^+ and ∂C_m^- have the same orientation with respect to the Sperner boundary conditions. □

This is the only point in the proof where we use the fact that k is even. However, it is indeed crucial, since this ensures that the two ends of line that we know are both sources (or both sinks), as we will see below. If k is odd, our reduction yields an instance with a known source and sink. In this case, there is no guarantee that a solution – another end of line – even exists.

2-source End-of-Line. We now explain how the problem of finding a panchromatic k-simplex in the triangulation T of C_m reduces to a 2-source End-of-Line problem. Let us briefly recall how a standard k-dimensional Sperner instance reduces to End-of-Line. A fully detailed and formal presentation of this is given in [EY10, Proposition 2.2]. In the standard k-dimensional Sperner problem, we have a big k-simplex triangulated into k-simplices and a colouring that satisfies the Sperner boundary conditions. The End-of-Line graph is constructed as follows. Every k-simplex of the triangulation is a node in the graph. There is an edge between two nodes if the corresponding k-simplices have a common facet coloured $1, \ldots, k$ (called a *door-facet*). The edge is directed by considering the orientation of the k-simplices with respect to the common facet. This yields an End-of-Line graph where every degree-1 node is either a panchromatic k-simplex or a k-simplex that has a door-facet lying on the boundary of the instance, i.e., on the boundary of the big k-simplex. Note that all the door-facets lying on the boundary of the instance lie on the $1, \ldots, k$-facet of the big k-simplex. In order to obtain a single source, there are various standard tricks, see, e.g., [Tod76, Chapter 4]. If we use the so-called "artificial start" trick, then as shown in [Tod76; EY10], we obtain a slightly larger Sperner instance that has a single door-facet "exposed" to the outside, i.e., lying on the boundary of the instance.

We can use these standard techniques on our problem. The boundary of our instance is ∂C_m. By Claim 4.6, all door-facets on the boundary are contained in ∂C_m^+ or ∂C_m^-. By applying the trick mentioned above on ∂C_m^+ (interpreted as a facet of a k-dimensional Sperner instance) we obtain a single door-facet that is "exposed" in this whole region. Applying the exact same trick on ∂C_m^- also yields a single door-facet that is "exposed" in this whole region. Thus, overall we only have

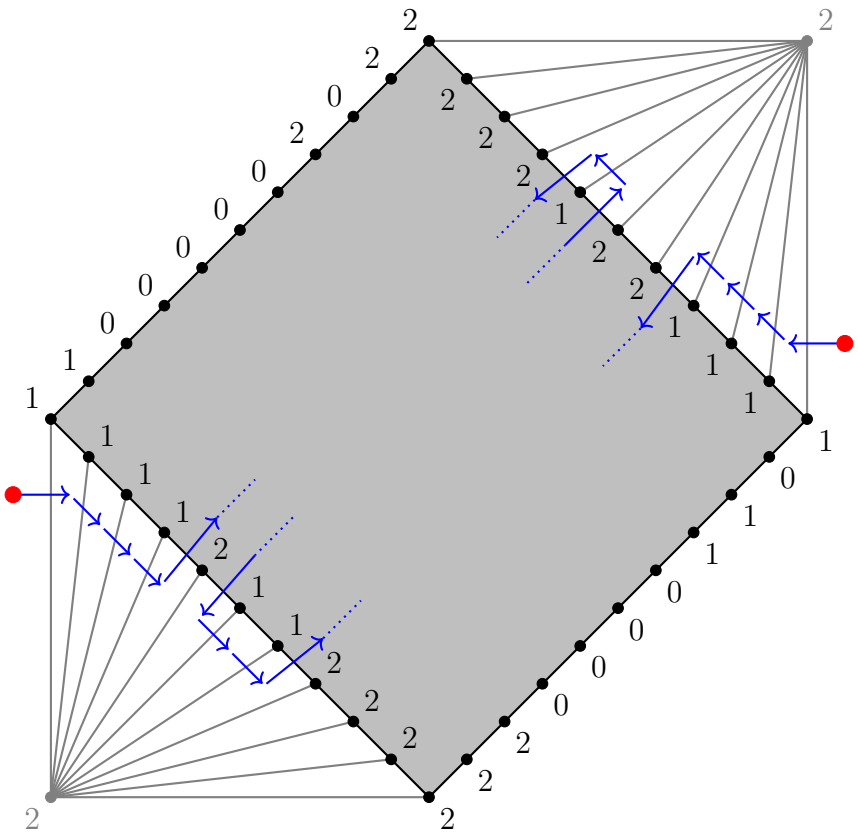


Figure 4.2: Example in 2 dimensions showing how the "artificial start" trick is used to obtain a 2-source End-of-Line instance. The cross-polytope C_m is shown in grey and the vertices of the triangulation of C_m that lie on ∂C_m are also shown together with their colours. The "artificial start" trick has introduced a new vertex with colour 2 that is connected to all the triangulation vertices on ∂C_m^+. The same has also been done for ∂C_m^-. The two new vertices and the new edges are shown in light grey colour. All the directed edges of the resulting End-of-Line instance that cross the boundary of C_m or lie outside C_m are shown in blue in this example. The two red points represent the two sources we obtain for the End-of-Line instance. Note that in this example the colouring is not perfectly antipodally symmetric, since this is not the case in general. However, it is easy to see that the displayed colouring satisfies Claims 4.5 to 4.7.

two door-facets that lie on the boundary of our instance. Figure 4.2 shows how the "artificial start" trick is used on a 2-dimensional example.

Finally, the important observation here is that the orientation of the "exposed" door-facet obtained by using the trick only depends on the orientation of the big facet on which it is used. Since ∂C_m^+ and ∂C_m^- have the same orientation (Claim 4.7), the two "exposed" door-facets have the same orientation. This means that we have two "entrances" for the standard Sperner path-following algorithm. By applying the formal arguments used in the reduction from Sperner to End-of-Line this yields a two-source End-of-Line instance.

4.3 PPAD- and FIXP-hardness for the Hairy Ball Problem

It is possible to prove Brouwer's fixed point theorem using the Hairy Ball Theorem as follows (see [Mil78] for the full details). Let $B^k \subset \mathbb{R}^k$ be the unit ball. If we assume that a function $f : B^k \to B^k$ does not have any fixed point, then we can construct a Hairy Ball function $g : S^k \to \mathbb{R}^{k+1}$ that does not have a zero. Brouwer's theorem follows by contradiction. The main idea for the construction of g is the following. Consider $f'(x) = f(x) - x$ and assume that it points directly inward on the boundary of B^k. Take one copy of B^k with the vector field f' and one copy with the vector field $-f'$, and glue their boundaries together. The resulting object can be deformed to yield the sphere S^k and the vector fields will perfectly match on the glued region. Thus, assuming that f' has no zero, g will have no zero either.

By making this idea fully constructive and efficient, we obtain reductions from Brouwer problems to Hairy Ball problems. Thus, existing PPAD- and FIXP-hardness results for Brouwer also hold for the corresponding Hairy Ball problems. We note that these reductions always involve using *two* copies of a Brouwer instance to obtain a single Hairy Ball instance. This further supports our claim that the fact that we obtain two sources when reducing kD-Hairy-Ball to End-of-Line (Section 4.2) is not an artefact of our reduction, but intrinsic to the problem.

4.3.1 FIXP-hardness

The tools introduced by Etessami and Yannakakis [EY10] allow us to study the complexity of finding an *exact* Hairy Ball solution, as well as other versions such as the strong approximation problem: find a solution that is close to an exact solution. See Section 2.4 for more details on this as well as a formal definition of the class FIXP.

In this section, we prove that these problems for the Hairy Ball Theorem are at least as hard as their Brouwer counterparts. We define our exact computation problem for the Hairy Ball Theorem as follows:

Definition 4.3. **Exact-Hairy-Ball**:

Input: An arithmetic circuit F (with gates $\{+, -, \times, /, \max, \min\}$, rational constants and that never divides by 0) that computes a tangent vector field $S^k \to \mathbb{R}^{k+1}$, k even.
Output: $x \in S^k$ such that $F(x) = 0$.

Note that the vector field will be continuous since the circuit never divides by 0. The condition that the circuit never divides by 0 is enforced as a promise on the input. The vector field can be syntactically forced to be tangent to the sphere because we can compute the projection exactly using this kind of circuit.

Theorem 4.4. Exact-Hairy-Ball *is* FIXP*-hard. Furthermore, the corresponding strong approximation, partial computation and decision problems are also hard for the corresponding versions of* FIXP *(as defined in [EY10]).*

Following Etessami and Yannakakis [EY10], we could define a class HB that captures the complexity of computing an exact Hairy Ball solution (and corresponding versions of the class for the related problems) by taking the set of all problems that reduce to Exact-Hairy-Ball. Then, Theorem 4.4 is saying that FIXP $\subseteq$ HB. Note that the three discrete problems that can be studied in the Turing machine model lie in PSPACE, by using the same technique as in [EY10, Proposition 4.2].

Proof of Theorem 4.4. The proof is based on an idea used by Milnor [Mil78] to show that the Hairy Ball theorem implies Brouwer's fixed point theorem. Note, however, that Milnor's proof proceeds by contradiction, whereas our proof is fully constructive. We embed two copies of a Brouwer instance (after some preprocessing) on the sphere such that any exact Hairy Ball solution yields an exact Brouwer fixed point.

Let G be an arithmetic circuit with gates $\{+, -, \times, /, \max, \min\}$ (and rational constants) that maps the unit cube $B_\infty^k = \{x \in \mathbb{R}^k : \|x\|_\infty \leq 1\}$ into itself and does not divide by 0. Computing a fixed point of G is a FIXP-complete problem and we will reduce this to Exact-Hairy-Ball. At the end, we also explain why our reduction is an SL-reduction (see Section 2.4 for a definition).

Step 1: Preprocessing. First of all, note that we can assume that k is even. Indeed, we can always consider $\bar{G} : B_\infty^{k+1} \to B_\infty^{k+1}$, $\bar{G}(x_0, x) = (0, G(x))$. Note that fixed points are mapped one-to-one by dropping the first coordinate. Let ℓ be large enough such that $2^\ell \geq 4k$. For any integer m let $mB^k = \{x \in \mathbb{R}^k : \|x\|_2 \leq m\}$. We construct the circuit G' that on input $x \in 2^\ell B^k$ does:

1. Compute y by letting $y_i = \max\{-1, \min\{1, x_i\}\}$ for all $i \in [k]$.
2. Output $-x \cdot \min\{1, \|x - y\|_\infty\} + (G(y) - x) \cdot \max\{0, 1 - \|x - y\|_\infty\}$

It is easy to check that the fixed points of G are exactly the zeros of G'. Furthermore, if $\|x\|_2 \geq 2^{\ell-1}$, then $G'(x) = -x$. Next, we change the domain from $2^\ell B^k$ to $1B^k =: B^k$. The circuit $\widehat{G}$ is constructed such that $\widehat{G}(x) = 2^{-\ell} G'(2^\ell x)$. The zeros of G' are exactly the zeros of $\widehat{G}$ scaled by 2^ℓ. Note that if $\|x\|_2 \geq 1/2$, we have $\widehat{G}(x) = -x$.

Step 2: From Brouwer to Hairy Ball. The high-level idea in order to construct a Hairy Ball vector field is to use the output of $\widehat{G}(x)$ as coordinates in some continuous tangent basis on S^k. However, as noted before, it is impossible to define a continuous tangent basis on all of S^k. As a result, we use two different bases (namely $\widehat{b}_i(x)$ and $-\widehat{b}_i(-x)$) and obtain two tangent vectors at x: w^- and w^+. The final output u is then

obtained by taking a convex combination of w^- and w^+, according to a parameter α. This parameter α ensures that we do not use a tangent basis when we are close to the pole where it is not a basis. Furthermore, it also ensures that the case where u is obtained as a *strict* convex combination (namely $\alpha \in (0,1)$) can only occur at points where $\widehat{G}(x) = -x$, which makes this case easier to analyse.

Formally, we construct the circuit $F : S^k \to \mathbb{R}^{k+1}$ that does the following on input $x = (x_0, x_1, \ldots, x_k)$

1. Compute $v = \widehat{G}(x_1, \ldots, x_k)$
2. Compute $w^- = \sum_{i=1}^k v_i \widehat{b}_i(x)$ and $w^+ = \sum_{i=1}^k (-v_i)\widehat{b}_i(-x)$
3. Compute $\alpha = \max\{0, \min\{1, 1/2 - x_0\}\}$
4. Output $u = \alpha w^- + (1-\alpha)w^+$

where

$$\left[\widehat{b}_i(x)\right]_j = \begin{cases} x_i(1-x_0) & \text{if } j = 0 \\ 1 - x_0 - x_i^2 & \text{if } j = i \\ -x_j x_i & \text{otherwise} \end{cases}$$

is the non-normalised version of the tangent basis used in the proof of Theorem 4.3. In particular, the $\widehat{b}_i(x)$ form an orthogonal basis of the tangent space at x (as long as $x_0 < 1$) and we have $\|\widehat{b}_i(x)\|_2 = 1 - x_0$. Note that the output u of the circuit is always tangent to the sphere at x.

If $x_0 \leq -1/2$, then $u = w^-$, and thus, using the fact that the $\widehat{b}_i(x)$ form an orthogonal basis, we obtain

$$\|u\|_2^2 = \sum_{i=1}^k v_i^2 \left\langle \widehat{b}_i(x), \widehat{b}_i(x) \right\rangle = \sum_{i=1}^k v_i^2 \|\widehat{b}_i(x)\|_2^2 = \|v\|_2^2 (1-x_0)^2.$$

Since $1 - x_0 \geq 1$, this implies that $\|u\|_2 \geq \|v\|_2$, i.e.,

$$\|F(x_0, x_1, \ldots, x_k)\|_2 \geq \|\widehat{G}(x_1, \ldots, x_k)\|_2.$$

This means that any zero of F yields a zero of $\widehat{G}$ by dropping the first coordinate.

If $x_0 \geq 1/2$, then $u = w^+$, and the analogous computation yields that $\|u\|_2^2 = \sum_{i=1}^k v_i^2 \|\widehat{b}_i(-x)\|_2^2 = \|v\|_2^2 (1+x_0)^2$. Since $1 + x_0 \geq 1$, we again obtain that $\|u\|_2 \geq \|v\|_2$, and the same conclusion follows.

The last remaining case is when $|x_0| < 1/2$. If we let $x' = (x_1, \ldots, x_k)$, then, since $\|x\|_2 = 1$, it must be that $\|x'\|_2 \geq 1/2$, which implies that $v = \widehat{G}(x') = -x'$. It follows

that

$$\begin{aligned} u_0 = \alpha w_0^- + (1-\alpha) w_0^+ &= \alpha \sum_{i=1}^{k} v_i \left[\widehat{b}_i(x)\right]_0 + (1-\alpha) \sum_{i=1}^{k} (-v_i) \left[\widehat{b}_i(-x)\right]_0 \\ &= \alpha \sum_{i=1}^{k} (-x_i) x_i (1-x_0) + (1-\alpha) \sum_{i=1}^{k} x_i (-x_i)(1+x_0) \\ &= \alpha \sum_{i=1}^{k} (-x_i^2)(1-x_0) + (1-\alpha) \sum_{i=1}^{k} (-x_i^2)(1+x_0) \\ &= -\|x'\|_2^2 (1 + x_0(1-2\alpha)) \leq -\|x'\|_2^2/2 \leq -1/8 \end{aligned}$$

which implies that F has no zero in this region.

The reduction we have provided is an SL-reduction, since a solution of the original instance can be obtained by applying a separable linear transformation (with rational coefficients that are computable in polynomial time from the original instance) to any solution of the final instance. Furthermore, the coefficients of the separable linear transformation are always powers of 2. It follows (see [EY10]) that the corresponding strong approximation, partial computation and decision problems are also hard for the corresponding FIXP classes. □

4.3.2 PPAD-hardness

Using the same approach we can also prove the following.

Theorem 4.5. *For all even $k \geq 2$, kD-Hairy-Ball is PPAD-hard.*

Proof. We essentially use the same construction as in the proof of Theorem 4.4. Consider any fixed even $k \geq 2$. It is known that computing an ε-approximate fixed point of a *linear* arithmetic circuit $G : B_\infty^k \to B_\infty^k$ is PPAD-complete [Meh14]. Starting from such a linear arithmetic circuit G, we perform the construction in the proof of Theorem 4.4 to obtain a Hairy Ball function $F : S^k \to \mathbb{R}^{k+1}$.

Now, it is easy to check that since G is a linear circuit, F will be a well-behaved arithmetic circuit (Section 2.2.2) without comparison gates. Furthermore, since G is a linear arithmetic circuit, it is L-Lipschitz-continuous for some L of polynomial representation size (Lemma 2.2). It is then easy to check that F will be cL-Lipschitz-continuous on S^k for some constant c that only depends on k. As a result, we can use Theorem 2.1 to approximate F by a linear arithmetic circuit $\widehat{F}$. Namely, we can construct $\widehat{F}$ such that $\|F(x) - \widehat{F}(x)\|_\infty \leq \delta$ for all $x \in S^k$, for some $\delta > 0$ to be determined later.

This circuit $\widehat{F}$ is our instance of kD-Hairy-Ball. Now, assume that we have some $x \in S^k$ that is a solution, i.e., such that $\|P_x[\widehat{F}(x)]\|_\infty \leq \varepsilon'$. Since P_x is a projection operator, it follows that $\|P_x[F(x) - \widehat{F}(x)]\|_\infty \leq \sqrt{k+1}\|F(x) - \widehat{F}(x)\|_\infty \leq \sqrt{k+1}\delta$.

Thus, $\|F(x)\|_\infty = \|P_x[F(x)]\|_\infty \leq \varepsilon' + \sqrt{k+1}\delta$, where we used the linearity of the projection operator. Inspection of the construction of F from G yields that $z \in B^k_\infty$, obtained from $x = (x_0, x_1, \ldots, x_k)$ by letting $z_i = \max\{-1, \min\{1, 2^\ell x_i\}\}$ for all $i \in [k]$, must then satisfy $\|G(x) - x\|_\infty \leq c'(\varepsilon' + \delta)$ where c' is some constant that only depends on k. By picking ε' and δ sufficiently small, we obtain that z is an ε-approximate fixed point of G. This means that we have provided a reduction from the problem of finding an approximate fixed point of a linear arithmetic circuit (in k dimensions) to the kD-Hairy-Ball problem. □

Note that the original proof of this result in [GH21] is a bit more tedious, since it goes through an intermediate discrete version of the problem that is termed the 2D-Hairy-Cube problem. The approximation technique given by Theorem 2.1 allows us to obtain a more direct proof here.

4.4 Conclusion and Future Directions

We have obtained a satisfying answer to the question of the computational complexity of the Hairy Ball Theorem, if we are looking for an approximate solution. For other solution concepts related to exact solutions, we have provided a FIXP-hardness result. This leaves open the question of whether the problem is FIXP-complete in this case. A first step in that direction would be to try to reduce Hairy Ball to Borsuk-Ulam, even though no such (fully constructive) mathematical proof seems to be known.

Our reduction from approximate Hairy Ball to approximate Brouwer cannot be used on the exact versions of these problems, because the step where we reduce from multi-source End-of-Line to standard End-of-Line corresponds to a *discontinuous* mapping between the corresponding topological structures. This raises the question of whether continuous mappings are an important subclass of reductions and motivates further study on this topic.

Chapter 5

The Generalised Circuit Problem

The *Generalised Circuit problem*, defined by Chen, Deng and Teng [CDT09], has been instrumental for proving many PPAD-hardness results, including for Nash equilibrium computation problems [DGP09; CDT09; Rub18], but also for problems arising in economics [CPY17; SSB17] or fair division [FFGZ18; GHI+20]. In this chapter, we show that it suffices to consider significantly restricted versions of the Generalised Circuit problem when proving PPAD-hardness results, and that an exact version of the problem can also be used to prove FIXP-hardness. These new simplified tools are then used to study equilibrium computation in first-price auctions in Chapter 6.

5.1 The GCircuit Problem with Two or Three Gates

Generalised circuits can be viewed as a generalisation of arithmetic circuits where we also allow *cycles*. This means that instead of representing a function, a generalised circuit represents a certain kind of constraint satisfaction problem. Indeed, the goal in the Generalised Circuit problem is to assign a value to each gate of the circuit such that all the gates are (approximately) satisfied. Importantly, gates are only allowed to take values in $[0, 1]$ and arithmetic operations are truncated accordingly. As a result, it can be shown that by Brouwer's fixed point theorem, there always exists an assignment of values that satisfies all the gates. However, computing even an approximate assignment is already PPAD-hard, i.e., essentially as hard as any Brouwer fixed point computation. We now provide some formal definitions.

Definition 5.1. A *generalised circuit*[1] with gate-types $\mathcal{G}$ is a list of gates $g_1, g_2, \ldots, g_n$. Every gate g_i is a 3-tuple $g_i = (G, j, k)$, where $G \in \mathcal{G}$ is the type of the gate, and $j, k \in [n] = \{1, \ldots, n\}$ are the indices of the input gates g_j, g_k (i, j, k distinct).

[1]Note that in the usual definition of generalised circuits, every gate also contains a rational parameter $\zeta \in [0, 1]$, which is used by some gate-types, e.g., a gate performing multiplication by the constant ζ. In our definition, gates do not contain this rational parameter, because, as we show in Theorems 5.1 and 5.2, these gate-types are actually not needed for the problems to be hard.

Before describing possible types of gates, we introduce some notation. For $t_1 < t_2$, we will let $\mathrm{T}_{[t_1,t_2]}$ denote the *truncation* of a value x to $[t_1, t_2]$, i.e., $\mathrm{T}_{[t_1,t_2]}(x) = \max\{t_1, \min\{t_2, x\}\}$. For convenience, we let $\mathrm{T} = \mathrm{T}_{[0,1]}$. Furthermore, we use the notation $x = y \pm \varepsilon$ to denote that $|x - y| \leq \varepsilon$.

Consider a generalised circuit $g_1, g_2, \ldots, g_n$ and an assignment $\mathbf{v} : [n] \to [0, 1]$ of values to its gates. We say that a gate is ε-satisfied by the assignment, if the constraint imposed by this gate is satisfied with error at most ε. The constraint that a gate $g_i = (G, j, k)$ must satisfy depends on its gate-type $G \in \mathcal{G}$, e.g.,

- if $G = G_1$, then $\mathbf{v}[g_i] = 1 \pm \varepsilon$ *(constant 1)*
- if $G = G_+$, then $\mathbf{v}[g_i] = \mathrm{T}(\mathbf{v}[g_j] + \mathbf{v}[g_k]) \pm \varepsilon$ *(addition)*
- if $G = G_-$, then $\mathbf{v}[g_i] = \mathrm{T}(\mathbf{v}[g_j] - \mathbf{v}[g_k]) \pm \varepsilon$ *(subtraction)*
- if $G = G_{1-}$, then $\mathbf{v}[g_i] = 1 - \mathbf{v}[g_j] \pm \varepsilon$ *(complement)*
- if $G = G_{\times 2}$, then $\mathbf{v}[g_i] = \mathrm{T}(2 \cdot \mathbf{v}[g_j]) \pm \varepsilon$ *(multiplication by 2)*
- if $G = G_\times$, then $\mathbf{v}[g_i] = \mathbf{v}[g_j] \cdot \mathbf{v}[g_k] \pm \varepsilon$ *(multiplication)*
- if $G = G_{(\cdot)^2}$, then $\mathbf{v}[g_i] = (\mathbf{v}[g_j])^2 \pm \varepsilon$ *(square)*

The associated computational problem is defined as follows for any $\varepsilon > 0$ and any choice of gate-types $\mathcal{G}$.

Definition 5.2. **ε-GCIRCUIT (with gate-types $\mathcal{G}$)**:
Input: A generalised circuit $g_1, g_2, \ldots, g_n$ with gate-types $\mathcal{G}$.
Output: An assignment $\mathbf{v} : [n] \to [0, 1]$ to the gates such that they are all ε-satisfied.

Rubinstein [Rub18] proved that this problem is PPAD-complete for some sufficiently small constant $\varepsilon > 0$ and a relatively large set of gate-types $\mathcal{G}$. In Section 5.2 we prove that the problem remains hard, even with a very restricted set of gate-types.

Theorem 5.1. *There exists a constant $\varepsilon > 0$ such that the problem ε-GCIRCUIT with gate-types $\mathcal{G} = \{G_+, G_{1-}\}$ is PPAD-complete. This continues to hold if we instead take $\mathcal{G} = \{G_1, G_-\}$.*

We can also define a problem EXACT-GCIRCUIT, where the goal is to find an assignment that *exactly* satisfies all constraints (i.e., with $\varepsilon = 0$). In Section 5.3 we prove the following result.

Theorem 5.2. *The problem EXACT-GCIRCUIT with gate-types $\mathcal{G} = \{G_{1-}, G_+, G_{(\cdot)^2}\}$ is FIXP-complete. This continues to hold if we instead take $\mathcal{G} = \{G_{1-}, G_{\times 2}, G_\times\}$.*

5.2 PPAD-completeness

In this section, we prove Theorem 5.1, namely the PPAD-completeness of the simplified ε-GCIRCUIT problem.

Membership in PPAD follows from the fact that a generalised circuit with gates $g_1, \ldots, g_n$ can be interpreted as defining an arithmetic circuit $F : [0,1]^n \to [0,1]^n$, where for $x \in [0,1]^n$ and $i \in [n]$ we let $F_i(x) = G(x_j, x_k)$, where $g_i = (G, j, k)$. Then, it is known that the problem of computing an ε-approximate fixed point of such a function F lies in PPAD [Pap94; EY10] (and in fact, even when ε is provided in the input in binary representation). Finally, note that an ε-approximate fixed point of F exactly corresponds to an ε-satisfying assignment for the generalised circuit.

In order to prove PPAD-hardness, consider the ε-GCIRCUIT problem with gate-types $\mathcal{G} = \{G_{1-}, G_+\}$, for some sufficiently small constant $\varepsilon > 0$ (which will be set later). We begin by showing that additional gate-types can be simulated if we allow a larger (but still constant) error.

$G_=$: Copy. The goal of such a gate is to copy the value of some gate g_1. For this, we use the fact that $1-(1-x) = x$. Thus, we introduce a gate g_2 of type G_{1-} with input g_1 and a gate g_3 of type G_{1-} with input g_2. It holds that $\mathbf{v}[g_3] = 1 - \mathbf{v}[g_2] \pm \varepsilon = \mathbf{v}[g_1] \pm 2\varepsilon$. In other words, we can simulate a copy gate with error at most 2ε.

G_1: Constant 1. In order to obtain a gate that has value 1, we use the fact that $x + (1 - x) = 1$. First, we introduce an arbitrary gate g_1. Then, we introduce a gate g_2 of type G_{1-} with input g_1, and a gate g_3 of type G_+ with inputs g_1 and g_2. It holds that $\mathbf{v}[g_3] = \mathrm{T}(\mathbf{v}[g_1] + \mathbf{v}[g_2]) \pm \varepsilon = 1 \pm 2\varepsilon$. Thus, we can simulate a constant 1 with error at most 2ε.

G_-: Subtraction. The goal of this gate is to compute $\mathrm{T}(\mathbf{v}[g_1] - \mathbf{v}[g_2])$. For this, we use the identity

$$\mathrm{T}(x - y) = 1 - \mathrm{T}\big((1 - x) + y\big)$$

which allows us to express subtraction using only addition and the complement operation. With this in hand, we can implement subtraction as follows. We introduce a gate g_3 of type G_{1-} with input g_1, a gate g_4 of type G_+ with inputs g_3 and g_2, and finally a gate g_5 of type G_{1-} with input g_4. Then, it holds that $\mathbf{v}[g_5] = 1 - \mathbf{v}[g_4] \pm \varepsilon = 1 - \mathrm{T}(\mathbf{v}[g_3] + \mathbf{v}[g_2]) \pm 2\varepsilon = 1 - \mathrm{T}(1 - \mathbf{v}[g_1] + \mathbf{v}[g_2]) \pm 3\varepsilon = \mathrm{T}(\mathbf{v}[g_1] - \mathbf{v}[g_2]) \pm 3\varepsilon$. Thus, we can simulate a subtraction gate with error at most 3ε.

$G_{/2}$: Division by 2. The goal of this gate is to compute $\mathbf{v}[g_1]/2$. This is achieved by constructing a cycle. Namely, we introduce two gates g_2 and g_3. The gate g_2

is of type G_- with inputs g_1 and g_3, and the gate g_3 is of type $G_=$ with input g_2. As a result, it holds that

$$\mathbf{v}[g_3] = \mathbf{v}[g_2] \pm 2\varepsilon = \mathrm{T}(\mathbf{v}[g_1] - \mathbf{v}[g_3]) \pm 5\varepsilon.$$

From this, it follows that $\mathbf{v}[g_3] = \mathbf{v}[g_1]/2 \pm 5\varepsilon$. To see this, note that if $\mathbf{v}[g_3] \geq \mathbf{v}[g_1]$, then $\mathbf{v}[g_3] = 0 \pm 5\varepsilon = \mathbf{v}[g_1]/2 \pm 5\varepsilon$, since $[0, 5\varepsilon] \subseteq [\mathbf{v}[g_1]/2 - 5\varepsilon, \mathbf{v}[g_1]/2 + 5\varepsilon]$ (because $\mathbf{v}[g_1]/2 \leq \mathbf{v}[g_3]/2 \leq 5\varepsilon$). On the other hand, if $\mathbf{v}[g_3] < \mathbf{v}[g_1]$, then we obtain that $2\mathbf{v}[g_3] = \mathbf{v}[g_1] \pm 5\varepsilon$, which again yields the same conclusion, namely $\mathbf{v}[g_3] = \mathbf{v}[g_1]/2 \pm 5\varepsilon$. Thus, we can simulate division by 2 with error at most 5ε.

$G_{\times\zeta}$: Multiplication by $\zeta \in [0, 1]$. If $\zeta = 0$, then we can simply output $G_{1-}(G_1) = 0 \pm 3\varepsilon$. If $\zeta = 1$, we can simply use a $G_=$ gate that has error at most 2ε. Consider now the case where $\zeta \in (0, 1)$. Let $k = \lceil \log_2(1/\varepsilon) \rceil$. Recall that ε will be a fixed constant, so k will also be a fixed constant. It is easy to see that in polynomial time (in the representation size of ζ) we can find $a \in \{1, 2, \ldots, 2^k - 1\}$ such that $|\zeta - a/2^k| \leq \varepsilon$.

Let g_1 denote the input. Our goal now is to compute $(a/2^k) \cdot \mathbf{v}[g_1]$, since this will be ε-close to $\zeta \cdot \mathbf{v}[g_1]$. We compute $(a/2^k) \cdot \mathbf{v}[g_1]$ in a careful manner to ensure that the error remains small. This is achieved as follows. Using the binary representation of $a = \sum_{i=0}^{k-1} a_i 2^i$, $a_i \in \{0, 1\}$, we can express the product $(a/2^k) \cdot x$ as

$$\frac{\frac{0 + a_0 \frac{x}{2}}{2} + a_1 \frac{x}{2}}{2} + a_2 \frac{x}{2}$$
$$\ddots$$

We implement this as follows. First, introduce g_2 such that $\mathbf{v}[g_2] = \mathbf{v}[g_1]/2 \pm 5\varepsilon$. Next, introduce g_3 such that (i) if $a_0 = 0$, then $\mathbf{v}[g_3] = 0 \pm 3\varepsilon$, (ii) if $a_0 = 1$, then $g_3 = g_2$. In both cases we have

$$\mathbf{v}[g_3] = a_0 \mathbf{v}[g_2] \pm 3\varepsilon.$$

Next, introduce g_4 such that (i) if $a_1 = 0$, then $\mathbf{v}[g_4] = \mathbf{v}[g_3]/2 \pm 5\varepsilon = a_0\mathbf{v}[g_2]/2 \pm 5(1 + 1/2)\varepsilon$, (ii) if $a_1 = 1$, then $\mathbf{v}[g_4] = \mathbf{v}[g_3]/2 + \mathbf{v}[g_2] \pm 6\varepsilon = a_0\mathbf{v}[g_2]/2 + \mathbf{v}[g_2] \pm 6(1 + 1/2)\varepsilon$. In both cases we have

$$\mathbf{v}[g_4] = a_0\mathbf{v}[g_2]/2 + a_1\mathbf{v}[g_2] \pm 6(1 + 1/2)\varepsilon = (a_0 + 2a_1)\mathbf{v}[g_2]/2 \pm 6(1 + 1/2)\varepsilon.$$

Next, introduce g_5 such that (i) if $a_2 = 0$, then $\mathbf{v}[g_5] = \mathbf{v}[g_4]/2 \pm 5\varepsilon = (a_0 + 2a_1)\mathbf{v}[g_2]/4 \pm 6(1 + 1/2 + 1/4)\varepsilon$, (ii) if $a_2 = 1$, then $\mathbf{v}[g_5] = \mathbf{v}[g_4]/2 + \mathbf{v}[g_2] \pm 6\varepsilon = (a_0 + 2a_1)\mathbf{v}[g_2]/4 + \mathbf{v}[g_2] \pm 6(1 + 1/2 + 1/4)\varepsilon$. In both cases we have

$$\begin{aligned}\mathbf{v}[g_5] &= (a_0 + 2a_1)\mathbf{v}[g_2]/4 + a_2\mathbf{v}[g_2] \pm 6(1 + 1/2 + 1/4)\varepsilon \\ &= (a_0 + 2a_1 + 4a_2)\mathbf{v}[g_2]/4 \pm 6(1 + 1/2 + 1/4)\varepsilon.\end{aligned}$$

Continuing in the same manner, it follows by induction that after $k-1$ such steps we obtain

$$\mathbf{v}[g_{k+2}] = \left(\sum_{i=0}^{k-1} a_i 2^i\right) \mathbf{v}[g_2]/2^{k-1} \pm 12\varepsilon = \frac{a}{2^k}\left(2\mathbf{v}[g_2]\right) \pm 12\varepsilon = \frac{a}{2^k}\mathbf{v}[g_1] \pm 22\varepsilon$$
$$= \zeta \cdot \mathbf{v}[g_1] \pm 23\varepsilon.$$

Thus, we can compute multiplication by $\zeta \in [0,1]$ with error at most 23ε. Note that this gadget can be constructed in polynomial time in the representation size of ζ. Furthermore, the number of gates needed to construct the gadget is $O(k)$, which is constant, since $k = \lceil \log_2(1/\varepsilon) \rceil$ and ε will be a fixed constant.

We are now ready to show PPAD-hardness. To do this, we reduce from a slightly modified version of GCIRCUIT studied by Goldberg et al. [GHI+20], that we call GCIRCUIT$^{[-1,1]}$. This modified version operates on $[-1,1]$ instead of $[0,1]$, and it uses the gates $G_+^{[-1,1]}$, $G_1^{[-1,1]}$ and $G_{\times-\zeta}^{[-1,1]}$ (where the gates truncate to $[-1,1]$, and $\zeta \in [0,1]$). Goldberg et al. [GHI+20] proved that ε'-GCIRCUIT$^{[-1,1]}$ is PPAD-hard for some sufficiently small constant $\varepsilon' > 0$. We now set $\varepsilon := \varepsilon'/50$. Below, we show that ε'-GCIRCUIT$^{[-1,1]}$ reduces to ε-GCIRCUIT (with gate-types $\mathcal{G} = \{G_{1-}, G_+\}$).

Given a generalised circuit with gates $G_+^{[-1,1]}$, $G_1^{[-1,1]}$ and $G_{\times-\zeta}^{[-1,1]}$, we construct a corresponding circuit with gates G_{1-} and G_+ as follows. Every gate g of the original circuit is replaced by two gates g^+ and g^-. The idea is that the value of g, which lies in $[-1,1]$, will be encoded by the values of g^+ and g^-, which lie in $[0,1]$. Formally, we interpret $\mathbf{v}[g] := \mathbf{v}[g^+] - \mathbf{v}[g^-]$. Next, we show that the constraints of the original circuit can be enforced by corresponding constraints on the new circuit.

Simulating $G_1^{[-1,1]}$. In order to enforce that $\mathbf{v}[g] = 1 \pm \varepsilon'$, we proceed as follows. We simply let $\mathbf{v}[g^+] = 1 \pm 2\varepsilon$ and $\mathbf{v}[g^-] = 0 \pm 3\varepsilon$ (using the constructions described above). Thus, it holds that $\mathbf{v}[g] = \mathbf{v}[g^+] - \mathbf{v}[g^-] = 1 \pm 5\varepsilon = 1 \pm \varepsilon'$.

Simulating $G_{\times-\zeta}^{[-1,1]}$. In order to enforce that $\mathbf{v}[g_2] = -\zeta \cdot \mathbf{v}[g_1] \pm \varepsilon'$, for some $\zeta \in [0,1]$, we proceed as follows. Using the constructions described above, we can enforce that $\mathbf{v}[g_2^+] = \zeta \cdot \mathbf{v}[g_1^-] \pm 23\varepsilon$ and $\mathbf{v}[g_2^-] = \zeta \cdot \mathbf{v}[g_1^+] \pm 23\varepsilon$. Thus, $\mathbf{v}[g_2] = -\zeta \cdot \mathbf{v}[g_1] \pm 46\varepsilon = -\zeta \cdot \mathbf{v}[g_1] \pm \varepsilon'$.

Simulating $G_+^{[-1,1]}$. In order to enforce that $\mathbf{v}[g_3] = \mathrm{T}_{[-1,1]}(\mathbf{v}[g_1] + \mathbf{v}[g_2]) \pm \varepsilon'$, we proceed in two steps. First, using our construction for performing subtraction, we "normalize" the gates by letting $\mathbf{v}[h_1^+] = \mathrm{T}(\mathbf{v}[g_1^+] - \mathbf{v}[g_1^-]) \pm 3\varepsilon$ and $\mathbf{v}[h_1^-] = \mathrm{T}(\mathbf{v}[g_1^-] - \mathbf{v}[g_1^+]) \pm 3\varepsilon$, which yields $\mathbf{v}[h_1] = \mathbf{v}[g_1] \pm 6\varepsilon$. We similarly obtain h_2 from g_2. This "normalisation" will ensure that addition is then performed correctly.

In the second step, using the addition gate G_+, we let $\mathbf{v}[g_3^+] = \mathrm{T}(\mathbf{v}[h_1^+] + \mathbf{v}[h_2^+]) \pm \varepsilon$ and $\mathbf{v}[g_3^-] = \mathrm{T}(\mathbf{v}[h_1^-] + \mathbf{v}[h_2^-]) \pm \varepsilon$. Thus, it holds that

$$\begin{aligned}\mathbf{v}[g_3] = \mathbf{v}[g_3^+] - \mathbf{v}[g_3^-] &= \mathrm{T}(\mathbf{v}[h_1^+] + \mathbf{v}[h_2^+]) - \mathrm{T}(\mathbf{v}[h_1^-] + \mathbf{v}[h_2^-]) \pm 2\varepsilon \\ &= \mathrm{T}_{[-1,1]}(\mathbf{v}[h_1^+] + \mathbf{v}[h_2^+]) - \mathrm{T}_{[-1,1]}(\mathbf{v}[h_1^-] + \mathbf{v}[h_2^-]) \pm 2\varepsilon.\end{aligned}$$

Because of the "normalisation" step, we know that

$$\min\{\mathbf{v}[h_1^+], \mathbf{v}[h_1^-]\} \leq 3\varepsilon \quad \text{and} \quad \min\{\mathbf{v}[h_2^+], \mathbf{v}[h_2^-]\} \leq 3\varepsilon.$$

In the case where $\mathbf{v}[h_1^-] \leq 3\varepsilon$ and $\mathbf{v}[h_2^-] \leq 3\varepsilon$, it holds that $\mathbf{v}[h_1] = \mathbf{v}[h_1^+] \pm 3\varepsilon$ and $\mathbf{v}[h_2] = \mathbf{v}[h_2^+] \pm 3\varepsilon$, which implies that

$$\mathbf{v}[g_3] = \mathrm{T}_{[-1,1]}(\mathbf{v}[h_1] + \mathbf{v}[h_2]) - \mathrm{T}_{[-1,1]}(\mathbf{v}[h_1^-] + \mathbf{v}[h_2^-]) \pm 8\varepsilon = \mathrm{T}_{[-1,1]}(\mathbf{v}[h_1] + \mathbf{v}[h_2]) \pm 14\varepsilon.$$

In the case where $\mathbf{v}[h_1^+] \leq 3\varepsilon$ and $\mathbf{v}[h_2^-] \leq 3\varepsilon$, it holds that $\mathbf{v}[h_1] = -\mathbf{v}[h_1^-] \pm 3\varepsilon$ and $\mathbf{v}[h_2] = \mathbf{v}[h_2^+] \pm 3\varepsilon$, which implies that

$$\begin{aligned}\mathbf{v}[g_3] &= \mathrm{T}_{[-1,1]}(\mathbf{v}[h_1^+] + \mathbf{v}[h_2]) - \mathrm{T}_{[-1,1]}(-\mathbf{v}[h_1] + \mathbf{v}[h_2^-]) \pm 8\varepsilon \\ &= \mathbf{v}[h_1] + \mathbf{v}[h_2] \pm 14\varepsilon = \mathrm{T}_{[-1,1]}(\mathbf{v}[h_1] + \mathbf{v}[h_2]) \pm 14\varepsilon.\end{aligned}$$

The remaining two cases are handled in the same way, and thus we always obtain that

$$\begin{aligned}\mathbf{v}[g_3] = \mathrm{T}_{[-1,1]}(\mathbf{v}[h_1] + \mathbf{v}[h_2]) \pm 14\varepsilon &= \mathrm{T}_{[-1,1]}(\mathbf{v}[g_1] + \mathbf{v}[g_2]) \pm 26\varepsilon \\ &= \mathrm{T}_{[-1,1]}(\mathbf{v}[g_1] + \mathbf{v}[g_2]) \pm \varepsilon'.\end{aligned}$$

Clearly, this construction can be performed in polynomial time in the size of the original generalised circuit. Furthermore, given any ε-satisfying assignment of the new generalised circuit, we can easily obtain an ε'-satisfying assignment of the original generalised circuit by setting $\mathbf{v}[g] := \mathbf{v}[g^+] - \mathbf{v}[g^-] \in [-1, 1]$ for all gates g. It follows that the ε-GCIRCUIT problem with gate-types $\mathcal{G} = \{G_{1-}, G_+\}$ is PPAD-hard.

Finally, note that if we let $\mathcal{G} = \{G_1, G_-\}$ instead, we again obtain the same result, because G_{1-} and G_+ can easily be simulated. Indeed, it is clear that G_{1-} can immediately be simulated. Furthermore, G_+ can be simulated by using the equation $\mathrm{T}(x + y) = 1 - \mathrm{T}((1 - x) - y)$.

5.3 FIXP-completeness

In this section, we prove Theorem 5.2, namely the FIXP-completeness of the EXACT-GCIRCUIT problem.

Membership in FIXP follows immediately by noting that a generalised circuit with gates $g_1, \ldots, g_n$ defines an arithmetic circuit $F : [0, 1]^n \to [0, 1]^n$, where for $x \in [0, 1]^n$

and $i \in [n]$ we let $F_i(x) = G(x_j, x_k)$, where $g_i = (G, j, k)$. Indeed, any fixed point of F corresponds to an assignment that exactly satisfies the gate constraints. In particular, note that all the gate-types we consider can be exactly computed using the usual operations allowed in FIXP, namely $+, \times, \max$ and rational constants. Furthermore, it is easy to see that this trivially yields an SL-reduction (defined in Section 2.4).

In order to prove FIXP-hardness we will show that our very restricted set of gates is actually enough to simulate various more complex gates. Deligkas et al. [DFMS21, Section 7.2], using a special Brouwer function for the FIXP-complete problem 3-Nash given by Etessami and Yannakakis [EY10], proved that computing fixed points of very restricted arithmetic circuits is already FIXP-hard. In more detail, they consider functions $F : [0,1]^n \to [0,1]^n$ computed by circuits with a restricted set of gates and such that every gate always has value in $[0,1]$, for any input $x \in [0,1]^n$ to the circuit. Because of this property we can use our gates that truncate to $[0,1]$ without changing any of the computations.

In more detail, they allow the following gates: G_ζ (constant $\zeta \in \mathbb{Q} \cap [0,1]$), G_+, G_- (subtraction truncated to $[0,1]$), $G_\times$, $G_{\times 2}^{[0,1]}$, $G_{\max}$ and $G_{\min}$. We show below that we can simulate all of these gates, using only the gates G_{1-}, $G_{\times 2}$ and $G_\times$ (or alternatively, G_{1-}, G_+ and $G_{(\cdot)^2}$). In particular, $G_{\times 2}^{[0,1]}$ is a restricted gate $G_{\times 2}$ that only works on inputs in $[0, 1/2]$. Since our $G_{\times 2}$ gate has the same behavior as that gate for such inputs, it is correctly simulated.

Finally, we simply use copy gates $G_=$ to enforce the fixed point constraint, namely that the ith input to F be equal to its ith output. It is easy to see that this construction yields a polynomial-time reduction, and that it is in fact an SL-reduction (as defined in Section 2.4), since we only need to extract the values assigned to the input gates in order to obtain a fixed point of F. In the remainder of this proof, we show how all the required gates can be simulated using our restricted set of gates G_{1-}, $G_{\times 2}$ and $G_\times$.

$G_=$: Copy. In order to copy the value of some gate g_1, we use the complement gate G_{1-} twice. Namely, we first introduce a gate g_2 of type G_{1-} with input g_1, and then another gate g_3 of type G_{1-} with input g_2. Clearly it holds that $\mathbf{v}[g_3] = 1 - \mathbf{v}[g_2] = 1 - (1 - \mathbf{v}[g_1]) = \mathbf{v}[g_1]$.

$G_{1/2}$: Constant 1/2. In order to obtain a gate that has value $1/2$, we create a small cycle. We introduce two gates g_1 and g_2. The gate g_1 is of type $G_=$ with input g_2, and the gate g_2 is of type G_{1-} with input g_1. It follows that $\mathbf{v}[g_1]$ satisfies $\mathbf{v}[g_1] = 1 - \mathbf{v}[g_1]$, which implies $\mathbf{v}[g_1] = 1/2$. Note that together with the $G_\times$ gate we can now also perform multiplication by $1/2$, denoted by $G_{\times 1/2}$.

G_-: Subtraction. In the proof of Lemma 5.1 below, we show how to construct a subtraction gate given access only to G_{1-}, $G_{\times 2}$ and a special gate G_ϕ, where

$\phi : [0,1]^2 \to [0,1]$, $(x,y) \mapsto (x+1)(y+1)/4$. Thus, to obtain the subtraction gate, it is enough for us here to construct a gate G_ϕ. Since we have access to $G_\times$, it suffices to construct a gate that implements the function $x \mapsto (x+1)/2$. Let g_1 be the input gate. We introduce a gate g_2 of type G_{1-} with input g_1, a gate g_3 of type $G_{\times 1/2}$ with input g_2, and finally a gate g_4 of type G_{1-} with input g_3. It follows that $\mathbf{v}[g_4] = 1 - \mathbf{v}[g_3] = 1 - \mathbf{v}[g_2]/2 = 1 - (1 - \mathbf{v}[g_1])/2 = (\mathbf{v}[g_1] + 1)/2$, as desired.

G_+: Addition. Addition can easily be obtained from subtraction by using the following equality for all $x, y \in [0,1]$

$$\mathrm{T}(x+y) = 1 - \mathrm{T}((1-x) - y) = G_{1-}(G_-(G_{1-}(x), y)).$$

$G_{\max}, G_{\min}$: Maximum and Minimum. The function $(x,y) \mapsto \max\{x,y\}$ can easily be simulated with existing gates by noting that

$$\max\{x,y\} = \mathrm{T}(x + \mathrm{T}(y-x)) = G_+(x, G_-(y,x)).$$

Then, $(x,y) \mapsto \min\{x,y\}$ can simply be obtained by $\min\{x,y\} = 1 - \max\{1-x, 1-y\}$.

$G_{\times k}$: Multiplication by integer k. Let k be an integer that is given in binary representation, i.e., $k = \sum_{i=0}^{\ell} a_i 2^i$, where $a_i \in \{0,1\}$. Our goal is to construct a gate that computes $x \mapsto \mathrm{T}(k \cdot x)$. Using the $G_{\times 2}$ gate we can compute $\mathrm{T}(2^i \cdot x)$ for $i = 0, 1, \ldots, \ell$. This requires ℓ separate $G_{\times 2}$ gates. Then, we use the addition gate to compute

$$\mathrm{T}\left(\sum_{i:\, a_i = 1} \mathrm{T}(2^i \cdot x)\right) = \mathrm{T}\left(\sum_{i=0}^{\ell} a_i 2^i x\right) = \mathrm{T}(k \cdot x).$$

This uses at most ℓ separate G_+ gates. Thus, overall we use a number of gates that is polynomial in the representation length of k.

G_ζ: Constant $\zeta \in [0,1] \cap \mathbb{Q}$. If $\zeta = 1$, we can simply do $G_{\times 2}(G_{1/2}) = 1$. If $\zeta = 0$, we can do $G_{1-}(G_{\times 2}(G_{1/2})) = 0$. Now assume that $\zeta \in (0,1)$. Write $\zeta = c/d$ where c and d are positive integers, $c \geq 1$, $c < d$, $d \geq 2$. Clearly, if we can construct the constant $1/d$, then we can use a $G_{\times k}$ gate with $k = c$ to obtain ζ. In order to construct $1/d$, we use a small cycle. We introduce two gates g_1 and g_2. The gate g_1 is of type $G_{\times k}$ with $k = d - 1$ and with input g_2. The gate g_2 is of type G_{1-} with input g_1. Thus, it holds that $\mathbf{v}[g_2] = 1 - \mathbf{v}[g_1] = 1 - \mathrm{T}((d-1) \cdot \mathbf{v}[g_2])$. It is easy to check that the only solution of this equation is $\mathbf{v}[g_2] = 1/d$.

Finally, let us show that the set of gate-types G_{1-}, G_+ and $G_{(\cdot)^2}$ also suffices to simulate all the gates above, by showing that they can simulate G_{1-}, $G_{\times 2}$ and $G_\times$. As before, G_{1-} can be used to create $G_=$. Then, G_+ and $G_=$ can be used to obtain $G_{\times 2}$. Thus, it remains to simulate $G_\times$.

Note that G_- can be obtained by $\mathrm{T}(x-y) = 1-(\mathrm{T}((1-x)+y))$. Furthermore, we can construct $G_{\times 1/2}$ on input gate g_1 as follows. We introduce two gates g_2 and g_3. The gate g_2 is of type G_- and has inputs g_1 and g_3. The gate g_3 is of type $G_=$ with input g_2. It follows that $\mathbf{v}[g_3] = \mathrm{T}(\mathbf{v}[g_1] - \mathbf{v}[g_3])$, which has the only solution $\mathbf{v}[g_3] = \mathbf{v}[g_1]/2$.

In order to simulate $G_\times$, note that

$$\left(\frac{x}{2} + \frac{y}{2}\right)^2 = (x/2)^2 + (y/2)^2 + xy/2.$$

We can easily compute $x/2 + y/2$ and then square using $G_{(\cdot)^2}$. Similarly, we can also compute $(x/2)^2 + (y/2)^2$. By using G_-, we then obtain $xy/2$, and thus xy after using a $G_{\times 2}$ gate.

5.4 PPAD- and FIXP-completeness with a Customised Gate

In this section, we consider the GCIRCUIT problem with a customised set of gates, which will be used in Chapter 6 for the first-price auction equilibrium problem. This also serves as an illustration of how flexible the GCIRCUIT problem can be.

In more detail, we consider the gate-types $\mathcal{G} = \{G_{\times 2}, G_{1-}, G_\phi\}$, where $\phi : [0,1]^2 \to [0,1]$, $(x,y) \mapsto \frac{1}{4}(x+1)(y+1)$. This means that a gate $g_i = (G_\phi, j, k)$ enforces the constraint $\mathbf{v}[g_i] = \phi(\mathbf{v}[g_j], \mathbf{v}[g_k]) \pm \varepsilon$. We prove that the problem remains hard with these gates.

Lemma 5.1. *Let $\mathcal{G} = \{G_{\times 2}, G_{1-}, G_\phi\}$. There exists a constant $\varepsilon > 0$ such that the problem ε-*GCIRCUIT *with gate-types $\mathcal{G}$ is* PPAD*-complete. Furthermore,* EXACT-GCIRCUIT *with gate-types $\mathcal{G}$ is* FIXP*-complete.*

Proof. In order to prove that the problem remains hard with $\mathcal{G} = \{G_{\times 2}, G_{1-}, G_\phi\}$, we will show that other gate-types can be simulated using only these three gate-types. Let $\varepsilon \in [0, 1/14]$ and assume that we have access to gates of type $G_{\times 2}$, G_{1-} and G_ϕ.

***G_1*: Constant 1.** In order to create a constant 1 we use the fact that for any $x, y \in [0,1]$

$$\mathrm{T}\left(2^3 \cdot \phi(x,y)\right) = \mathrm{T}\left(2^3(x+1)(y+1)/4\right) \geq \mathrm{T}(2) = 1.$$

In more detail, we use a gate g_1 of type G_ϕ (with arbitrary inputs), then a gate g_2 of type $G_{\times 2}$ with input g_1, another gate g_3 of type $G_{\times 2}$ with input g_2, and finally another gate g_4 of type $G_{\times 2}$ with input g_3. We have that $\mathbf{v}[g_1] \geq 1/4 - \varepsilon$, $\mathbf{v}[g_2] \geq \mathrm{T}(2 \cdot \mathbf{v}[g_1]) - \varepsilon \geq 1/2 - 3\varepsilon$, $\mathbf{v}[g_3] \geq \mathrm{T}(2 \cdot \mathbf{v}[g_2]) - \varepsilon \geq 1 - 7\varepsilon$, and $\mathbf{v}[g_4] \geq \mathrm{T}(2 \cdot \mathbf{v}[g_3]) - \varepsilon \geq 1 - \varepsilon$, since $\varepsilon \leq 1/14$. Thus, we can construct a gate that has the value $1 \pm \varepsilon$.

$G_{/2}$: Division by 2. In order to divide the value of some gate g_1 by 2, we use the fact that

$$1 - \phi(1 - \mathbf{v}[g_1], 1) = 1 - (2 - \mathbf{v}[g_1])(1 + 1)/4 = \mathbf{v}[g_1]/2.$$

In more detail, we use a gate g_2 of type G_{1-} with input g_1, then we use a gate g_3 of type G_ϕ with inputs g_2 and a constant $1 \pm \varepsilon$, and finally we use a gate g_4 of type G_{1-} with input g_3. It holds that $\mathbf{v}[g_2] = 1 - \mathbf{v}[g_1] \pm \varepsilon$, $\mathbf{v}[g_3] = \phi(\mathbf{v}[g_2], 1 \pm \varepsilon) \pm \varepsilon = 1 - \mathbf{v}[g_1]/2 \pm 2\varepsilon$, and $\mathbf{v}[g_4] = 1 - \mathbf{v}[g_3] \pm \varepsilon = \mathbf{v}[g_1]/2 \pm 3\varepsilon$. Thus, we can construct a gate that performs division by 2 with error at most 3ε.

$G_{=}$: Copy. It is easy to see that using two gates of type G_{1-}, one after the other, copies the original value with error at most 2ε.

G_{inv}: Inverse. We now show how to construct the gate G_{inv}, which computes the function $x \mapsto -1 + 4/(2 + x)$, and will be very useful to construct the subtraction gate below. The construction of G_{inv} uses a cycle. Let g_1 be the input gate. We first use a gate g_2 of type G_{1-} with input g_1, then we use a gate g_3 of type G_ϕ with input g_2 and g_4, and finally we let gate g_4 be of type $G_=$ with input g_3. We have that $\mathbf{v}[g_2] = 1 - \mathbf{v}[g_1] \pm \varepsilon$, $\mathbf{v}[g_3] = \phi(\mathbf{v}[g_2], \mathbf{v}[g_4]) \pm \varepsilon$, and $\mathbf{v}[g_4] = \mathbf{v}[g_3] \pm 2\varepsilon$. It follows that $\mathbf{v}[g_4]$ must satisfy the equation

$$\mathbf{v}[g_4] = \phi(\mathbf{v}[g_2], \mathbf{v}[g_4]) \pm 3\varepsilon = (\mathbf{v}[g_2] + 1)(\mathbf{v}[g_4] + 1)/4 \pm 3\varepsilon$$

which implies that

$$\mathbf{v}[g_4] = \frac{1 + \mathbf{v}[g_2]}{3 - \mathbf{v}[g_2]} \pm 6\varepsilon.$$

As a result, we obtain that

$$\mathbf{v}[g_4] = \frac{2 - \mathbf{v}[g_1]}{2 + \mathbf{v}[g_1]} \pm 8\varepsilon = -1 + \frac{4}{2 + \mathbf{v}[g_1]} \pm 8\varepsilon$$

i.e., we can compute the function with error at most 8ε.

G_{-}: Subtraction. Given gates g_1 and g_2, we want to obtain $\mathrm{T}(\mathbf{v}[g_1] - \mathbf{v}[g_2])$. To achieve this, we first use the fact that

$$\phi\left(\phi\left(-1 + \frac{4}{2 + y}, 1 - x\right), \frac{y}{2}\right) = \phi\left(\frac{2 - x}{2 + y}, \frac{y}{2}\right) = \frac{1}{2} + \frac{1}{8}(y - x).$$

In more detail, we first use a gate g_3 of type G_{inv} with input g_2, then a gate g_4 of type G_{1-} with input g_1, then a gate g_5 of type G_ϕ with inputs g_3 and g_4, then a gate g_6 of type $G_{/2}$ with input g_2, and finally a gate g_7 of type G_ϕ with inputs g_5 and g_6. We thus obtain that $\mathbf{v}[g_3] = -1 + 4/(2 + \mathbf{v}[g_2]) \pm 8\varepsilon$, $\mathbf{v}[g_4] = 1 - \mathbf{v}[g_1] \pm \varepsilon$, and $\mathbf{v}[g_5] = (2 - \mathbf{v}[g_1])(2 + \mathbf{v}[g_2]) \pm 7\varepsilon$. Furthermore, it holds that $\mathbf{v}[g_6] = \mathbf{v}[g_2]/2 \pm 3\varepsilon$, and thus $\mathbf{v}[g_7] = 1/2 + (\mathbf{v}[g_2] - \mathbf{v}[g_1])/8 \pm 11\varepsilon$.

Next, we can obtain the subtraction operation from this by noting that

$$4\left(1-\mathrm{T}\left(2\left(\frac{1}{2}+\frac{1}{8}(y-x)\right)\right)\right)=4\left(1-\left(1-\frac{1}{4}\,\mathrm{T}(x-y)\right)\right)=4\frac{\mathrm{T}(x-y)}{4}=\mathrm{T}(x-y).$$

This is implemented by using a gate g_8 of type $G_{\times 2}$ with input g_7, then a gate g_9 of type G_{1-} with input g_8, then a gate g_{10} of type $G_{\times 2}$ with input g_9, and finally another gate g_{11} of type $G_{\times 2}$ with input g_{10}. It holds that

$$\mathbf{v}[g_8]=\mathrm{T}(2\cdot\mathbf{v}[g_7])\pm\varepsilon=1-\mathrm{T}(\mathbf{v}[g_1]-\mathbf{v}[g_2])/4\pm 23\varepsilon.$$

As a result, it then holds that $\mathbf{v}[g_9]=\mathrm{T}(\mathbf{v}[g_1]-\mathbf{v}[g_2])/4\pm 24\varepsilon$, $\mathbf{v}[g10]=\mathrm{T}(\mathbf{v}[g_1]-\mathbf{v}[g_2])/2\pm 49\varepsilon$, and finally $\mathbf{v}[g_{11}]=\mathrm{T}(\mathbf{v}[g_1]-\mathbf{v}[g_2])\pm 99\varepsilon$. Thus, we can compute subtraction with error at most 99ε.

$G_\times$: Multiplication. Given gates g_1 and g_2, we want to obtain $\mathbf{v}[g_1]\cdot\mathbf{v}[g_2]$. We only perform the construction for the case $\varepsilon=0$, since we only need this gate for the FIXP-hardness. Note that we can multiply by 4 using two consecutive $G_{\times 2}$ gates. Similarly, we can divide by 4 using two consecutive $G_{/2}$ gadgets. To perform multiplication, we use the fact that

$$\phi(x,y)-\frac{1}{4}-\frac{x}{4}-\frac{y}{4}=\frac{xy}{4}.$$

In more detail, we first use a gate g_3 of type G_ϕ with input g_1 and g_2, then a gate g_4 of type $G_{/4}$ with input the constant 1, then a gate g_5 of type G_- with inputs g_3 and g_4, then a gate g_6 of type $G_{/4}$ with input g_1, then a gate g_7 of type G_- with inputs g_5 and g_6, then a gate g_8 of type $G_{/4}$ with input g_2, then a gate g_9 of type G_- with inputs g_7 and g_8, and finally a gate g_{10} of type $G_{\times 4}$ with input g_9. We have that

$$\mathbf{v}[g_3]=\phi(\mathbf{v}[g_1],\mathbf{v}[g_2])=(\mathbf{v}[g_1]+\mathbf{v}[g_2]+\mathbf{v}[g_1]\cdot\mathbf{v}[g_2]+1)/4.$$

Then we obtain that $\mathbf{v}[g_5]=(\mathbf{v}[g_1]+\mathbf{v}[g_2]+\mathbf{v}[g_1]\cdot\mathbf{v}[g_2])/4$, $\mathbf{v}[g_7]=(\mathbf{v}[g_2]+\mathbf{v}[g_1]\cdot\mathbf{v}[g_2])/4$, $\mathbf{v}[g_9]=\mathbf{v}[g_1]\cdot\mathbf{v}[g_2]/4$, and finally $\mathbf{v}[g_{10}]=\mathbf{v}[g_1]\cdot\mathbf{v}[g_2]$. Thus, we can perform exact multiplication when $\varepsilon=0$.

Hardness. We have shown that we can simulate gates G_1 and G_- with error at most 99ε. Thus, by Theorem 5.1, the PPAD-hardness of our customised version follows. For the case $\varepsilon=0$, we have shown that we can exactly simulate gates $G_{\times 2}$, G_{1-} and $G_\times$. As a result, by Theorem 5.2, the exact problem for our customised version is FIXP-hard. The membership results hold as before. □

Chapter 6

The Complexity of First-Price Auctions with Subjective Priors

In this chapter, we study the complexity of computing an equilibrium in a first-price auction with continuous value distributions and discrete bidding space. We prove that when bidders have independent *subjective* prior beliefs about the value distributions of the other bidders, computing an ε-equilibrium of the auction is PPAD-complete, and computing an *exact* equilibrium is FIXP-complete.

6.1 The First-Price Auction

Auctions are prime examples of economic environments in which the element of strategic behaviour is prevalent. The associated theory can be traced back to as early as the 1960s and the seminal work of Vickrey [Vic61]. Over the years, auction theory and mechanism design have produced some of the most celebrated results in economics, as can be evidenced, e.g., by the relevant 1996, 2007 and 2020 Nobel Prizes.[1] Among the plethora of auction formats that this rich literature has proposed, some stand out, such as the second-price auction of Vickrey [Vic61] or the revenue-maximising auction of Myerson [Mye81].

Arguably, though, the most fundamental auction format is that of the *first-price auction*, in which the highest bidder wins and is charged an amount equal to her bid. Compared to its counterparts mentioned above, the first-price auction does not enjoy the same desirable incentive properties: participants may have an incentive to misreport their true bids. At the same time, however, the first-price auction is very natural and simple to describe, implement and participate in, making it very suitable for a range of important applications. As a matter of fact, several online ad exchanges,

[1] For the official Nobel Prize announcements see:
https://www.nobelprize.org/prizes/economic-sciences/1996/summary/
https://www.nobelprize.org/prizes/economic-sciences/2007/summary/
https://www.nobelprize.org/prizes/economic-sciences/2020/summary/

including Google Ad Manager, have adopted this auction format for selling their ads, in what has been coined "the first-price movement" (see, e.g., [PST20]).

There has been a large body of work studying incentives and bidding behaviour in first-price auctions, dating back to the original paper of Vickrey [Vic61]. In particular, the literature has studied the equilibria of the auction in an incomplete information setting where the bidders have only probabilistic prior beliefs (or simply *priors*) about the values of other bidders, via the lens of Bayesian game theory [Har67] (see also [Mye97; Har12]). Several different scenarios of interest have been analysed; see, e.g., [GLS67; RS81; Plu92; MMRS94; Leb96; Leb99; MR00; LP00; Ath01; RZ04; CH13; BBM17]. It is no exaggeration to say that understanding the Bayes-Nash equilibria of the first-price auction has historically been one of the most important questions of auction theory.

The aforementioned literature has been primarily concerned with identifying conditions under which (pure Bayes-Nash) equilibria are guaranteed to exist. Among those, the seminal paper of Athey [Ath01] has been pivotal in establishing the existence of equilibria for fairly general settings with continuous priors. A natural follow-up question posed explicitly by Athey [Ath01], which was also very much present in earlier works, is whether these equilibria can also be "found"; in the context of the related literature, this is usually interpreted as coming up with closed-form solutions that describe them.

The computational hardness results we prove in this chapter can be interpreted intuitively as justification of why research in economics has only had limited success in providing closed forms or characterisations for the equilibria of the first-price auction. In addition, we consider it to be a quite valuable addition to the literature of total search problems [MP91], as it concerns the computation of equilibria of one of the most fundamental games in auction theory.

Continuous Priors, Discrete Bids. We consider the setting where the bidders' beliefs about the values of other bidders are continuous distributions, whereas the bidding space is a discrete set. The former assumption is standard in auction theory (see, e.g., [Mye97, Sec. 3.11] or [Kri09]). From a technical standpoint, this also guarantees the existence of equilibria [Ath01].[2] The assumption of the discrete bidding space is clearly motivated by any real-world scenario, in which the bids will be increments of some minimum monetary amount, e.g., 1 dollar or 1 cent, depending on the application. This setting has in fact been studied in several works for first-price auctions in particular (see, e.g., [Chw89; Ath01; EMS09; CWG10; RG21]).

Subjective Priors. For our hardness results we assume that the priors are subjective, meaning that two different bidders might have different beliefs about the values of some

[2]It is important to note here that in some versions of the problem, even *mixed* Bayes-Nash equilibria are not guaranteed to exist; see, e.g., [Leb96].

other bidder. In the auction theory literature, it is often assumed that a "universal" prior exists, which is common knowledge among all players; this is known as the *independent private values* model. Indeed, such common priors are quite convenient in settings where there is an aggregate objective that needs to be optimised in expectation (e.g., the social welfare or the seller's revenue), since it can be used by the designer to tune the parameters of the auction in a way that works best for the optimisation goal at hand; this is the case, e.g., for Myerson's revenue-maximizing auction [Mye81].

From our perspective however, where the goal is to study the players' incentives and compute an equilibrium, we believe it is natural to make the more general assumption that priors are still independent, but subjective: this is enough for the bidders to come up with their best responses. As a matter of fact, Harsanyi's original paper [Har67], as well as classic textbooks in economics (e.g., [Mye97; JR01]) introduce Bayesian games directly in the context of subjective beliefs.[3] Similar notions of subjective priors and "subjective equilibria" have also been studied rather extensively for general Bayesian games in economics [Hah73; FL86; BG97; BGM92; KL93; KL95; RW94] and computer science [WP12; FW16].

The subjective priors assumption is necessary for our hardness results. Settling the case of common priors remains an important open question. Our results for subjective priors, besides being of standalone interest, can also be viewed as an important first step in the quest of answering this question. We remark that our PPAD- and FIXP-membership results obviously also apply to common priors, which are just a special case of subjective beliefs.

Related work. As we mentioned earlier, there is a significant amount of work in economic theory on the equilibria of the first-price auction [GLS67; RS81; Plu92; MMRS94; Leb96; Leb99; MR00; LP00; Ath01; RZ04; BBM17]. Among those, the most relevant work to us is that of Athey [Ath01], who established the existence of pure Bayes-Nash equilibria in games with discontinuous payoffs which satisfy the *single crossing property* of Milgrom and Shannon [MS94], of which the first-price auction is a special case. Athey's proof applies to both discrete and continuous bidding spaces, and in fact the latter is established through the former, via a limit argument similar in spirit to [Leb96; MR00].

To the best of our knowledge, there are only a few prior works on the computational complexity of equilibria in first-price auctions. Escamocher, Miltersen and Santillan R. [EMS09] study the problem of computing equilibria when *both* the priors and the bidding space are discrete. In that case, it is not hard to construct counter-examples that show that pure equilibria may not exist, and therefore they are concerned with

[3] These works also usually provide discussions on "consistency" conditions, e.g., see [Har67] and [Mye97, Sec. 2.8].

the question of *deciding* their existence. Their results do not provide a conclusive answer (i.e., neither NP-hardness nor polynomial-time solvability is proven), except for the very special case of two bidders with bi-valued distributions. Wang, Shen and Zuo [WSZ20] very recently studied the equilibrium computation problem in settings with *discrete priors* and *continuous bids* (in a sense, the opposite of what we do here), and under the *Vickrey tie-breaking rule* for deciding the winner of the auction in case of a tie. According to this rule, ties are resolved by running an auxiliary second-price (Vickrey) auction among the potential winners of the first-price auction; effectively this allocates the item to the bidder with highest true valuation. This tie-breaking rule was introduced by Maskin and Riley [MR00] primarily as a technical tool in proving their existence results for the *uniform tie-breaking rule*, where ties are broken uniformly at random among the bidders with the highest bid. Our results are proven for the uniform tie-breaking rule, which is the standard rule in the literature of the problem [Leb96; MR00; Ath01; Kri09].

Finally, we note that Cai and Papadimitriou [CP14] have studied the complexity of Bayesian *combinatorial* auctions, a more complicated auction format which typically involves multiple items for sale and more complex agents' valuations over subsets of items. The complexity of *general* Bayesian games (beyond auctions) has been studied in the literature, primarily resulting in NP-hardness results for several cases of interest, e.g., see [GGM07; CS08].

6.1.1 Model and Notation

In a (Bayesian) *first-price auction* (FPA), there is a set $N = \{1, 2, \ldots, n\}$ of *bidders* (or *players*) and one item for sale. Each player i submits a *bid* $b_i \in B$, where the *bidding space* $B \subseteq [0, 1]$ is a finite set. We will also make the standard assumption (often referred to as the "null bid" in the literature) that $0 \in B$, which can be interpreted as the option of the bidders to not participate in the auction (see, e.g., [MR00; Ath01]).

The item is allocated to the player with the highest bid, who is charged a payment equal to her bid. If there are multiple players submitting the same highest bid, the winner is determined based on the *uniform tie-breaking* rule. Formally, for a *bid profile* $\boldsymbol{b} = (b_1, \ldots, b_n)$, the *ex-post utility* of player i with true value v_i is given by

$$\tilde{u}_i(\boldsymbol{b}; v_i) \equiv \begin{cases} \frac{1}{|W(\boldsymbol{b})|}(v_i - b_i), & \text{if } \; i \in W(\boldsymbol{b}), \\ 0, & \text{otherwise}, \end{cases} \qquad \text{where } \; W(\boldsymbol{b}) = \operatorname*{argmax}_{j \in N} b_j \tag{6.1}$$

For each pair of players $i, j \in N$, $i \neq j$, there is a continuous value distribution $F_{i,j}$ over $[0, 1]$; we call this distribution the *prior* of bidder i over the values of bidder j. The *subjective belief* of player i for the values $\boldsymbol{v}_{-i} = (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$ of the other bidders is then given by the product distribution $\boldsymbol{F}_{-i} \equiv \times_{j \neq i} F_{i,j}$. In other words,

from the perspective of bidder i, the values v_j for $j \neq i$ are drawn *independently* from distributions $F_{i,j}$. Notice that the special case where $F_{i,j} = F_{i',j}$ for all $j \in N$ and $i, i' \in N \setminus \{j\}$ corresponds to the classic *independent private values* model of auction theory, where the value of each bidder is drawn (independently of the others) from a single distribution. More formally, simplifying the notation by using F_j instead of $F_{i,j}$, $\boldsymbol{v}$ is drawn from the *common prior* distribution $\boldsymbol{F} = \times_{i \in N} F_i$.

The FPA described above naturally induces a game in which each bidder i selects her bid based on her own (true) value v_i, and her beliefs $\boldsymbol{F}_{-i}$. A *strategy* of bidder i is a function $\beta_i : [0,1] \to B$ mapping values to bids. Given a strategy profile $\boldsymbol{\beta}_{-i}$ of the other players, the (ex-interim) *utility* of player i with true value v_i when bidding $b \in B$ is

$$u_i(b, \boldsymbol{\beta}_{-i}; v_i) \equiv \mathbb{E}_{\boldsymbol{v}_{-i} \sim \boldsymbol{F}_{-i}} \left[\tilde{u}_i(b, \boldsymbol{\beta}_{-i}(\boldsymbol{v}_{-i}); v_i) \right],$$

where $\boldsymbol{\beta}_{-i}(\boldsymbol{v}_{-i})$ is a shorthand for $(\beta_1(v_1), \ldots, \beta_{i-1}(v_{i-1}), \beta_{i+1}(v_{i+1}), \cdots, \beta_n(v_n))$. Intuitively, the player calculates her (expected) utility by drawing a value v_j for each bidder $j \neq i$ from her corresponding subjective prior distribution $F_{i,j}$, and then using the strategy "rules" $\boldsymbol{\beta}_{-i}$ of the others to map their values to actual bids in B.

We are interested in "stable" states of the FPA, i.e., strategy profiles from which no bidder would like to unilaterally deviate to a different strategy. Formally, we have the following definition.

Definition 6.1 (ε-Bayes-Nash equilibrium of the FPA)**.** Let $\varepsilon > 0$. A strategy profile $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$ is a (pure, ex-interim) ε-Bayes-Nash equilibrium (ε-BNE) of the FPA if for any bidder $i \in N$ and any value $v_i \in [0,1]$,

$$u_i(\beta_i(v_i), \boldsymbol{\beta}_{-i}; v_i) \geq u_i(b, \boldsymbol{\beta}_{-i}; v_i) - \varepsilon \qquad \text{for all } \; b \in B.$$

Given a fixed strategy profile $\boldsymbol{\beta}_{-i}$ of the other bidders, we will denote the set of *ε-best responses* of player i by

$$BR_i^{\varepsilon}(\boldsymbol{\beta}_{-i}) = \left\{ \beta_i \;\middle|\; u_i(\beta_i(v_i), \boldsymbol{\beta}_{-i}; v_i) \geq \max_{b \in B} u_i(b, \boldsymbol{\beta}_{-i}; v_i) - \varepsilon \quad \text{for all } \; v_i \in [0,1] \right\}$$

Using this, the condition in Definition 6.1 can be equivalently written as $\beta_i \in BR_i^{\varepsilon}(\boldsymbol{\beta}_{-i})$ for all players i. For the special case of $\varepsilon = 0$, i.e., *exact* best-responses, we will drop the ε superscript.

Notice that, in Definition 6.1 we define a relaxed equilibrium concept, in which the bidder does not want to change to a different strategy unless it increases her utility by an additive factor larger than ε; obviously, when $\varepsilon = 0$ we recover the standard definition of the (exact) Bayes-Nash equilibrium.

No Overbidding. As part of our model, we will make the assumption that bidders will never submit a bid b_i which is higher than their valuation v_i. This is a standard

assumption in the literature of the first-price auction [MR00; MR03; Leb06; EMS09; WSZ20] and auctions in general [CKK+15; LB10; BR11; FFGL20; CKS16; LT10]. The rationale behind it stems from the fact that, given the format of the utilities in the FPA (see (6.1)), it is arguably unreasonable to overbid, as bidding 0 will *always* result in at least the same utility. In game-theoretic terms, the overbidding strategy is *weakly dominated* by bidding 0, which can be interpreted as abstaining from the auction. These strategies are typically excluded from consideration to rule out unnatural equilibria (see [FFGL20] for a discussion).

We are now ready to formally define our computational problem of finding an ε-equilibrium of the FPA.

Definition 6.2. **ε-BNE-FPA**:

Input:

- a set of bidders $N = \{1, 2, \ldots, n\}$,
- a finite bidding space $B \subseteq [0, 1]$,
- for each pair $i, j \in N$, a continuous value distribution $F_{i,j}$ over $[0, 1]$.

Output: An ε-Bayes-Nash equilibrium $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$.

We will use the term EXACT-BNE-FPA instead of 0-BNE-FPA to denote the computational problem of finding an exact Bayes-Nash equilibrium of the auction. Some remarks related to the definition above are in order.

The Input Model for the Distributions. We have intentionally vaguely stated that the distributions $F_{i,j}$ should be provided as input to the problem, but we have not specified exactly how. Our membership results hold even when the functions $F_{i,j}$ are fairly general, and can be concisely and efficiently represented in a form that is appropriate for computation. The appropriate input models for PPAD and FIXP are presented in Section 6.3.3 and Section 6.3.2 respectively. For the negative results on the other hand, we use fairly simple distributions $F_{i,j}$ – this only makes our results stronger. In particular, we use *piecewise-constant* density functions, which can be represented by the endpoints and the value for each interval.

Explicit Bidding Space. We assume that the bidding space is explicitly given as part of the input. This assumption is required in Section 6.2 in order to show that we can compute best-responses efficiently. Even in the mildest of settings where the bidding space is given implicitly, computing best-responses turns out to be computationally and information-theoretically hard, see [FGH+21, Appendix B].

Equilibrium Representation. Besides the representation of the input, the output of our computational problem, i.e., the equilibrium of the FPA, should also be represented

in some concise and efficient way. Following the standard literature of the problem, we will consider equilibria for which the strategy $\beta_i(v_i)$ of each bidder is a non-decreasing function of her value v_i (e.g., see [Ath01; MR00; RZ04] and [Kri09, Appendix G]) for which the existence of an equilibrium is always guaranteed [Ath01]. These equilibria are in a sense the only "natural" ones, as, similar to the case of overbidding (see earlier discussion), any bidder's strategy is weakly dominated by a non-decreasing strategy.

Based on this, there is a straightforward and computationally efficient way of representing the best response of each player, as a step function with a finite set of "jump points", corresponding to the values at which the bidder "jumps" from one bid to the next [Ath01]. Formally, we define

$$\alpha_i(b) = \sup\{v \mid \beta_i(v) \leq b\}. \tag{6.2}$$

Intuitively, $\alpha_i(b)$ is the largest value for which player i would bid b or lower. With a slight abuse of notation, we can write $\alpha_i = \beta_i^{-1}$, that is, α_i can be interpreted as an *inverse bidding* strategy. In that way, we can also rework β_i from α_i, as $\beta_i(v) = b$, where $v \in (\alpha_i(b^-), \alpha_i(b)]$ for any $b \in B$. Here we let b^- denote the previous bid, i.e., the largest $b' \in B$ with $b' < b$. Finally, to be able to handle the corner cases in a unified way, we set $a_i(b^-) = 0$ when $b = 0$ and $\alpha_i(b) = 1$ when $b = \max B$.

In particular, this implies that bidding strategies are left-continuous (which is without loss of generality given our value distributions), as shown in Figure 6.1.

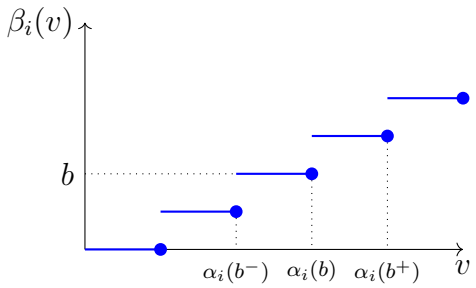


Figure 6.1: A monotone bidding strategy $\beta_i(\cdot)$ can be succinctly represented by its jump points, $\alpha_i(b)$ for $b \in B$.

Irrational Equilibria. As mentioned earlier, for our PPAD-completeness result, we will be looking for an ε-approximate equilibrium, rather than an exact one. Of course, this only makes our hardness results even stronger; but besides that, it is actually very much necessary for our membership result in PPAD as well. In particular, as demonstrated by the example below, the FPA can have *only irrational* equilibria, even when all input parameters are rational numbers.

Example 6.1. Consider a FPA with $n = 3$ bidders and common priors, whose values are independently and identically distributed according to the uniform distribution on $[0, 1]$; that is, $F_i(x) = x$ for $i = 1, 2, 3$. Let the bidding space be $B = \{0, 1/2\}$. Clearly, this auction can be represented with piecewise-constant density functions (with a single piece) and with a finite number of rational quantities. It can be verified that the auction has a unique equilibrium, where a bidder bids 0 if and only if her valuation is below $\frac{-1+\sqrt{5}}{2} \approx 0.618$; therefore, the unique equilibrium is irrational. A detailed derivation can be found in [FGH+21, Appendix C].

6.2 Equilibrium Characterisation and Best Response Computation

In this section we begin by presenting a useful characterisation of ε-BNE that will be useful throughout the chapter. Then, we show how best-responses of bidders can be checked and computed in polynomial time. We remark that the reductions that we will construct in Section 6.3 to show the PPAD-membership and the FIXP-membership of the problem do not technically require the computation of the whole best-response function, but rather only the probabilities of winning the item given the bidder's bid and the bidding strategies of the other bidders. However, the best-response computation is interesting in its own right, and that is why we present this here.

Characterisation. The following lemma essentially states that an ε-BNE is characterised by the behavior of the bidding function at the jump points. Recall that for any bid b, we let b^- denote the previous bid, and we use the conventions $\alpha_i(b^-) = 0$ when $b = 0$, and $\alpha_i(b) = 1$ when $b = \max B$.

Lemma 6.1 (Characterisation of ε-BNE)**.** *Fix an $\varepsilon \geq 0$. A strategy profile $\boldsymbol{\beta}$ is an ε-BNE of the FPA, if and only if, for every bidder i and every bid b with $\alpha_i(b^-) < \alpha_i(b)$,*

$$u_i(b, \boldsymbol{\beta}_{-i}; \alpha_i(b^-)) \geq u_i(b', \boldsymbol{\beta}_{-i}; \alpha_i(b^-)) - \varepsilon \qquad \text{for all } b' < b \tag{6.3}$$

and

$$u_i(b, \boldsymbol{\beta}_{-i}; \alpha_i(b)) \geq u_i(b', \boldsymbol{\beta}_{-i}; \alpha_i(b)) - \varepsilon \qquad \text{for all } b' > b. \tag{6.4}$$

The H-functions. Before proving this characterisation, we introduce some useful notation. We use the term $H_i(b, \boldsymbol{\beta}_{-i})$ to denote the (perceived) probability that bidder i wins the item with bid b, when the other bidders use bids according to the bidding strategy $\boldsymbol{\beta}_{-i}$, i.e.,

$$H_i(b, \boldsymbol{\beta}_{-i}) = \mathbb{P}\left[\text{bidder } i \text{ wins} | b, \boldsymbol{\beta}_{-i}\right] \tag{6.5}$$

The utility can easily be expressed in terms of this function, namely $u_i(b, \boldsymbol{\beta}_{-i}; v_i) = (v_i - b) \cdot H_i(b, \boldsymbol{\beta}_{-i})$.

Proof of Lemma 6.1. ($\Rightarrow$): Fix a bidder i and a bid b with $\alpha_i(b^-) < \alpha_i(b)$. Since bidder i bids b inside the non-empty interval $(\alpha_i(b^-), \alpha_i(b)]$, and $\boldsymbol{\beta}$ is an ε-BNE, we get that $u_i(b, \boldsymbol{\beta}_{-i}; v_i) \geq u_i(b', \boldsymbol{\beta}_{-i}; v_i) - \varepsilon$ for every $v_i \in (\alpha_i(b^-), \alpha_i(b)]$ and $b' \neq b$. Since the utilities are continuous functions on v_i, the inequalities must also hold at the interval endpoints.

($\Leftarrow$): Suppose (6.3, 6.4) hold. Take any bidder i and any valuation v_i, and let $(\alpha_i(b^-), \alpha_i(b)]$ be the interval containing v_i. Notice that the utilities $u_i(b, \boldsymbol{\beta}_{-i}; v_i)$, $u_i(b', \boldsymbol{\beta}_{-i}; v_i)$ are linear functions on v_i, with slopes given by $H_i(b, \boldsymbol{\beta}_{-i})$, $H_i(b', \boldsymbol{\beta}_{-i})$ respectively. For $b' < b$, we know that $H_i(b', \boldsymbol{\beta}_{-i}) \leq H_i(b, \boldsymbol{\beta}_{-i})$ and $u_i(b, \boldsymbol{\beta}_{-i}; v) \geq u_i(b', \boldsymbol{\beta}_{-i}; v) - \varepsilon$ holds at $v = \alpha_i(b^-)$; therefore it must hold also at $v = v_i$. Similarly for $b' > b$, we know that $H_i(b', \boldsymbol{\beta}_{-i}) \geq H_i(b, \boldsymbol{\beta}_{-i})$ and $u_i(b, \boldsymbol{\beta}_{-i}; v) \geq u_i(b', \boldsymbol{\beta}_{-i}; v) - \varepsilon$ holds at $v = \alpha_i(b)$; therefore it must hold also at $v = v_i$. We thus conclude that $\boldsymbol{\beta}$ is an ε-BNE. □

We now consider the basic computational problems of checking and computing best-responses of bidders. We assume throughout that bidding strategies provided in the input are given via rational quantities corresponding to the jump points $\alpha_j(b)$, as defined in Section 6.1.1. The first step to be able to check or compute best-responses is the efficient computation of the H-functions defined above.

Computation of the H-functions. The probability appearing in (6.5) clearly depends on bidder i's prior on the other bidders' distributions, as well as on whether b is the highest bid, and if it is, how many other highest bids there are in the auction, in case of a tie. While the form of the functions H_i can be devised analytically, the expression involves exponentially many terms in the number of bidders n; therefore it is not obvious that it can be computed efficiently. The following lemma states that this is in fact possible.

Lemma 6.2. *Given a bidder i, a bid b and bidding strategies $\boldsymbol{\beta}_{-i}$ of the other bidders, the probability $H_i(b, \boldsymbol{\beta}_{-i})$ of bidder i winning the item can be computed in polynomial time.*

Proof. For ease of notation, we present the proof for bidder $i = n$. The cases for the other bidders are analogous and can be handled, e.g., via an appropriate relabelling. The probability that bidder n wins (given her bid and the bidding strategies of the other bidders) can be written as

$$H_n(b, \boldsymbol{\beta}_{-n}) = \sum_{k=0}^{n-1} \frac{1}{k+1} T(b, n-1, k), \tag{6.6}$$

where, for $0 \leq k \leq \ell \leq n-1$, we use $T(b, \ell, k)$ to denote the probability that *exactly* k out of the first ℓ bidders bid exactly b, and the remaining $\ell - k$ bidders all bid

below b; in other words, for the special case where $\ell = n-1$ in the above expression, $T(b, n-1, k)$ is the probability of the highest bid being b, with $k+1$ bidders (including bidder n) being tied for the highest bid. Next, for a given bidder j, let

$$G_{j,b^-} = F_{n,j}(\alpha_j(b^-)) = \mathbb{P}\left[\beta_j(v_j) < b\right], \quad g_{j,b} = F_{n,j}(\alpha_j(b)) - G_{jb^-} = \mathbb{P}\left[\beta_j(v_j) = b\right]$$

denote the (perceived from the perspective of bidder n) probabilities that bidder j bids below b, and exactly b, respectively. Note that G_{j,b^-} and $g_{j,b}$ can be efficiently computed with access to $F_{n,j}$ and $\boldsymbol{\alpha}_{-n}$. Moreover, one could write

$$T(b, n-1, k) = \sum_{\substack{S \subseteq [n-1] \\ |S|=k}} \prod_{j \in S} g_{j,b} \cdot \prod_{j \notin S} G_{j,b^-}. \tag{6.7}$$

Notice that (6.7) does not yield an efficient way of computing the probabilities, as the number of summands can be exponential in n. To bypass this obstacle, we observe that, more generally, the probabilities $T(b, \ell, k)$ can be computed from G_{ℓ,b^-} and $g_{\ell,b}$ via dynamic programming, by conditioning on bidder ℓ's bid, in the following way:

$$\begin{aligned}
T(b, 0, 0) &= 1; \\
T(b, \ell, k) &= 0, && \text{for } k > \ell; \\
T(b, \ell+1, 0) &= T(b, \ell, 0) G_{\ell+1,b^-}; \\
T(b, \ell+1, k+1) &= T(b, \ell, k) g_{\ell+1,b} + T(b, \ell, k+1) G_{\ell+1,b^-}; && \text{for } k \leq \ell.
\end{aligned}$$

Thus, all values of $T(b, n-1, k)$, for $k = 0, \ldots, n-1$, can be computed with a total number of $O(n^2)$ recursive calls, so that $H_n(b, \boldsymbol{\beta}_{-n})$ can be computed in polynomial time. □

Lemma 6.2 implies that the utilities in (6.3, 6.4) of the characterisation (Lemma 6.1) can be computed in polynomial time. Since there are $O(n|B|^2)$ inequalities to check in Lemma 6.1, we immediately conclude the following.

Corollary 6.1. *Given $\varepsilon \geq 0$, and a strategy profile $\boldsymbol{\beta}$ in a first-price auction with subjective priors, one can determine in polynomial time if $\boldsymbol{\beta}$ constitutes an ε-BNE.*

Using Lemma 6.2, we can now also efficiently compute best-responses, and, in fact, even *exact* best-responses (i.e., ε-best-responses for $\varepsilon = 0$).

Theorem 6.1. *In a first-price auction with subjective priors, the bidders' best-responses can be computed in polynomial time.*

Proof. Given a bidder i and the vector of bidding strategies $\boldsymbol{\beta}_{-i}$, one can compute in polynomial time the probabilities $H_i(b, \boldsymbol{\beta}_{-i})$ for each bid $b \in B$ using Lemma 6.2. Now recall that the utility of bidder i, when having a valuation of v_i and bidding b, is given by $u_i(b, \boldsymbol{\beta}_{-i}; v_i) = (v_i - b) \cdot H_i(b, \boldsymbol{\beta}_{-i})$, which is a linear function on v_i having slope $H_i(b, \boldsymbol{\beta}_{-i})$. Thus, maximizing the utility amounts to taking the maximum (or *upper envelope*) of $|B|$ linear functions; the result is a piecewise linear function whose jump points can be efficiently computed by solving linear equations. In particular, given bids $b < b'$, we can compute $\alpha = \tilde{\alpha}_i(b, b')$ as the solution of $u_i(b, \boldsymbol{\beta}_{-i}; \alpha) = u_i(b', \boldsymbol{\beta}_{-i}; \alpha)$, that is,

$$\tilde{\alpha}_i(b, b') = \begin{cases} \frac{b' H_i(b', \boldsymbol{\beta}_{-i}) - b H_i(b, \boldsymbol{\beta}_{-i})}{H_i(b', \boldsymbol{\beta}_{-i}) - H_i(b, \boldsymbol{\beta}_{-i})} & \text{if} \quad H_i(b', \boldsymbol{\beta}_{-i}) \neq H_i(b, \boldsymbol{\beta}_{-i}), \\ +\infty & \text{otherwise.} \end{cases}$$

Intuitively, $\tilde{\alpha}_i(b, b')$ is the jump point corresponding to bidding b versus bidding b': bidder i achieves higher utility by bidding b iff $v_i < \tilde{\alpha}_i(b, b')$. Now the highest value for which bidder i (weakly) prefers bidding b versus any other higher bid is $\min_{b' > b} \tilde{\alpha}_i(b, b')$; if at this valuation, bidding b also achieves higher utility than bidding any other lower bid, then $\min_{b' > b} \tilde{\alpha}_i(b, b')$ is indeed one of the desired jump points. Otherwise, b is a degenerate bid, in the sense that there is no valuation for which b is an optimal response. Therefore, the jump points introduced in (6.2) are given by $\alpha_i(b) = \max_{b' \leq b} \min_{b'' > b'} \tilde{\alpha}_i(b', b'')$.[4] Clearly then, the $\alpha_i(b)$ can be found in polynomial time. □

6.3 Existence and Membership in PPAD and FIXP

The existence of equilibria in our setting can essentially be established by adapting a proof by Athey [Ath01], which relies on Kakutani's fixed point theorem. Unfortunately, proofs that are based on this fixed point theorem cannot easily be turned into membership results for computational classes such as PPAD and FIXP. This is especially true for FIXP which is defined as the class of all problems that can be solved by finding a Brouwer fixed point. In order to circumvent this obstacle we present a new proof that uses Brouwer's fixed point theorem. In this section, we first present this proof, and then utilise it to prove membership of our problems of interest in PPAD and FIXP.

[4] The maximisation over $b' \leq b$ serves to exclude degenerate cases, e.g., if $b' < b < b''$ but $\tilde{\alpha}_i(b, b'') < \tilde{\alpha}_i(b', b'') < \tilde{\alpha}_i(b, b')$.

6.3.1 Existence of Equilibria via Brouwer's Fixed Point Theorem

Theorem 6.2. *Every first-price auction with continuous subjective priors and finite bidding space admits a monotone non-decreasing and non-overbidding pure Bayes-Nash equilibrium.*

Proof. Let $N = \{1, 2, \dots, n\}$ be the set of bidders, $F_{i,j}$ the continuous subjective priors, and $0 = b_0, b_1, \dots, b_m$ be the ordered list of bids, i.e., the elements of $B \subseteq [0, 1]$. Recall that a monotone non-decreasing strategy $\beta_i : [0, 1] \to B$ can be represented by its jump points $\alpha_i(b)$. Let

$$\mathcal{D} = \{\boldsymbol{\alpha} \in ([0, 1]^m)^n \mid \forall i \in N, j \in [m] : \alpha_i(b_{j-2}) \leq \alpha_i(b_{j-1}) \wedge b_j \leq \alpha_i(b_{j-1})\}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_n)$ and we use the convention $\alpha_i(b_{-1}) := 0$ to keep the notation simple. The domain $\mathcal{D}$ is the set of all monotone non-decreasing non-overbidding strategy profiles, represented by their jump points. Note that $\mathcal{D}$ is compact and convex.

In what follows we slightly abuse notation by replacing the strategy profile $\boldsymbol{\beta}$ by its representation $\boldsymbol{\alpha}$ in some terms. Recall the functions $H_i(b, \boldsymbol{\alpha}_{-i})$ defined in Section 6.2, which represent the probability that bidder i wins the auction, if they bid b. By inspecting the proof of Lemma 6.2, it is easy to see that the quantities G_{jb^-} and g_{jb} are continuous with respect to $\boldsymbol{\alpha}_{-i}$, since the distributions are continuous. As a result, the terms $T(b, n-1, j)$ are also continuous in $\boldsymbol{\alpha}_{-i}$ (by (6.7)), which implies that $H_i(b, \boldsymbol{\beta}_{-i})$ is also continuous in $\boldsymbol{\alpha}_{-i}$. Since the utility functions can be written as $u_i(b, \boldsymbol{\alpha}_{-i}; v_i) = (v_i - b) \cdot H_i(b, \boldsymbol{\alpha}_{-i})$, it follows that the functions $(\boldsymbol{\alpha}_{-i}, v_i) \mapsto u_i(b, \boldsymbol{\alpha}_{-i}; v_i)$ are continuous.

We now construct a function $G : \mathcal{D} \to \mathcal{D}$. For any bidder $i \in N$ and any $j \in [m]$, define the continuous function $\Delta_j^i : \mathcal{D} \to \mathbb{R}$ by

$$\Delta_j^i(\boldsymbol{\alpha}) = u_i(b_{j-1}, \boldsymbol{\alpha}_{-i}; \alpha_i(b_{j-1})) - \max_{\ell \geq j} u_i(b_\ell, \boldsymbol{\alpha}_{-i}; \alpha_i(b_{j-1})).$$

Now, for any $\boldsymbol{\alpha} \in \mathcal{D}$, let $G(\boldsymbol{\alpha}) = \boldsymbol{\alpha}'$, where for all $i \in N$ and $j = 1, 2, \dots, m$ (consecutively and in that order)

$$\alpha_i'(b_{j-1}) = \mathrm{T}_{[\max\{b_j, \alpha_i'(b_{j-2})\}, 1]}(\alpha_i(b_{j-1}) + \Delta_j^i(\boldsymbol{\alpha})). \tag{6.8}$$

Note in particular that this is well-defined, since $\alpha_i'(b_{j-2})$ is defined before $\alpha_i'(b_{j-1})$. The truncation operator immediately ensures that $\boldsymbol{\alpha}' \in \mathcal{D}$. Since G is also clearly continuous, and $\mathcal{D}$ is compact and convex, it follows by Brouwer's fixed point theorem that there exists a $\boldsymbol{\alpha} \in \mathcal{D}$ with $G(\boldsymbol{\alpha}) = \boldsymbol{\alpha}$. It remains to prove that $\boldsymbol{\alpha}$ corresponds to an equilibrium of the auction.

Consider some bidder $i \in N$. We will show that α_i is a best-response to $\boldsymbol{\alpha}_{-i}$ using the characterisation of Lemma 6.1. Consider any non-empty interval of non-empty interior $[\alpha_i(b_{j-1}), \alpha_i(b_j)]$, for some $j \in \{0, 1, \ldots, m\}$, where we use the convention that $\alpha_i(b_{-1}) = 0$ and $\alpha_i(b_m) = 1$.

- First, we show that $u_i(b_j, \boldsymbol{\alpha}_{-i}; \alpha_i(b_j)) \geq \max_{\ell > j} u_i(b_\ell, \boldsymbol{\alpha}_{-i}; \alpha_i(b_j))$. Clearly, for $j = m$ this holds trivially. For $j < m$, this can immediately be rephrased as showing $\Delta^i_{j+1}(\boldsymbol{\alpha}) \geq 0$. Now, note that by assumption we have $\alpha_i(b_j) > \alpha_i(b_{j-1})$. Thus, since $\alpha_i(b_j)$ remains fixed under G, it must be that $\alpha_i(b_j) = b_{j+1}$ or $\Delta^i_{j+1}(\boldsymbol{\alpha}) \geq 0$. However, if $\alpha_i(b_j) = b_{j+1}$, then it also trivially holds that $\Delta^i_{j+1}(\boldsymbol{\alpha}) \geq 0$.

- Next, we show that $u_i(b_j, \boldsymbol{\alpha}_{-i}; \alpha_i(b_{j-1})) \geq \max_{\ell < j} u_i(b_\ell, \boldsymbol{\alpha}_{-i}; \alpha_i(b_{j-1}))$. Again, this holds trivially for $j = 0$, so we now consider $j > 0$. By the first bullet above, it holds that

  $$u_i(b_j, \boldsymbol{\alpha}_{-i}; \alpha_i(b_j)) = \max_{\ell \geq j} u_i(b_\ell, \boldsymbol{\alpha}_{-i}; \alpha_i(b_j)).$$

  As a result, by the monotonicity of the H-functions (see the proof of Lemma 6.1), this continues to hold if we replace $\alpha_i(b_j)$ by $\alpha_i(b_{j-1})$, i.e.,

  $$u_i(b_j, \boldsymbol{\alpha}_{-i}; \alpha_i(b_{j-1})) = \max_{\ell \geq j} u_i(b_\ell, \boldsymbol{\alpha}_{-i}; \alpha_i(b_{j-1})).$$

  On the other hand, since $\alpha_i(b_{j-1}) < \alpha_i(b_j)$, it follows in particular that $\alpha_i(b_k) < 1$ for all $k < j$. As a result, since $\alpha_i(b_k)$ remains fixed under G, it must be that $\Delta^i_{k+1}(\boldsymbol{\alpha}) \leq 0$ for all $k < j$, i.e.,

  $$u_i(b_k, \boldsymbol{\alpha}_{-i}; \alpha_i(b_k)) \leq \max_{\ell \geq k+1} u_i(b_\ell, \boldsymbol{\alpha}_{-i}; \alpha_i(b_k))$$

  which by monotonicity of the H-functions (see the proof of Lemma 6.1), continues to hold if we replace $\alpha_i(b_k)$ by $\alpha_i(b_{j-1})$, i.e., for all $k < j$ we have

  $$u_i(b_k, \boldsymbol{\alpha}_{-i}; \alpha_i(b_{j-1})) \leq \max_{\ell \geq k+1} u_i(b_\ell, \boldsymbol{\alpha}_{-i}; \alpha_i(b_{j-1})).$$

  As a result it follows by induction that for all $k < j$

  $$u_i(b_k, \boldsymbol{\alpha}_{-i}; \alpha_i(b_{j-1})) \leq \max_{\ell \geq j} u_i(b_\ell, \boldsymbol{\alpha}_{-i}; \alpha_i(b_{j-1})) = u_i(b_j, \boldsymbol{\alpha}_{-i}; \alpha_i(b_{j-1})).$$

By Lemma 6.1, it immediately follows that α_i is a best-response to $\boldsymbol{\alpha}_{-i}$. Since this holds for all bidders $i \in N$, $\boldsymbol{\alpha}$ is an equilibrium. □

6.3.2 FIXP Membership

In order to study the exact equilibrium problem for the first-price auction in the context of FIXP, we consider the model where the distributions $F_{i,j}$ are given by algebraic circuits using the operations $\{+, -, \times, /, \max, \min, \sqrt[k]{\cdot}\}$ and rational constants,[5] as is usual in this setting (see Section 2.4). We show that the proof of existence in the previous section can be turned into a reduction.

Theorem 6.3. *The problem* EXACT-BNE-FPA *lies in* FIXP.

Proof. Clearly, the domain $\mathcal{D}$ of the function $G : \mathcal{D} \to \mathcal{D}$ from the proof of Theorem 6.2 can be represented by a set of linear inequalities that can be constructed in polynomial time in n, m and the representation length of B. Thus, it remains to show that we can construct in polynomial time an algebraic circuit that computes G.

We now describe how to construct a circuit for G that only uses operations $\{+, -, \times, /, \max, \min, \sqrt[k]{\cdot}\}$ and rational constants. First of all, note that probabilities of the form $\mathbb{P}_{v_j \sim F_{i,j}}[\beta_j(v_j) \leq b] = F_{i,j}(\alpha_j(b))$ can easily be computed by the circuit, since the (cumulative) distribution functions $F_{i,j}$ are provided as algebraic circuits, and $\boldsymbol{\alpha}$ is the input to the circuit. It follows that the quantities G_{jb^-} and g_{jb} defined in the proof of Lemma 6.2 can also be computed by the circuit. As a result, we can use the dynamic programming procedure described in the proof of Lemma 6.2, to compute the terms $T(b, n-1, j)$ by only using a polynomial number of operations. Note in particular, that the dynamic programming assignment rules can all be implemented using the available set of operations. With the terms $T(b, n-1, j)$ in hand, we can then easily compute the terms $H_i(b, \boldsymbol{\alpha}_{-i})$ for all $b \in B$, and thus evaluate the utility function $u_i(b, \boldsymbol{\alpha}_{-i}; v_i) = (v_i - b) \cdot H_i(b, \boldsymbol{\alpha}_{-i})$ at any given $v_i \in [0, 1]$. Finally, using the utility functions and the max operation we can now compute the terms $\Delta_j^i(\boldsymbol{\alpha})$ from the proof of Theorem 6.2, and then using $+$, max, min and the constant 1 we can output $\boldsymbol{\alpha}' = G(\boldsymbol{\alpha})$ by noting that

$$\alpha_i'(b_{j-1}) = \max\{\max\{b_j, \alpha_i'(b_{j-2})\}, \min\{1, \alpha_i(b_{j-1}) + g_j^i(\boldsymbol{\alpha})\}\}. \qquad \square$$

6.3.3 PPAD Membership

In order to study the approximate equilibrium problem for the first-price auction in the context of PPAD, we consider a model where the distributions $F_{i,j}$ are polynomially computable and polynomially continuous, as defined in Section 2.2.3. In this section we show that in this model, the problem of computing an ε-BNE lies in the class PPAD.

[5] As usual in the context of FIXP, we are promised that the circuit never divides by 0 and never takes an even root of a negative number. Here, we are additionally promised that the function computed by the circuit is indeed a cumulative distribution function.

Formally, let $\mathcal{F}$ be a polynomially computable and polynomially continuous class of cumulative distribution functions on the interval $[0,1]$. In other words, for any $F \in \mathcal{F}$ and any $x \in [0,1]$, $F(x)$ is the probability of the interval $[0,x]$ according to F. Note that distribution functions given by piecewise-constant density functions on the interval $[0,1]$ are an example of such a class $\mathcal{F}$. In this case, the density functions are represented explicitly, i.e., as a list of "blocks," where for every block we give the sub-interval of $[0,1]$ that it occupies and the height of the block.

We begin by observing that the polynomial continuity of the distribution functions $F_{i,j}$ implies that the utility functions are also polynomially continuous.

Lemma 6.3. *If the distributions $F_{i,j}$ are polynomially continuous, then so are the utility functions $\boldsymbol{\alpha} \mapsto u_i(b, \boldsymbol{\alpha}_{-i}; v_i)$. In more detail, given $\varepsilon > 0$, we can in polynomial time compute $\delta > 0$ such that for all $i \in N$, $b \in B$ and $v_i \in [0,1]$*

$$\|\boldsymbol{\alpha} - \boldsymbol{\alpha}'\|_\infty \leq \delta \implies |u_i(b, \boldsymbol{\alpha}_{-i}; v_i) - u_i(b, \boldsymbol{\alpha}'_{-i}; v_i)| \leq \varepsilon.$$

In particular, δ can be represented using a polynomial number of bits.

Proof. Since the distributions are polynomially continuous, it follows that given any $\varepsilon > 0$, we can compute $\delta > 0$ in polynomial time such that $|F_{i,j}(x) - F_{i,j}(y)| \leq \varepsilon/2^{n+1}$ for all x, y with $|x - y| \leq \delta$ and all $i, j \in N$ $(i \neq j)$.

Consider any $\boldsymbol{\alpha}, \boldsymbol{\alpha}' \in \mathcal{D}$ (see the proof of Theorem 6.2) with $\|\boldsymbol{\alpha} - \boldsymbol{\alpha}'\|_\infty \leq \delta$. Then, we have

$$\begin{aligned}\left| \mathop{\mathbb{P}}_{v_i \sim F_{j,i}} [\beta_i(v_i) \leq b] - \mathop{\mathbb{P}}_{v_i \sim F_{j,i}} [\beta'_i(v_i) \leq b] \right| &\leq \left| \mathop{\mathbb{P}}_{v_i \sim F_{j,i}} [v_i \leq \alpha_i(b)] - \mathop{\mathbb{P}}_{v_i \sim F_{j,i}} [v_i \leq \alpha'_i(b)] \right| \\ &\leq |F_{j,i}(\alpha_i(b)) - F_{j,i}(\alpha'_i(b))| \\ &\leq \varepsilon/2^{n+1}\end{aligned}$$

for all $i, j \in N$ $(i \neq j)$ and $b \in B$. It follows that $\mathbb{P}_{v_i \sim F_{j,i}} [\beta_i(v_i) < b]$ differs from $\mathbb{P}_{v_i \sim F_{j,i}} [\beta'_i(v_i) < b]$ by at most $\varepsilon/2^{n+1}$. Similarly, $\mathbb{P}_{v_i \sim F_{j,i}} [\beta_i(v_i) = b]$ differs from $\mathbb{P}_{v_i \sim F_{j,i}} [\beta'_i(v_i) = b]$ by at most $\varepsilon/2^n$.

Let $T_i(b, \ell; \boldsymbol{\alpha}_{-i})$ denote the probability that, from the perspective of bidder i, exactly ℓ out of the bidders $N \setminus \{i\}$ bid exactly b, and the remaining $n - 1 - \ell$ bidders bid below b. We can write

$$T_i(b, \ell; \boldsymbol{\alpha}_{-i}) = \sum_{\substack{S \subseteq N \setminus \{i\} \\ |S| = \ell}} \prod_{k \in S} \mathop{\mathbb{P}}_{v_k \sim F_{i,k}} [\beta_k(v_k) = b] \prod_{k \in N \setminus (\{i\} \cup S)} \mathop{\mathbb{P}}_{v_k \sim F_{i,k}} [\beta_k(v_k) < b] .$$

From this it follows that $T_i(b, \ell; \boldsymbol{\alpha}_{-i})$ and $T_i(b, \ell; \boldsymbol{\alpha}'_{-i})$ differ by at most $\binom{n-1}{\ell} n\varepsilon/2^n$, for all $i \in N$, $b \in B$ and $\ell \in \{0, 1, \ldots, n-1\}$. As defined in Section 6.2, recall

that $H_i(b, \boldsymbol{\alpha}_{-i})$ denotes the probability that bidder i wins if she bids b and the other bidders bid according to $\boldsymbol{\alpha}_{-i}$. Then, we can write

$$H_i(b, \boldsymbol{\alpha}_{-i}) = \sum_{\ell=0}^{n-1} \frac{1}{\ell+1} T_i(b, \ell; \boldsymbol{\alpha}_{-i}).$$

It follows that $H_i(b, \boldsymbol{\alpha}_{-i})$ differs from $H_i(b, \boldsymbol{\alpha}'_{-i})$ by at most

$$\sum_{\ell=0}^{n-1} \frac{1}{\ell+1} \binom{n-1}{\ell} n\varepsilon/2^n = \sum_{\ell=0}^{n-1} \binom{n}{\ell+1} \varepsilon/2^n \leq \varepsilon$$

for all $i \in N$ and $b \in B$. Finally, note that $u_i(b, \boldsymbol{\alpha}_{-i}; v_i) = H_i(b, \boldsymbol{\alpha}_{-i}) \cdot (v_i - b)$. Thus, we obtain

$$\left|u_i(b, \boldsymbol{\alpha}_{-i}; v_i) - u_i(b, \boldsymbol{\alpha}'_{-i}; v_i)\right| \leq \left|H_i(b, \boldsymbol{\alpha}_{-i}) - H_i(b, \boldsymbol{\alpha}'_{-i})\right| |v_i - b| \leq \varepsilon$$

for all $i \in N$, $b \in B$ and $v_i \in [0, 1]$, since $|v_i - b| \leq 1$. □

We are now ready to state and prove the main result of this section.

Theorem 6.4. *The problem* ε-BNE-FPA *lies in* PPAD.

Proof. We show that the existence proof of Theorem 6.2 can be turned into a polynomial-time many-one reduction to the problem of computing an approximate Brouwer fixed point of a polynomially computable and polynomially continuous function over a bounded polytope given by linear inequalities, known to lie in PPAD [EY10, Proposition 2].

Since the distributions $F_{i,j}$ are polynomially computable, and by the arguments provided in the proof of Theorem 6.3 (including the dynamic programming procedure from Lemma 6.2), it immediately follows that G is polynomially computable. The polynomial continuity of G also immediately follows from the polynomial continuity of the utility functions (Lemma 6.3). Thus, the problem of computing an approximate fixed point of G lies in PPAD.

Given $\varepsilon > 0$, by Lemma 6.3 we can compute $\delta > 0$ so that for all $i \in N$, $b \in B$ and $v_i \in [0, 1]$

$$\|\boldsymbol{\alpha} - \boldsymbol{\alpha}'\|_\infty \leq \delta \implies |u_i(b, \boldsymbol{\alpha}_{-i}; v_i) - u_i(b, \boldsymbol{\alpha}'_{-i}; v_i)| \leq \frac{\varepsilon}{16m}.$$

Now consider any δ-approximate fixed point of G, i.e., $\boldsymbol{\alpha} \in \mathcal{D}$ such that $\|G(\boldsymbol{\alpha}) - \boldsymbol{\alpha}\|_\infty \leq \delta$. Let $\boldsymbol{\alpha}' = G(\boldsymbol{\alpha})$. We prove that $\boldsymbol{\alpha}'$ is an ε-approximate equilibrium of the first-price auction. This shows that ε-BNE-FPA reduces to the Brouwer fixed point computation problem, and thus lies in PPAD.

Since $\boldsymbol{\alpha}' = G(\boldsymbol{\alpha})$ and $\|G(\boldsymbol{\alpha}) - \boldsymbol{\alpha}\|_\infty \leq \delta$, it holds that $\|\boldsymbol{\alpha} - \boldsymbol{\alpha}'\|_\infty \leq \delta$ and thus

$$|u_i(b, \boldsymbol{\alpha}_{-i}; v_i) - u_i(b, \boldsymbol{\alpha}'_{-i}; v_i)| \leq \frac{\varepsilon}{16m} \tag{6.9}$$

for all $i \in N$, $b \in B$ and $v_i \in [0,1]$. In particular, we also have that $|\Delta_j^i(\boldsymbol{\alpha}) - \Delta_j^i(\boldsymbol{\alpha}')| \leq 2(\varepsilon/16m + \delta) \leq \varepsilon/4m$ (since the utility functions are also 1-Lipschitz with respect to v_i). Note that here we assumed without loss of generality that $\delta \leq \varepsilon/16m$.

Fix some bidder $i \in N$. Consider any non-empty interval $[\alpha'_i(b_{j-1}), \alpha'_i(b_j)]$ for some $j \in \{0, 1, \ldots, m\}$, where we use the convention that $\alpha'_i(b_{-1}) = 0$ and $\alpha'_i(b_m) = 1$.

- First, we show that $u_i(b_j, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_j)) \geq \max_{\ell > j} u_i(b_\ell, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_j)) - \varepsilon/2$. Clearly, for $j = m$ this holds trivially. For $j < m$, this can immediately be rephrased as showing $\Delta_{j+1}^i(\boldsymbol{\alpha}') \geq -\varepsilon/2$. By (6.9), it suffices to show that $\Delta_{j+1}^i(\boldsymbol{\alpha}) \geq -\varepsilon/2 + \varepsilon/4m$. But if $\Delta_{j+1}^i(\boldsymbol{\alpha}) < -\varepsilon/2 + \varepsilon/4m \leq -\varepsilon/16m \leq -\delta$, then by construction of G, since $|\alpha_i(b_j) - \alpha'_i(b_j)| \leq \delta$, it must be that $\alpha'_i(b_j) = b_{j-1}$ or $\alpha'_i(b_j) = \alpha'_i(b_{j-1})$. In the former case, it trivially holds that $\Delta_{j+1}^i(\boldsymbol{\alpha}') \geq 0 \geq -\varepsilon$. The latter case is impossible, since we assumed that $\alpha'_i(b_{j-1}) < \alpha'_i(b_j)$.

- Next, we show that $u_i(b_j, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_{j-1})) \geq \max_{\ell < j} u_i(b_\ell, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_{j-1})) - \varepsilon$. Again, this holds trivially for $j = 0$, so we now consider $j > 0$. By the first bullet above, it holds that

  $$u_i(b_j, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_j)) \geq \max_{\ell \geq j} u_i(b_\ell, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_j)) - \varepsilon/2.$$

  As a result, by the monotonicity of the H-functions (see the proof of Lemma 6.1), this continues to hold if we replace $\alpha'_i(b_j)$ by $\alpha'_i(b_{j-1})$, i.e.,

  $$u_i(b_j, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_{j-1})) \geq \max_{\ell \geq j} u_i(b_\ell, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_{j-1})) - \varepsilon/2. \tag{6.10}$$

  On the other hand, since $\alpha'_i(b_{j-1}) < \alpha'_i(b_j)$, it follows in particular that $\alpha'_i(b_k) < 1$ for all $k < j$. As a result, by construction of G, and since $|\alpha_i(b_k) - \alpha'_i(b_k)| \leq \delta$, it must be that $\Delta_{k+1}^i(\boldsymbol{\alpha}) \leq \delta$ for all $k < j$. By (6.9) it follows that $\Delta_{k+1}^i(\boldsymbol{\alpha}') \leq \delta + \varepsilon/4m \leq \varepsilon/2m$ for all $k < j$, which yields

  $$u_i(b_k, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_k)) \leq \max_{\ell \geq k+1} u_i(b_\ell, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_k)) + \frac{\varepsilon}{2m}$$

  which by monotonicity of the H-functions (see the proof of Lemma 6.1), continues to hold if we replace $\alpha'_i(b_k)$ by $\alpha'_i(b_{j-1})$, i.e., for all $k < j$ we have

  $$u_i(b_k, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_{j-1})) \leq \max_{\ell \geq k+1} u_i(b_\ell, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_{j-1})) + \frac{\varepsilon}{2m}.$$

  As a result it follows by induction that for all $k < j$

  $$u_i(b_k, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_{j-1})) \leq \max_{\ell \geq j} u_i(b_\ell, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_{j-1})) + (j-k)\frac{\varepsilon}{2m}$$

  which together with (6.10) yields that for all $k < j$

  $$u_i(b_k, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_{j-1})) \leq u_i(b_j, \boldsymbol{\alpha}'_{-i}; \alpha'_i(b_{j-1})) + m\frac{\varepsilon}{2m} + \frac{\varepsilon}{2}.$$

By Lemma 6.1, it immediately follows that α'_i is a ε-best-response to $\boldsymbol{\alpha}'_{-i}$. Since this holds for all bidders $i \in N$, $\boldsymbol{\alpha}'$ is an ε-equilibrium. □

6.4 PPAD- and FIXP-hardness

In this section we prove computational hardness results for the problem of computing an equilibrium of a first-price auction with subjective priors. Namely, we show that computing an ε-BNE is PPAD-hard, while computing an exact BNE is FIXP-hard. Our computational hardness results are particularly robust, because they hold even if we apply all of the following restrictions:

- the bidding space is $B = \{0, 1/5, 2/5, 3/5, 4/5\}$,
- the value distributions $F_{i,j}$ are given by very simple piecewise constant density functions,
- ε is some sufficiently small *constant*. *(only relevant for ε-BNE)*

In particular, by a simple rescaling argument, the hardness results also hold when the bidding space consists of all monetary amounts that are increments of some fixed denomination (e.g., one cent) up to some number m.[6] For example, there exists a sufficiently small constant ε such that it is PPAD-hard to compute an ε-BNE when the bidding space is $B = \{0, 1/100, 2/100, \ldots, 99/100, 1, 101/100, \ldots, m - 1/100, m\}$.

Together with the corresponding membership results proved in the previous section (Theorems 6.3 and 6.4), we thus obtain the following two theorems, which are the main results of this chapter.

Theorem 6.5. *There exists a constant $\varepsilon > 0$ such that the problem* ε-BNE-FPA *is* PPAD*-complete.*

Theorem 6.6. *The problem* EXACT-BNE-FPA *is* FIXP*-complete.*

In the rest of this section, we present the proof of our hardness results. A nice feature of our proof is that we provide a *single* reduction to prove both PPAD- and FIXP-hardness, thanks to our results on the Generalised Circuit problem presented in Chapter 5.

In more detail, we present a reduction that achieves the following: given a generalised circuit (defined in Chapter 5), it constructs (in polynomial time) an instance of the first-price auction problem, such that for all $\varepsilon \in [0, 1/10^5]$, from any ε-BNE we

[6] Note that m should be provided in the input in *unary* representation. This is necessary to ensure that the bidding space has polynomial size, thus allowing efficient computation of best-responses. See the discussion in Section 6.1.1 regarding our assumption of an explicit bidding space.

can extract an 500ε-satisfying assignment for the generalised circuit. Furthermore, this "extraction" of the assignment from an ε-BNE can be done efficiently and, in fact, using a simple so-called separable linear transformation. This ensures that in the case $\varepsilon = 0$, we obtain an SL-reduction (see Section 2.4) from EXACT-GCIRCUIT, which yields the FIXP-hardness result. If we let $\tilde{\varepsilon} > 0$ be a constant such that $\tilde{\varepsilon}$-GCIRCUIT is PPAD-hard, then for $\varepsilon = \min\{1/10^5, \tilde{\varepsilon}/500\}$ the reduction is a valid polynomial-time many-one reduction, which yields the PPAD-hardness result.

An obstacle to obtaining the desired reduction is that it is unclear how to simulate a G_+-gate or a $G_\times$-gate. As a result, we reduce from the GCIRCUIT problem with gate-types $\mathcal{G} = \{G_{\times 2}, G_{1-}, G_\phi\}$, where $\phi : [0,1]^2 \to [0,1]$, $(x,y) \mapsto \frac{1}{4}(x+1)(y+1)$. This means that a gate $g_i = (G_\phi, j, k)$ enforces the constraint $\mathbf{v}[g_i] = \phi(\mathbf{v}[g_j], \mathbf{v}[g_k]) \pm \varepsilon$. In Lemma 5.1 we proved that this set of gate-types is sufficient for our desired hardness results.

The reduction. We begin with a high-level description of the reduction. Consider a generalised circuit $g_1, g_2, \ldots, g_m$ with gate-types $\mathcal{G} = \{G_{\times 2}, G_{1-}, G_\phi\}$. We construct a first-price auction with bidding space $B = \{0, 1/5, 2/5, 3/5, 4/5\}$ and a set of bidders $N = \{1, 2, \ldots, n\}$ where $n = 10m$. For every $i \in [m]$, bidder i will "correspond" to gate g_i, in the sense that, in any ε-BNE $\boldsymbol{\beta}$, the position of the second jump point of β_i, i.e., $\alpha_i(1/5)$ will encode the value $\mathbf{v}[g_i]$ that we will assign to gate g_i. Thus, we will refer to the bidders $1, 2, \ldots, m$ as *gate-bidders*. The rest of the bidders will be used as intermediate steps to enforce the desired constraints on the strategies of the gate-bidders. Accordingly, we will refer to them as *auxiliary-bidders*. Note that for every gate-bidder, there are 9 auxiliary-bidders available (if needed). For convenience, we describe the construction with the value space $[0,5]$ instead of $[0,1]$. This is without consequence, since this re-scaling of the instance simply means that we have to replace ε by 5ε at the end. Note that as a result of the re-scaling, the bidding space is now simply $B = \{0,1,2,3,4\}$.

Valid strategies and encoded value. Let $\boldsymbol{\beta}$ be any ε-BNE of the auction. A bidder $i \in N$ is said to be *valid*, if $\alpha_i(0) \in [1, 1+1/2]$, $\alpha_i(1) \in [2+1/3-2\varepsilon, 2+2/3+2\varepsilon]$, $\alpha_i(2) \in [3+1/2, 5]$ and $\alpha_i(3) = 5$. The bidder i is *almost-valid*, if the condition on $\alpha_i(1)$ is relaxed to $\alpha_i(1) \in [2,3]$. For every bidder $i \in N$, we define the value encoded by bidder i according to $\boldsymbol{\beta}$, as

$$\mathbf{v}_{\boldsymbol{\beta}}[i] = \begin{cases} \mathrm{T}_{[0,1]}(3(\alpha_i(1) - 2 - 1/3)) & \text{if } i \text{ is valid,} \\ \text{null} & \text{otherwise.} \end{cases}$$

Note that we always have $\mathbf{v}_{\boldsymbol{\beta}}[i] \in [0,1] \cup \{\text{null}\}$. In the rest of the proof, we drop the subscript $\boldsymbol{\beta}$, since it is understood from the context. Our construction will ensure that for all $i \in [m]$, bidder i is valid and as a result $\mathbf{v}[i] \in [0,1]$. Furthermore, letting $\mathbf{v}[g_i] := \mathbf{v}[i]$ will yield an 100ε-satisfying assignment of the generalised circuit.

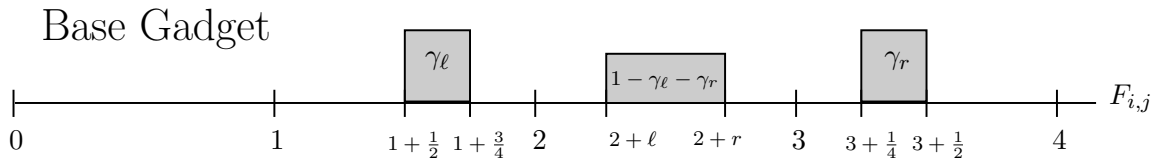


Figure 6.2: An illustration of the base gadget. The density of $F_{i,j}$ is depicted. When $\gamma_\ell = \gamma_r = 1/3$, $\ell = 1/3$ and $r = 2/3$ we obtain a *standard* base gadget, which essentially (approximately) "copies" the value $\mathbf{v}[i]$ of the input bidder i to the value $\mathbf{v}[j]$ of the output bidder j.

Gadgets. The rest of the proof describes the construction of the distribution functions $F_{i,j}$. We begin by constructing some *unary* gadgets. A unary gadget has a single "input" bidder $j \in N$ and an output bidder $i \in N \setminus \{j\}$. The goal of such a gadget is to establish a constraint on β_i that depends on β_j, but not on the strategy of any other bidder. This is achieved by setting $F_{i,k}$ for all $k \in N \setminus \{i, j\}$, such that its (piecewise constant) density function has a single piece of volume 1 lying in $[0, 1]$. As a result, because of the no-overbidding assumption, bidder i will believe that all bidders $k \in N \setminus \{i, j\}$ bid 0 with probability 1. The behavior of the gadget is then determined by the precise construction of $F_{i,j}$.

Base Gadget. The base gadget with input bidder j and output bidder i has four parameters $\gamma_\ell, \gamma_r, \ell, r \in [0, 1]$ with $\gamma_\ell + \gamma_r < 1$ and $r - \ell > 0$. The piecewise constant density function of $F_{i,j}$ is defined as follows. There is a piece of volume γ_ℓ in the interval $[1 + 1/2, 1 + 3/4]$, a piece of volume $1 - \gamma_\ell - \gamma_r$ in $[2 + \ell, 2 + r]$, and finally a piece of volume γ_r in $[3 + 1/4, 3 + 1/2]$. See Figure 6.2 for an illustration.

When the parameters are $(\gamma_\ell, \gamma_r, \ell, r) = (1/3, 1/3, 1/3, 2/3)$, we call this the *standard* base gadget. It will immediately follow from Claim 6.1 below that if the input bidder j of the standard base gadget is valid, then so is the output bidder i, and furthermore $\mathbf{v}[i] = \mathbf{v}[j] \pm 6\varepsilon$. In other words, this gadget can be used to copy the value encoded by one bidder onto some other bidder.

Claim 6.1. *Let $\gamma_\ell, \gamma_r, \ell, r \in [0, 1]$ with $\gamma_\ell, \gamma_r \geq 1/20$, $\gamma_\ell + \gamma_r < 1$ and $\ell < r$. Consider a base gadget with input bidder j and output bidder i, and parameters $(\gamma_\ell, \gamma_r, \ell, r)$. It holds that:*

- *If the input bidder j is almost-valid, then the output bidder i is also almost-valid.*
- *If $\gamma_\ell, \gamma_r \geq 1/3$ and j is almost-valid, then i is valid and*

$$\mathbf{v}[i] = (3\gamma_\ell - 1) + 3(1 - \gamma_\ell - \gamma_r)\frac{T_{[2+\ell,2+r]}(\alpha_j(1)) - (2 + \ell)}{r - \ell} \pm 6\varepsilon.$$

Proof. We begin by obtaining some equations that will be useful for various proofs in this section. Consider any unary gadget with input bidder j and output bidder i. To simplify notation, for $b \in \{0,1,2,3,4\}$, let p_b be the probability that bidder j bids b, as perceived by bidder i. Formally,

$$p_b := \mathop{\mathbb{P}}_{v_j \sim F_{i,j}}[\beta_j(v_j) = b] = \begin{cases} F_{i,j}(\alpha_j(b)) - F_{i,j}(\alpha_j(b-1)) & \text{if } b \in \{1,2,3,4\} \\ F_{i,j}(\alpha_j(0)) & \text{if } b = 0. \end{cases}$$

Recall the quantity $H_i(b, \boldsymbol{\beta}_{-i})$ defined in Section 6.2, which represents the probability that bidder i wins the auction if she bids b, and the other bidders act according to $\boldsymbol{\beta}_{-i}$. We drop $\boldsymbol{\beta}_{-i}$ from the notation, since it is clear from the context. Going back to our unary gadget, it is easy to see that $H_i(0) = p_0/n$, $H_i(1) = p_0 + p_1/2$, $H_i(2) = p_0+p_1+p_2/2$, $H_i(3) = p_0+p_1+p_2+p_3/2$ and $H_i(4) = p_0+p_1+p_2+p_3+p_4/2$. By Lemma 6.1 the first jump point $\alpha_i(0)$ of β_i must necessarily satisfy $u_i(0, \boldsymbol{\beta}_{-i}; \alpha_i(0)) \geq u_i(1, \boldsymbol{\beta}_{-i}; \alpha_i(0)) - \varepsilon$ (because the interval $(0, \alpha_i(0))$ is non-empty by the non-overbidding assumption). We can rewrite this as $H_i(0) \cdot (\alpha_i(0) - 0) \geq H_i(1) \cdot (\alpha_i(0) - 1) - \varepsilon$, which yields

$$\alpha_i(0) \leq \frac{H_i(1) + \varepsilon}{H_i(1) - H_i(0)} = 1 + \frac{H_i(0) + \varepsilon}{H_i(1) - H_i(0)} = 1 + \frac{p_0/n + \varepsilon}{p_0(n-1)/n + p_1/2} \tag{6.11}$$

where the fraction is interpreted as $+\infty$ when $p_0 + p_1 = 0$. Similarly, by Lemma 6.1, the fourth jump point must satisfy $u_i(4, \boldsymbol{\beta}_{-i}; \alpha_i(3)) \geq u_i(3, \boldsymbol{\beta}_{-i}; \alpha_i(3)) - \varepsilon$, unless $\alpha_i(3) = 5$. Rewriting this as $H_i(4) \cdot (\alpha_i(3) - 4) \geq H_i(3) \cdot (\alpha_i(3) - 3) - \varepsilon$, we obtain that $\alpha_i(3) = 5$ or

$$\alpha_i(3) \geq \frac{4H_i(4) - 3H_i(3) - \varepsilon}{H_i(4) - H_i(3)} = 4 + \frac{H_i(3) - \varepsilon}{H_i(4) - H_i(3)} = 4 + \frac{p_0 + p_1 + p_2 + p_3/2 - \varepsilon}{p_3/2 + p_4/2}. \tag{6.12}$$

By Lemma 6.1, the third jump point must satisfy $u_i(3, \boldsymbol{\beta}_{-i}; \alpha_i(2)) \geq u_i(2, \boldsymbol{\beta}_{-i}; \alpha_i(2)) - \varepsilon$, unless $\alpha_i(2) = \alpha_i(3)$, and it must satisfy $u_i(2, \boldsymbol{\beta}_{-i}; \alpha_i(2)) \geq u_i(3, \boldsymbol{\beta}_{-i}; \alpha_i(2)) - \varepsilon$, unless $\alpha_i(2) = \alpha_i(1)$. Thus it follows that

$$\alpha_i(2) = \mathrm{T}_{[\alpha_i(1), \alpha_i(3)]}\left(\frac{3H_i(3) - 2H_i(2) \pm \varepsilon}{H_i(3) - H_i(2)}\right) = \mathrm{T}_{[\alpha_i(1), \alpha_i(3)]}\left(3 + \frac{2p_0 + 2p_1 + p_2 \pm 2\varepsilon}{p_2 + p_3}\right) \tag{6.13}$$

Finally, by Lemma 6.1, the second jump point must satisfy $u_i(2, \boldsymbol{\beta}_{-i}; \alpha_i(1)) \geq u_i(1, \boldsymbol{\beta}_{-i}; \alpha_i(1)) - \varepsilon$, unless $\alpha_i(1) = \alpha_i(2)$, and it must satisfy $u_i(1, \boldsymbol{\beta}_{-i}; \alpha_i(1)) \geq u_i(2, \boldsymbol{\beta}_{-i}; \alpha_i(1)) - \varepsilon$, unless $\alpha_i(1) = \alpha_i(0)$. As a result, it must be that

$$\alpha_i(1) = \mathrm{T}_{[\alpha_i(0), \alpha_i(2)]}\left(\frac{2H_i(2) - H_i(1) \pm \varepsilon}{H_i(2) - H_i(1)}\right) = \mathrm{T}_{[\alpha_i(0), \alpha_i(2)]}\left(2 + \frac{2p_0 + p_1 \pm 2\varepsilon}{p_1 + p_2}\right) \tag{6.14}$$

We are now ready to prove Claim 6.1. Consider a base gadget with input bidder j, output bidder i and parameters $(\gamma_\ell, \gamma_r, \ell, r)$, such that $\gamma_\ell, \gamma_r \geq 1/20$, $\gamma_\ell + \gamma_r < 1$ and $\ell < r$. Let p_b denote the probability that bidder j bids b, as perceived by bidder i.

Assume first that bidder j is almost-valid. Then, by the construction of $F_{i,j}$, we obtain that $p_0 = p_3 = p_4 = 0$, $p_1 \in [\gamma_\ell, 1 - \gamma_r]$ and $p_2 = 1 - p_1$. Using (6.11) we have that $\alpha_i(0) \leq 1 + \frac{\varepsilon}{p_1/2} \leq 1 + \frac{2\varepsilon}{\gamma_\ell} \leq 1 + 1/2$ since $\gamma_\ell \geq 4\varepsilon$. Using (6.12) we obtain that $\alpha_i(3) = 5$, since $p_3 = p_4 = 0$ and $1 - \varepsilon > 0$. (6.13) yields that $\alpha_i(2) \geq 3 + \frac{1+p_1-2\varepsilon}{1-p_1} \geq 3 + 1/2$, since $\varepsilon \leq 1/4$. Thus, in order to show that bidder i is almost-valid, it remains to prove that $\alpha_i(1) \in [2, 3]$. Using (6.14) we can write

$$\alpha_i(1) = \mathrm{T}_{[\alpha_i(0),\alpha_i(2)]}\left(2 + \frac{2p_0 + p_1 \pm 2\varepsilon}{p_1 + p_2}\right) = \mathrm{T}_{[\alpha_i(0),\alpha_i(2)]}\left(2 + p_1 \pm 2\varepsilon\right) = 2 + p_1 \pm 2\varepsilon$$

where we used the fact that $p_1 + 2\varepsilon \leq 1$, since $p_1 \leq 1 - \gamma_r$ and $\gamma_r \geq 2\varepsilon$. Note that this also yields that $\alpha_i(1) \leq 3$, while the bound $\alpha_i(1) \geq 2$ holds because $p_1 \geq \gamma_\ell$ and $\gamma_\ell \geq 2\varepsilon$ (or simply because of the no-overbidding assumption). As a result, bidder i is almost-valid.

Now consider the case where, in addition, $\gamma_\ell, \gamma_r \geq 1/3$. We can write

$$p_1 = \gamma_\ell + (1 - \gamma_\ell - \gamma_r)\frac{\mathrm{T}_{[2+\ell,2+r]}(\alpha_j(1)) - (2 + \ell)}{r - \ell}.$$

In particular, it holds that $p_1 \in [1/3, 2/3]$. Since, as shown above, $\alpha_i(1) = 2 + p_1 \pm 2\varepsilon$, we immediately obtain that $\alpha_i(1) \in [2 + 1/3 - 2\varepsilon, 2 + 2/3 + 2\varepsilon]$, i.e., bidder i is valid. Furthermore, we can write

$$\begin{aligned}\mathbf{v}[i] &= \mathrm{T}_{[0,1]}(3(\alpha_i(1) - 2 - 1/3)) \\ &= 3p_1 - 1 \pm 6\varepsilon = (3\gamma_\ell - 1) + 3(1 - \gamma_\ell - \gamma_r)\frac{\mathrm{T}_{[2+\ell,2+r]}(\alpha_j(1)) - (2 + \ell)}{r - \ell} \pm 6\varepsilon\end{aligned}$$

which proves the claim. □

Projection Gadget. The projection gadget with input bidder j and output bidder i, uses two additional auxiliary-bidders k and k', and consists of three uses of the standard base gadget. Concretely, the first standard base gadget has input j and output k, the second such gadget has input k and output k', and the third has input k' and output i. See Figure 6.3a for an illustration. As stated in the claim below, the projection gadget has the notable property that the output bidder i is *always* valid. This gadget will be used to ultimately ensure that all the gate-bidders are valid.

Claim 6.2. *The projection gadget with input bidder j and output bidder i ensures that:*

- *the output bidder i is valid, and*

- *if the input bidder j is valid, then* $\mathbf{v}[i] = \mathbf{v}[j] \pm 18\varepsilon$.

Proof. The second point follows immediately from Claim 6.1 applied to the standard base gate. Thus, it remains to show that the output bidder i is always valid. Consider the first standard base gadget, which has input bidder j and output bidder k. Let p_b denote the probability that bidder j bids b, as perceived by bidder k. Since the density function of $F_{k,j}$ has a block of volume $1/3$ lying in $[1+1/2, 1+3/4]$, and since we do not allow overbidding, it follows that $p_0 + p_1 \geq 1/3$. Using (6.11) this implies that

$$\alpha_k(0) \leq 1 + \frac{p_0/n + \varepsilon}{p_0(n-1)/n + p_1/2} \leq 1 + 6/n + 6\varepsilon \leq 1 + 1/2$$

since wlog $n \geq 24$ and $\varepsilon \leq 1/24$. Next, using (6.12) we immediately get that $\alpha_k(3) \geq 4$ since $\varepsilon < 1/3$ (or just by using the no-overbidding assumption). Then, (6.13) implies that

$$\alpha_k(2) = \mathrm{T}_{[\alpha_k(1),\alpha_k(3)]}\left(3 + \frac{2p_0 + 2p_1 + p_2 \pm 2\varepsilon}{p_2 + p_3}\right) \geq 4 - 2\varepsilon \geq 3 + 1/2$$

where we used $p_0 + p_1 \geq 1/3$, $p_2 + p_3 \leq 2/3$, and $\varepsilon \leq 1/4$. Finally, note that $\alpha_k(1) \geq 2$ by the no-overbidding assumption.

Next, consider the second standard base gadget, which has input bidder k and output bidder k'. Let p_b denote the probability that bidder k bids b, as perceived by bidder k'. From the construction of the density function of $F_{k',k}$ and the bounds obtained on the jump points of k in the first step, it follows that $p_0 = p_3 = p_4 = 0$ and $p_1 \geq 1/3$. Using Equations (6.11) to (6.13) similarly to above, we obtain that $\alpha_{k'}(0) \leq 1 + 1/2$, $\alpha_{k'}(2) \geq 3 + 1/2$ and $\alpha_{k'}(3) = 5$. As before, we have that $\alpha_{k'}(1) \geq 2$ by the no-overbidding assumption, and using (6.14) we also obtain that

$$\alpha_{k'}(1) = \mathrm{T}_{[\alpha_{k'}(0),\alpha_{k'}(2)]}\left(2 + \frac{2p_0 + p_1 \pm 2\varepsilon}{p_1 + p_2}\right) \leq 2 + 1 + 2\varepsilon \leq 3 + 1/4$$

since $\varepsilon \leq 1/8$.

Finally, consider the third and last standard base gadget, which has input bidder k' and output bidder i. Let p_b denote the probability that bidder k' bids b, as perceived by bidder i. From the construction of the density function of $F_{i,k'}$ and the bounds obtained on the jump points of k' in the previous step, it follows that $p_0 = p_3 = p_4 = 0$, $p_1 \geq 1/3$ and $p_2 \geq 1/3$. Again using Equations (6.11) to (6.13) as in the previous step, we obtain that $\alpha_i(0) \leq 1 + 1/2$, $\alpha_i(2) \geq 3 + 1/2$ and $\alpha_i(3) = 5$. Using (6.14) we have that

$$\alpha_i(1) = \mathrm{T}_{[\alpha_i(0),\alpha_i(2)]}\left(2 + \frac{2p_0 + p_1 \pm 2\varepsilon}{p_1 + p_2}\right) = 2 + p_1 \pm 2\varepsilon \in [2 + 1/3 - 2\varepsilon, 2 + 2/3 + 2\varepsilon]$$

and thus bidder i is indeed valid. □

$G_{\times 2}$ Gadget. The $G_{\times 2}$ gadget with input bidder j and output bidder i, uses an additional auxiliary-bidder k, and consists of one use of the base gadget and one use of the projection gadget. In more detail, the base gadget has input j, output k and parameters $(\gamma_\ell, \gamma_r, \ell, r) = (1/3, 1/3, 1/3, 1/2)$, while the projection gate has input k and output i. See Figure 6.3b for an illustration.

Claim 6.3. *The $G_{\times 2}$ gadget with input bidder j and output bidder i ensures that:*

- *the output bidder i is valid, and*
- *if the input bidder j is valid, then $\mathbf{v}[i] = \mathrm{T}(2 \cdot \mathbf{v}[j]) \pm 24\varepsilon$.*

Proof. The fact that bidder i is valid follows from our use of the projection gadget and the first bullet point in Claim 6.2. Now consider the case where bidder j is valid. Since $\gamma_\ell = \gamma_r = 1/3$, by Claim 6.1 we know that bidder k is also valid and it holds that

$$\mathbf{v}[k] = \frac{\mathrm{T}_{[2+\ell,2+r]}(\alpha_j(1)) - (2+\ell)}{r-\ell} \pm 6\varepsilon = \mathrm{T}_{[0,1]}\left(6\alpha_j(1) - 14\right) \pm 6\varepsilon = \mathrm{T}_{[0,1]}(2 \cdot \mathbf{v}[j]) \pm 6\varepsilon.$$

Since k is valid, we can use the second bullet point in Claim 6.2, which yields $\mathbf{v}[i] = \mathbf{v}[k] \pm 18\varepsilon = \mathrm{T}_{[0,1]}(2 \cdot \mathbf{v}[j]) \pm 24\varepsilon$. □

G_{1-} Gadget. The G_{1-} gadget with input bidder j and output bidder i uses three additional auxiliary-bidders k_1, k_2, k_3. First, a base gadget is used with input j, output k_1 and parameters $(\gamma_\ell, \gamma_r, \ell, r) = (1/6, 2/3, 1/3, 2/3)$. Next, the density function of F_{k_2,k_1} has a block of volume $2/3$ in $[1 + 1/2, 1 + 3/4]$, and a block of volume $1/3$ in $[4, 5]$. Then, we use a base gadget with input k_2, output k_3 and parameters $(\gamma_\ell, \gamma_r, \ell, r) = (1/3, 1/3, 2/3, 5/6)$. Finally, we use a projection gadget with input k_3 and output i. See Figure 6.3c for an illustration.

The crucial idea behind the construction of this gadget is that the third jump point (instead of the second one) is used to encode information in some intermediate step. This allows us to simulate the non-monotone operation $x \mapsto 1 - x$.

Claim 6.4. *The G_{1-} gadget with input bidder j and output bidder i ensures that:*

- *the output bidder i is valid, and*
- *if the input bidder j is valid, then $\mathbf{v}[i] = 1 - \mathbf{v}[j] \pm 60\varepsilon$.*

Proof. First of all, note that i must be valid, because of the corresponding property of the projection gadget (Claim 6.2). Now consider the case where j is valid. By Claim 6.1 it follows that bidder k_1 is almost-valid, in particular $\alpha_{k_1}(3) = 5$ and $\alpha_{k_1}(1) \leq 3$. Let p_b denote the probability that bidder j bids b, as perceived by bidder k_1. Since j is valid, we immediately obtain that $p_0 = p_3 = p_4 = 0$. Furthermore, by the construction of $F_{k_1,j}$, it is easy to see that $p_1 = 1/6 + (1 - 1/6 - 2/3)\mathbf{v}[j] = 1/6 + \mathbf{v}[j]/6$. Next, using (6.13) we can write

$$\begin{aligned}
\alpha_{k_1}(2) &= \mathrm{T}_{[\alpha_{k_1}(1),\alpha_{k_1}(3)]}\left(3 + \frac{2p_0 + 2p_1 + p_2 \pm 2\varepsilon}{p_2 + p_3}\right) \\
&= \mathrm{T}_{[\alpha_{k_1}(1),\alpha_{k_1}(3)]}\left(3 + \frac{1 + p_1 \pm 2\varepsilon}{1 - p_1}\right) \\
&= 3 + \frac{7/6 + \mathbf{v}[j]/6}{5/6 - \mathbf{v}[j]/6} \pm 3\varepsilon \\
&= 4 + \frac{2 + 2\mathbf{v}[j]}{5 - \mathbf{v}[j]} \pm 3\varepsilon.
\end{aligned}$$

Now consider bidder k_2. Let p_b denote the probability that bidder k_1 bids b, as perceived by bidder k_2. By construction of F_{k_2,k_1} and since k_1 is almost-valid, it is easy to see that $p_0 = p_4 = 0$, $p_1 = 2/3$ and $p_2 + p_3 = 1/3$. By the same arguments used in the proof of Claim 6.1 it follows that $\alpha_{k_2}(0) \leq 1 + 1/2$. By using (6.12) we obtain $\alpha_{k_2}(3) \geq 4 + \frac{2/3 + 1/6 - \varepsilon}{1/6} \geq 5$. Next, using (6.13) we obtain

$$\alpha_{k_2}(2) \geq \mathrm{T}_{[\alpha_{k_2}(1),\alpha_{k_2}(3)]}\left(3 + \frac{2p_0 + 2p_1 + p_2 \pm 2\varepsilon}{p_2 + p_3}\right) \geq \mathrm{T}_{[\alpha_{k_2}(1),5]}\left(3 + \frac{4/3 - 2\varepsilon}{1/3}\right) = 5.$$

Now observe that by construction of F_{k_2,k_1} and the expression obtained earlier for $\alpha_{k_1}(2)$

$$p_2 = \frac{\mathrm{T}_{[4,5]}(\alpha_{k_1}(2)) - 4}{3} = \frac{2 + 2\mathbf{v}[j]}{15 - 3\mathbf{v}[j]} \pm \varepsilon.$$

As a result, it follows that

$$\frac{2p_0 + p_1}{p_1 + p_2} = \frac{2/3}{2/3 + \frac{2+2\mathbf{v}[j]}{15-3\mathbf{v}[j]} \pm \varepsilon} = \frac{2/3}{2/3 + \frac{2+2\mathbf{v}[j]}{15-3\mathbf{v}[j]}} \pm 3\varepsilon = 5/6 - \mathbf{v}[j]/6 \pm 3\varepsilon$$

where we used $\varepsilon \leq 2/15$. Finally, using (6.14) we obtain

$$\begin{aligned}
\alpha_{k_2}(1) &= \mathrm{T}_{[\alpha_{k_2}(0),\alpha_{k_2}(2)]}\left(2 + \frac{2p_0 + p_1 \pm 2\varepsilon}{p_1 + p_2}\right) \\
&= \mathrm{T}_{[\alpha_{k_2}(0),\alpha_{k_2}(2)]}\left(2 + 5/6 - \mathbf{v}[j]/6 \pm 3\varepsilon \pm \frac{2\varepsilon}{p_1 + p_2}\right) \\
&= 2 + 5/6 - \mathbf{v}[j]/6 \pm 6\varepsilon.
\end{aligned}$$

Note in particular that bidder k_2 is almost-valid, since $\varepsilon \leq 1/36$.

Since bidder k_2 is almost-valid, and we use a base gadget with $\gamma_\ell = \gamma_r = 1/3$ with input k_2 and output k_3, it follows by Claim 6.1 that bidder k_3 is valid and

$$\mathbf{v}[k_3] = \frac{\mathrm{T}_{[2+\ell,2+r]}(\alpha_{k_2}(1)) - (2+4/6)}{1/6} \pm 6\varepsilon = 1 - \mathbf{v}[j] \pm 42\varepsilon.$$

Finally, the projection gadget with input k_3 and output i ensures that $\mathbf{v}[i] = \mathbf{v}[k_3] \pm 18\varepsilon = 1 - \mathbf{v}[j] \pm 60\varepsilon$. □

G_ϕ Gadget. The G_ϕ gadget with input bidders j_1 and j_2 and output bidder i is a binary gadget with additional auxiliary-bidders k_1, k_2, k_3. First of all, for all $t \in N \setminus \{j_1, j_2, k_1\}$, we set $F_{k_1,t}$ to have density function with a single block of volume 1 in $[0,1]$. We set *both* F_{k_1,j_1} and F_{k_1,j_2} to be distributions as in our construction of the base gadget with parameters $(\gamma_\ell, \gamma_r, \ell, r) = (1/20, 8/20, 1/3, 2/3)$. The density function of F_{k_2,k_1} has a block of volume $1/2$ in $[1+1/2, 1+3/4]$, and a block of volume $1/2$ in $[3+1/2, 5]$. Next, we use a base gadget with input k_2, output k_3 and parameters $(\gamma_\ell, \gamma_r, \ell, r) = (1/3, 1/3(1+1/4), 104/200, 779/800)$. Finally, we use a G_{1-} gadget with input k_3 and output i. See Figure 6.4 for an illustration. We have the following claim.

Claim 6.5. *The G_ϕ gadget with input bidders j_1, j_2 and output bidder i ensures that:*

- *the output bidder i is valid, and*
- *if the input bidders j_1 and j_2 are valid, then*

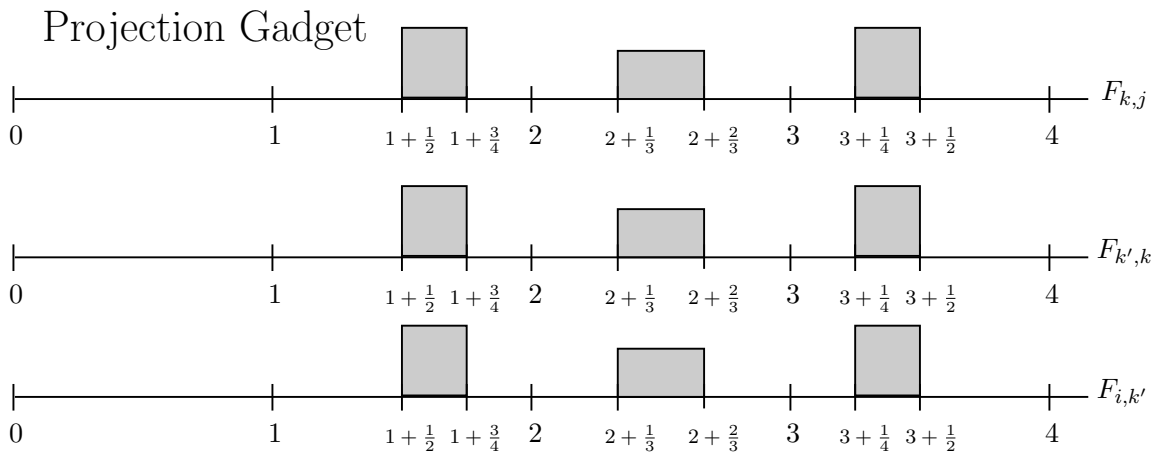
$$\mathbf{v}[i] = \phi(\mathbf{v}[j_1], \mathbf{v}[j_2]) \pm 86\varepsilon = \frac{1}{4}(\mathbf{v}[j_1]+1)(\mathbf{v}[j_2]+1) \pm 86\varepsilon.$$

Proof. Bidder i is guaranteed to be valid, because it is the output bidder of the G_{1-} gadget (Claim 6.4). Now assume that j_1 and j_2 are valid. Let p_b denote the probability that bidder j_1 bids b, as perceived by bidder k_1. Similarly, let q_b denote the probability that bidder j_2 bids b, as perceived by bidder k_1. By construction of F_{k_1,j_1} and F_{k_1,j_2}, and because j_1 and j_2 are valid, we know that $p_0 = p_3 = p_4 = q_0 = q_3 = q_4 = 0$, $p_1, q_1 \geq 1/20$ and $p_2, q_2 \geq 8/20$. Recall that $H_{k_1}(b)$ is used to denote the probability that bidder k_1 wins if she bids b (from k_1's perspective). Thus we immediately obtain that $H_{k_1}(0) = 0$, $H_{k_1}(1) = p_1q_1/3$, $H_{k_1}(2) = p_1q_1 + p_2q_1/2 + p_1q_2/2 + p_2q_2/3 = 1/3 + (p_1+q_1)/6 + p_1q_1/3$ and $H_{k_1}(3) = H_{k_1}(4) = 1$. With this in hand, we now obtain (just as we did for Equations (6.11) to (6.14)):

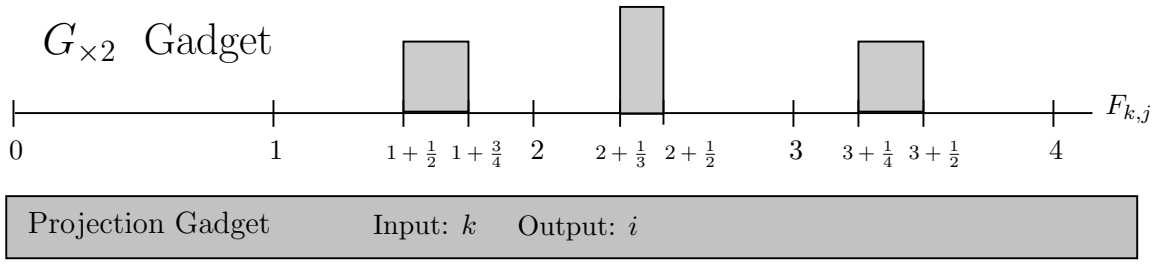
$$\alpha_{k_1}(0) \leq 1 + \frac{H_{k_1}(0)+\varepsilon}{H_{k_1}(1) - H_{k_1}(0)} = 1 + \frac{\varepsilon}{p_1q_1/3} \leq 1 + 1200\varepsilon \leq 1 + 1/2$$

since $\varepsilon \leq 1/2400$. Similarly, since $H_{k_1}(4) - H_{k_1}(3) = 0$ and $H_{k_1}(3) = 1 > \varepsilon$, we have that $\alpha_{k_1}(3) = 5$. We also have

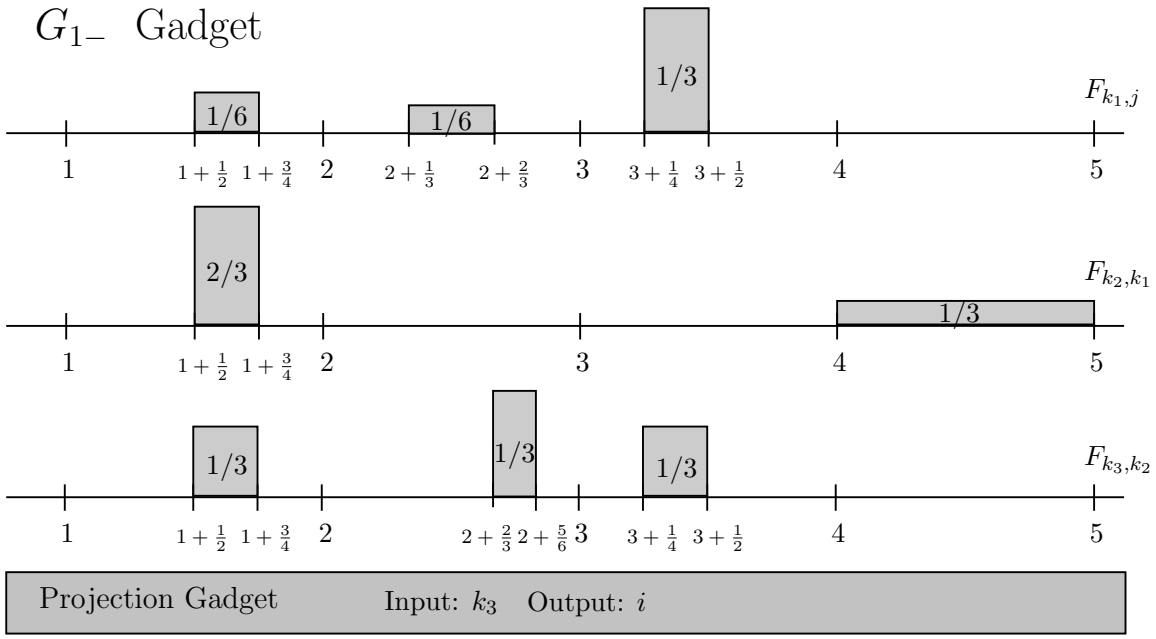
$$\alpha_{k_1}(1) \leq 2 + \frac{H_{k_1}(1)+\varepsilon}{H_{k_1}(2) - H_{k_1}(1)} \leq 2 + \frac{p_1q_1/3+\varepsilon}{1/3 + (p_1+q_1)/6} \leq 3$$



(a) The Projection gadget.



(b) The $G_{\times 2}$ gadget.



(c) The G_{1-} gadget.

Figure 6.3: The Projection, $G_{\times 2}$ and G_{1-} gadgets. The probability density functions of the corresponding subjective priors are shown.

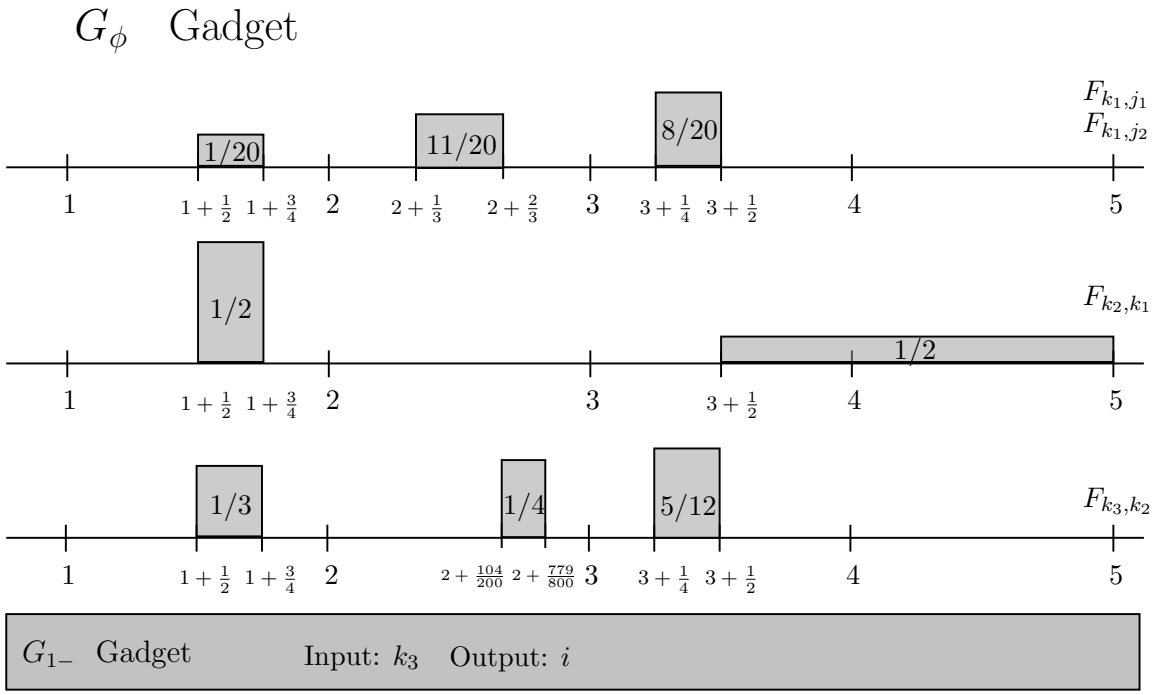


Figure 6.4: The G_ϕ gadget. The probability density functions of the corresponding subjective priors are shown.

where we used the bounds we have on these probabilities and $\varepsilon \leq 1/4$. Finally, we have

$$\begin{aligned}
\alpha_{k_1}(2) &= \mathrm{T}_{[\alpha_{k_1}(1),\alpha_{k_1}(3)]}\left(3 + \frac{H_{k_1}(2) \pm \varepsilon}{H_{k_1}(3) - H_{k_1}(2)}\right) \\
&= \mathrm{T}_{[3,5]}\left(3 + \frac{1/3 + (p_1+q_1)/6 + p_1q_1/3 \pm \varepsilon}{1 - (1/3 + (p_1+q_1)/6 + p_1q_1/3)}\right) \\
&= 3 + \frac{1 + (p_1+q_1)/2 + p_1q_1}{2 - (p_1+q_1)/2 - p_1q_1} \pm 3\varepsilon \\
&= 3 + \frac{1}{2} + \frac{3}{2}\frac{(p_1+q_1)/2 + p_1q_1}{2 - (p_1+q_1)/2 - p_1q_1} \pm 3\varepsilon
\end{aligned}$$

where we used the fact that $\frac{(p_1+q_1)/2+p_1q_1}{2-(p_1+q_1)/2-p_1q_1} \leq 1$, since $p_1, q_1 \leq 12/20$. As $p_1, q_1 \geq 1/20$ and $\varepsilon \leq 1/60$, we also have that $\alpha_{k_1}(2) \geq 3 + 1/2$. In particular, k_1 is almost-valid. Note that since j_1 and j_2 are valid, we have $p_1 = 1/20 + 11\mathbf{v}[j_1]/20$ and $q_1 = 1/20 + 11\mathbf{v}[j_2]/20$.

Next, we consider bidder k_2. Let p'_b denote the probability that bidder k_1 bids b, as perceived by bidder k_2. By the previous paragraph, we have $p'_0 = 0$, $p'_1 = 1/2$, $p'_4 = 0$ and

$$p'_2 = \frac{1}{3}\frac{3}{2}\frac{(p_1+q_1)/2 + p_1q_1}{2 - (p_1+q_1)/2 - p_1q_1} \pm 3\varepsilon = \frac{1}{2}\frac{(p_1+q_1)/2 + p_1q_1}{2 - (p_1+q_1)/2 - p_1q_1} \pm 3\varepsilon$$

where we used the fact that the height of the block of volume of F_{k_2,k_1} in $[3+1/2, 5]$ is $1/3$. Since the density function of F_{k_2,k_1} has a block of volume $1/2$ in $[1+1/2, 1+3/4]$,

as before we obtain that $\alpha_{k_2}(0) \le 1 + 1/2$. Using (6.12) and (6.13), we also have

$$\alpha_{k_2}(3) \ge 4 + \frac{p'_0 + p'_1 + p'_2 + p'_3/2 - \varepsilon}{p'_3/2 + p'_4/2} \ge 5$$

as well as

$$\alpha_{k_2}(2) \ge \mathrm{T}_{[\alpha_{k_2}(1),\alpha_{k_2}(3)]}\left(3 + \frac{2p'_0 + 2p'_1 + p'_2 \pm 2\varepsilon}{p'_2 + p'_3}\right) \ge 3 + 1/2.$$

Finally, (6.14) yields

$$\begin{aligned}
\alpha_{k_2}(1) &= \mathrm{T}_{[\alpha_{k_1}(0),\alpha_{k_1}(2)]}\left(2 + \frac{2p'_0 + p'_1 \pm 2\varepsilon}{p'_1 + p'_2}\right) \\
&= \mathrm{T}_{[\alpha_{k_1}(0),\alpha_{k_1}(2)]}\left(2 + \frac{1/2}{1/2 + \frac{1}{2}\frac{(p_1+q_1)/2+p_1q_1}{2-(p_1+q_1)/2-p_1q_1} \pm 3\varepsilon}\right) \pm 4\varepsilon \\
&= 2 + \frac{2 - (p_1+q_1)/2 - p_1q_1}{2} \pm 10\varepsilon.
\end{aligned}$$

Substituting in $p_1 = 1/20 + 11\mathbf{v}[j_1]/20$ and $q_1 = 1/20 + 11\mathbf{v}[j_2]/20$, we compute

$$\begin{aligned}
\alpha_{k_2}(1) &= 2 + 1 + 1/8 - \frac{1}{2}(11/20 + 11\mathbf{v}[j_1]/20)(11/20 + 11\mathbf{v}[j_2]/20) \pm 10\varepsilon \\
&= 2 + 9/8 - \frac{121}{200}\phi(\mathbf{v}[j_1], \mathbf{v}[j_2]) \pm 10\varepsilon.
\end{aligned}$$

Note that we have $\alpha_{k_2}(1) \in [2 + 104/200, 2 + 779/800] \pm 10\varepsilon$. In particular, bidder k_2 is almost-valid.

Since bidder k_3 is the output of a base gadget with input k_2 and parameters $(\gamma_\ell, \gamma_r, \ell, r) = (1/3, 1/3(1 + 1/4), 104/200, 779/800)$, it follows by Claim 6.1 that k_3 is valid and

$$\begin{aligned}
\mathbf{v}[k_3] &= 3(1 - \gamma_\ell - \gamma_r)\frac{\mathrm{T}_{[2+\ell,2+r]}(\alpha_{k_2}(1)) - (2+\ell)}{r - \ell} \pm 6\varepsilon \\
&= \frac{200}{121}(\alpha_{k_2}(1) - (2+\ell)) \pm 6\varepsilon \\
&= \frac{200}{121}\left(2 + 9/8 - \frac{121}{200}\phi(\mathbf{v}[j_1], \mathbf{v}[j_2]) - (2 + 104/200)\right) \pm 26\varepsilon \\
&= 1 - \phi(\mathbf{v}[j_1], \mathbf{v}[j_2]) \pm 26\varepsilon.
\end{aligned}$$

Finally, it is easy to see that the G_{1-} gadget with input k_3 and output i ensures the desired value for bidder i (Claim 6.4). □

Finishing the proof. Using the gadgets we have described above we can now enforce the constraints of the GCIRCUIT instance. Indeed, for each gate $g_i = (G, j, k)$ where $G \in \mathcal{G} = \{G_{\times 2}, G_{1-}, G_\phi\}$, it suffices to use the gadget corresponding to the gate-type G, with output bidder i and input bidder j (as well as k, in the case $G = G_\phi$). Since the distributions are subjective, we can re-use a bidder j as an input to multiple

different gadgets, without any interference. By Claims 6.3 to 6.5 it immediately follows that the gate-bidders $1, 2, \ldots, m$ must all be valid, since each of them is the output of some gadget. But this means that for any gate $g_i = (G, j, k)$, the input bidder j (and k, if applicable) will be valid, because she is also a gate-bidder. As a result, again by Claims 6.3 to 6.5, it follows that the gadgets will correctly enforce their constraints on all values $\mathbf{v}[i]$.

To obtain a solution, it suffices to set $\mathbf{v}[g_i] := \mathbf{v}[i]$ for all $i \in [m]$. For the case $\varepsilon = 0$, note that since every gate-bidder i is valid, we have that $\alpha_i(1) \in [2 + 1/3, 2 + 2/3]$ and as a result $\mathbf{v}[i] = \mathrm{T}_{[0,1]}(3(\alpha_i(1) - 2 - 1/3)) = 3(\alpha_i(1) - 2 - 1/3)$, which indeed yields an SL-reduction (as defined in Section 2.4). By scaling back to the original value space $[0, 1]$, the proof yields that for all $\varepsilon \in [0, 1/10^5]$, from any ε-BNE of the auction we can extract an 500ε-satisfying assignment for the generalised circuit. As discussed at the beginning of the section, this yields both PPAD- and FIXP-hardness.

6.5 Conclusion and Future Directions

In this chapter, we have classified the complexity of computing a Bayes-Nash equilibrium of the first-price auction with subjective priors, by proving that it is PPAD-complete. As we explained in the introduction, our result contributes fundamentally to our understanding of this celebrated auction format, as well as the literature on total search problems and TFNP. The challenging next step is to move towards the special case of the common priors assumption, where the value distribution of each bidder is common knowledge ($F_{i,j} = F_{i',j}$ for all i, i'). Our PPAD-membership result obviously already extends to this case, as it is a special case of the subjective priors setting. The really intriguing question is to extend our PPAD-hardness result to this case as well.

Another very meaningful question is to study the case where both the value distributions and the bidding space are discrete. A special case of this setting was studied by Escamocher, Miltersen and Santillan R. [EMS09], but they only obtained conclusive results for the case of two bidders with bi-valued distributions. We believe that some of our technical contributions (e.g., the computation of the best response functions or the gadgets used in the PPAD-hardness proof) can be adapted to show similar results for that case as well; we leave the details for future work.

Part II

$\mathsf{CLS} = \mathsf{PPAD} \cap \mathsf{PLS}$

Chapter 7

The Complexity of Gradient Descent: CLS = PPAD ∩ PLS

In this chapter, we study search problems that can be solved by performing Gradient Descent on a bounded convex polytopal domain and show that this class is equal to the intersection of two well-known classes: PPAD and PLS. Our results imply that the class CLS (Continuous Local Search) – which was defined by Daskalakis and Papadimitriou [DP11] as a more "natural" counterpart to PPAD ∩ PLS and contains many interesting problems – is itself equal to PPAD ∩ PLS.

The main technical contribution underpinning these results is Theorem 8.1, which states that computing a Karush-Kuhn-Tucker (KKT) point of a continuously differentiable function over the domain $[0, 1]^2$ is PPAD ∩ PLS-hard. The next chapter (Chapter 8) is dedicated to proving Theorem 8.1. In the present chapter, we define the problems and notions of interest, and explore the various ramifications of Theorem 8.1.

7.1 Overview

In this section we give a condensed and informal overview of the concepts, ideas, and techniques used in this chapter. We begin by providing informal definitions of the problems of interest and then present an overview of our results.

7.1.1 The problems of interest

The motivation for the problems we study stems from the ultimate goal of minimizing a continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ over some domain D. Unfortunately, this problem is known to be intractable, and so we instead consider relaxations where we are looking for a point where Gradient Descent terminates, or for a KKT point. Our investigation is restricted to bounded domains, namely we consider the setting where the domain D is a bounded convex polytope defined by a collection of linear inequalities. Furthermore, we also assume that the function f and its gradient ∇f

are Lipschitz-continuous over D, for some Lipschitz constant L provided in the input. Let $C^1_L(D, \mathbb{R})$ denote the set of continuously differentiable functions f from D to $\mathbb{R}$, such that f and ∇f are L-Lipschitz.

In order to define our Gradient Descent problem, we need to specify what we mean by "a point where Gradient Descent terminates." We consider the following two stopping criteria for Gradient Descent: (a) stop when we find a point such that the next iterate does not improve the objective function value, or (b) stop when we find a point such that the next iterate is the same point. In practice, of course, Gradient Descent is performed with some underlying precision parameter $\varepsilon > 0$. Thus, the appropriate stopping criteria are: (a) stop when we find a point such that the next iterate improves the objective function value by less than ε, or (b) stop when we find a point such that the next iterate is at most ε away. Importantly, note that, given a point, both criteria can be checked efficiently. This ensures that the resulting computational problems lie in TFNP. The totality of the problems follows from the simple fact that a local minimum must exist (since the domain is bounded) and any local minimum satisfies the stopping criteria. The first stopping criterion has a local search flavour and so we call the corresponding problem GD-Local-Search. The second stopping criterion is essentially asking for an approximate fixed point of the Gradient Descent dynamics, and yields the GD-Fixpoint problem.

Since we are performing Gradient Descent on a bounded domain, we have to ensure that the next iterate indeed lies in the domain D. The standard way to achieve this is to use so-called Projected Gradient Descent, which computes the next iterate as usual and then projects it onto the domain. Define Π_D to be the projection operator, that maps any point in D to itself, and any point outside D to its closest point in D (under the Euclidean norm). The two Gradient Descent problems are defined as follows.

Definition 7.1. GD-Local-Search and GD-Fixpoint *(informal)*:

Input: $\varepsilon > 0$, step size $\eta > 0$, domain D, $f \in C^1_L(D, \mathbb{R})$ and its gradient ∇f.
Output: Any point where (projected) gradient descent for f on D terminates. Namely, any $x \in D$ such that x and its next iterate $x' = \Pi_D(x - \eta \nabla f(x))$ satisfy:

- for GD-Local-Search: $f(x') \geq f(x) - \varepsilon$, *($f$ decreases by at most ε)*
- for GD-Fixpoint: $\|x - x'\| \leq \varepsilon$. *($x'$ is ε-close to x)*

In a certain sense, GD-Local-Search is a PLS-style version of Gradient Descent, while GD-Fixpoint is a PPAD-style version.[1] We show that these two versions are computationally equivalent by a triangle of reductions (see Figure 7.1). The other problem in that triangle of equivalent problems is the KKT problem, defined below.

[1] A very similar version of GD-Fixpoint was also defined by Daskalakis, Skoulakis and Zampetakis [DSZ21] and shown to be equivalent to finding an *approximate* local minimum (which is essentially the same as a KKT point).

Definition 7.2. **KKT** ***(informal)***:

Input: $\varepsilon > 0$, domain D, $f \in C^1_L(D, \mathbb{R})$ and its gradient ∇f.
Output: Any ε-KKT point of the minimisation problem for f on domain D.

A point x is a KKT point if x is feasible (it belongs to the domain D), and x is either a zero-gradient point of f, or alternatively x is on the boundary of D and the boundary constraints prevent local improvement of f. "ε-KKT" relaxes the KKT condition so as to allow inexact KKT solutions with limited numerical precision. For a formal definition of these notions see Section 7.2.1.

In the formal definitions of these problems, the function f and its gradient ∇f will be given as well-behaved arithmetic circuits (Section 2.2.2). Since we know of no easy way of checking that the circuit for ∇f indeed computes the gradient of f, and that the two functions are indeed L-Lipschitz, we add corresponding violation solutions. However, we note that our promise-preserving reductions (see Section 2.1) ensure that *our hardness results also hold for the promise versions of the problems.*

7.1.2 Results

The main technical contribution of this work is Theorem 8.1, which shows that the KKT problem is $\mathsf{PPAD} \cap \mathsf{PLS}$-hard, even when the domain is the unit square $[0,1]^2$. The hardness also holds for the promise version of the problem, because the hard instances that we construct always satisfy the promises. Theorem 8.1 is proved in the next chapter (Chapter 8), but we briefly present the consequences of this reduction here, and in more detail in the rest of this chapter.

A chain of reductions, presented in Section 7.3 and shown in Figure 7.1, which includes the "triangle" between the three problems of interest, establishes the following theorem.

Theorem 7.1. *The problems* KKT, GD-Local-Search, GD-Fixpoint *and* General-Continuous-Localopt *are* $\mathsf{PPAD} \cap \mathsf{PLS}$*-complete, even when the domain is fixed to be the unit square* $[0,1]^2$*. This hardness result continues to hold even if one considers the promise-versions of these problems, i.e., only instances without violations.*

These reductions are domain-preserving – which means that they leave the domain D unchanged – and promise-preserving – which means that they are also valid reductions between the promise versions of the problems. As a result, the other problems "inherit" the hardness result for KKT, including the fact that it holds for $D = [0,1]^2$ and even for the promise versions.

Consequences for CLS. The $\mathsf{PPAD} \cap \mathsf{PLS}$-hardness of General-Continuous-Localopt on domain $[0,1]^2$, and thus also on domain $[0,1]^3$, immediately implies the following surprising collapse.

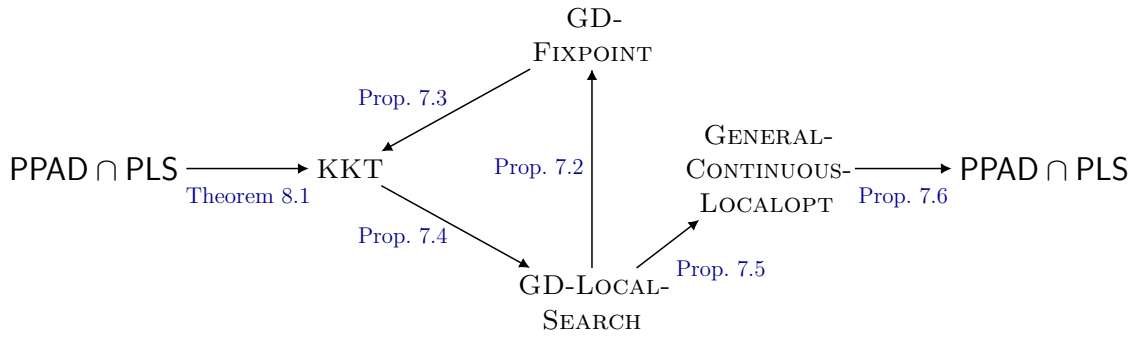


Figure 7.1: Our reductions. The main one (Theorem 8.1) is on the left; note that the other reductions are all domain- and promise-preserving.

Theorem 7.2. CLS = PPAD ∩ PLS.

As a result, it also immediately follows that the two known CLS-complete problems [DTZ18; FGMS17] are in fact PPAD ∩ PLS-complete.

Theorem 7.3. Banach *and* MetametricContraction *are* PPAD∩PLS*-complete.*

The fact that our hardness result holds on domain $[0, 1]^2$ implies that the n-dimensional variant of CLS is equal to the two-dimensional version, a fact that was not previously known. Furthermore, since our results hold even for the promise version of General-Continuous-Localopt, this implies that the definition of CLS is robust with respect to the removal of violations. Finally, we also show that restricting the circuits to be linear arithmetic circuits (that compute piecewise-linear functions) does not yield a weaker class, i.e., 2D-Linear-CLS = CLS. This is achieved using our approximation result presented in Section 2.2.2. All the consequences for CLS are discussed in detail in Section 7.4.

7.1.3 Further Related Work

Chatziafratis, Roughgarden and Wang [CRW19] showed that online gradient descent can encode general PSPACE-computations. In contrast, our result provides evidence that the problem itself (which gradient descent attempts to solve) is hard. The distinction between these two types of statements is most clearly apparent in the case of linear programming, where the simplex method can encode arbitrary PSPACE-computations [FS15], while the problem itself can be solved in polynomial time.

Daskalakis, Skoulakis and Zampetakis [DSZ21] study nonlinear *min-max* optimisation, a conceptually more complex problem than the purely "min" optimisation studied here. The PPAD-completeness they obtain reflects the extra structure present in such problems. An important point is that our hardness result requires inverse-exponential parameters, whereas Daskalakis, Skoulakis and Zampetakis [DSZ21] achieve hardness

with inverse-polynomial parameters – for us the inverse-exponential parameters are a necessary evil, since the problem can otherwise be solved in polynomial time, even in high dimension (by running Gradient Descent, see Lemma 7.5). Finally, note that in contrast to our hardness result, in the special case of *convex* optimisation our problem can be solved efficiently, even in high dimension and with inverse-exponential precision. Related work in nonlinear optimisation is covered in Section 7.2.1.

7.2 Computational Problems from Nonlinear Optimisation

In this section we formally define our three problems of interest. We begin by a brief introduction to nonlinear optimisation.

We briefly recall some notation from Section 2.2.2. Let $n \in \mathbb{N}$ be a positive integer. Throughout this chapter we use $\|\cdot\|$ to denote the standard Euclidean norm in n-dimensional space, i.e., the ℓ_2-norm in $\mathbb{R}^n$. The maximum-norm, or ℓ_∞-norm, is denoted by $\|\cdot\|_\infty$. For $x, y \in \mathbb{R}^n$, $\langle x, y\rangle := \sum_{i=1}^n x_i y_i$ denotes the inner product. For any non-empty closed convex set $D \subseteq \mathbb{R}^n$, let $\Pi_D : \mathbb{R}^n \to D$ denote the projection onto D with respect to the Euclidean norm. Formally, for any $x \in \mathbb{R}^n$, $\Pi_D(x)$ is the unique point $y \in D$ that minimizes $\|x - y\|$.

7.2.1 Background on nonlinear optimisation

The standard problem of nonlinear optimisation (also called nonlinear programming) can be formulated as follows:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0 \quad \forall i \in [m] \end{aligned} \tag{7.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function to be minimised, and $g_1, \ldots, g_m : \mathbb{R}^n \to \mathbb{R}$ are the inequality constraint functions. It is assumed that f, g_i are C^1, i.e., continuously differentiable. Throughout this chapter we consider the minimisation problem, but our results also apply to the maximisation problem, since we consider function classes that are closed under negation.

Global minimum. Unfortunately, solving the optimisation problem (7.1), namely computing a *global* minimum, is intractable, even for relatively simple objective functions and constraints ([MK87] in the context of quadratic programming, [BR92] in the context of neural networks).

Local minima. The most natural way to relax the requirement of a global minimum, is to look for a *local* minimum instead. A point $x \in \mathbb{R}^n$ is a local minimum of (7.1), if

it satisfies all the constraints, namely $x \in D$, where $D = \{y \in \mathbb{R}^n \mid g_i(x) \leq 0 \, \forall i \in [m]\}$, and if there exists $\epsilon > 0$ such that

$$f(x) \leq f(y) \qquad \forall y \in D \cap B_\epsilon(x) \tag{7.2}$$

where $B_\epsilon(x) = \{y \in \mathbb{R}^n \mid \|y - x\| \leq \epsilon\}$.

However, while the notion of a local minimum is very natural, an important issue arises when the problem is considered from the computational perspective. Looking at expression (7.2), it not clear how to efficiently check whether a given point x is a local minimum or not. Indeed, it turns out that deciding whether a given point is a local minimum is co-NP-hard, even for simple objective and constraint functions [MK87]. Furthermore, it was recently shown that computing a local minimum, even when it is guaranteed to exist, cannot be done in polynomial time unless P = NP [AZ20b], even for quadratic functions where the domain is a polytope.

Necessary optimality conditions. In order to avoid this issue, one can instead look for a point satisfying some so-called *necessary optimality conditions*. As the name suggests, these are conditions that must be satisfied for any local minimum, but might also be satisfied for points that are not local minima. Importantly, these conditions can usually be checked in polynomial time. For this reason, algorithms attempting to solve (7.1), usually try to find a point that satisfies some necessary optimality conditions instead.

KKT points. The most famous and simplest necessary optimality conditions are the Karush-Kuhn-Tucker (KKT) conditions. The KKT conditions are first-order conditions in the sense that they only involve the first derivatives (i.e., the gradients) of the functions in the problem statement. Formally, a point $x \in \mathbb{R}^n$ satisfies the KKT conditions if it is feasible, i.e., $x \in D = \{y \in \mathbb{R}^n \mid g_i(x) \leq 0 \, \forall i \in [m]\}$ and if there exist $\mu_1, \ldots, \mu_m \geq 0$ such that

$$\nabla f(x) + \sum_{i=1}^{m} \mu_i \nabla g_i(x) = 0$$

and $\mu_i g_i(x) = 0$ for all $i \in [m]$. This last condition ensures that $\mu_i > 0$ can only occur if $g_i(x) = 0$, i.e., if the ith constraint is tight. In particular, if no constraint is tight at x, then x is a KKT point if $\nabla f(x) = 0$ (in other words, if it is a stationary point). A point x that satisfies the KKT conditions is also called a KKT point. Note that given access to $\nabla f(x)$, $g_i(x)$ and $\nabla g_i(x)$, one can check in polynomial time whether x is a KKT point, since this reduces to checking the feasibility of a linear program.

Every local minimum of (7.1) must satisfy the KKT conditions, as long as the problem satisfies some so-called regularity conditions or constraint qualifications. In

this chapter, we restrict our attention to linear constraints (i.e., $g_i(x) = \langle a_i, x\rangle - b_i$). In this case, it is known that every local minimum is indeed a KKT point.

ε-KKT points. In practice, but also when studying the computational complexity in the standard Turing model (because of issues of representation), it is unreasonable to expect to find a point that exactly satisfies the KKT conditions. Instead, one looks for an *approximate* KKT point. Given $\varepsilon \geq 0$, we say that $x \in \mathbb{R}^n$ is an ε-KKT point if $x \in D$ and if there exist $\mu_1, \dots, \mu_m \geq 0$ such that

$$\left\| \nabla f(x) + \sum_{i=1}^{m} \mu_i \nabla g_i(x) \right\| \leq \varepsilon$$

and $\mu_i g_i(x) = 0$ for all $i \in [m]$. In particular, if no constraint is tight at x, then x is an ε-KKT point if $\|\nabla f(x)\| \leq \varepsilon$. Since $\|\cdot\|$ denotes the ℓ_2-norm, we can check whether a point is an ε-KKT point in polynomial time by using a convex quadratic program, which can be solved efficiently [KTK80]. If we instead use the ℓ_∞-norm or the ℓ_1-norm in the definition of an ε-KKT point, then we can check whether a point is an ε-KKT point in polynomial time by solving a linear program.

Since we focus on the case where $D = \{y \in \mathbb{R}^n \mid Ay \leq b\}$, $(A, b) \in \mathbb{R}^{m\times n} \times \mathbb{R}^m$, we can rewrite the KKT conditions as follows. A point $x \in \mathbb{R}^n$ is an ε-KKT point if $x \in D$ and if there exist $\mu_1, \dots, \mu_m \geq 0$ such that

$$\left\| \nabla f(x) + A^T \mu \right\| \leq \varepsilon$$

and $\langle \mu, Ax - b\rangle = 0$. Note that this exactly corresponds to the earlier definition adapted to this case. In particular, the condition "$\mu_i[Ax - b]_i = 0$ for all $i \in [m]$" is equivalent to $\langle \mu, Ax - b\rangle = 0$, since $\mu_i \geq 0$ and $[Ax - b]_i \leq 0$ for all $i \in [m]$.

It is known that if there are no constraints, then it is NP-hard to decide whether a KKT point exists [AZ20a]. This implies that, in general, unless P = NP, there is no polynomial-time algorithm that computes a KKT point of (7.1). However, this hardness result does not say anything about one very important special case, namely when the feasible region D is a compact set (in particular, when it is a bounded polytope defined by linear constraints). Indeed, in that case, a KKT point is guaranteed to exist—since a local minimum is guaranteed to exist—and easy to verify, and thus finding a KKT point is a total search problem in the class TFNP. In particular, this means that, for compact D, the problem of computing a KKT point cannot be NP-hard, unless NP = co-NP [MP91]. In this chapter, we provide strong evidence that the problem remains hard for such bounded domains, and, in fact, even when the feasible region is as simple as $D = [0, 1]^2$.

The problem of finding an ε-KKT point has also been studied in the "black box" model, where we only have oracle access to the function and its gradient, and count

the number of oracle calls needed to solve the problem. Vavasis [Vav93] proved that at least $\Omega(\sqrt{L/\varepsilon})$ calls are needed to find an ε-KKT point of a continuously differentiable function $f : [0,1]^2 \to \mathbb{R}$ with L-Lipschitz gradient. It was recently shown by Bubeck and Mikulincer [BM20] that this bound is tight up to a logarithmic factor. For the high-dimensional case, Carmon et al. [CDHS20] showed a tight bound of $\Theta(1/\varepsilon^2)$, when the Lipschitz constant is fixed.

7.2.2 The KKT problem

Given the definition of ε-KKT points in the previous section, we can formally define a computational problem where the goal is to compute such a point. Our formalisation of this problem assumes that f and ∇f are provided in the input as well-behaved arithmetic circuits (Section 2.2.2). However, it is unclear if, given a circuit f, we can efficiently determine whether it corresponds to a continuously differentiable function, and whether the circuit for ∇f indeed computes its gradient. Thus, one has to either consider the promise version of the problem (where this is guaranteed to hold for the input), or add violation solutions like in the definition of Continuous-Localopt (Section 2.3.4). In order to ensure that our problem is in TFNP, we formally define it with violation solutions. However, we note that our hardness results also hold for the promise versions.

The type of violation solution that we introduce to ensure that ∇f is indeed the gradient of f is based on the following version of Taylor's theorem.

Lemma 7.1 (Taylor's theorem)**.** *Let $D \subseteq \mathbb{R}^n$ be convex and $f : \mathbb{R}^n \to \mathbb{R}$. Let $g : \mathbb{R}^n \to \mathbb{R}^n$ be L-Lipschitz-continuous on D (w.r.t. the ℓ_2-norm). Then, f is continuously differentiable on D with $\nabla f = g$, if and only if for all $x, y \in D$ we have*

$$\left|f(y) - f(x) - \langle g(x), y - x\rangle\right| \leq \frac{L}{2}\|y - x\|^2.$$

Proof. First, let us assume that f is continuously differentiable on D and $\nabla f = g$. Consider any points $x, y \in D$ and let $h : [0,1] \to \mathbb{R}$ be defined by $h(t) = f(x+t(y-x))$. Then, h is continuously differentiable on $[0,1]$ and $h'(t) = \langle \nabla f(x + t(y-x)), y - x\rangle = g(x + t(y-x)), y - x\rangle$. Furthermore, h' is $(L\|x-y\|^2)$-Lipschitz-continuous on $[0,1]$, since

$$\begin{aligned}
|h'(t_1) - h'(t_2)| &= |\langle g(x + t_1(y-x)) - g(x + t_2(y-x)), y - x\rangle| \\
&\leq \|g(x + t_1(y-x)) - g(x + t_2(y-x))\| \cdot \|y - x\| \\
&\leq L \cdot \|t_1(y-x) - t_2(y-x)\| \cdot \|y - x\| \\
&\leq L \cdot |t_1 - t_2| \cdot \|y - x\|^2
\end{aligned}$$

where we used the Cauchy-Schwarz inequality. We also used the fact that g is L-Lipschitz on D, and $x + t(y - x) \in D$ for all $t \in [0, 1]$. Now, we can write

$$f(y) - f(x) - \langle g(x), y - x \rangle = h(1) - h(0) - h'(0) = \int_0^1 (h'(t) - h'(0)) \, \mathrm{d}t$$

and thus

$$\begin{aligned} |f(y) - f(x) - \langle g(x), y - x \rangle| \leq \int_0^1 |h'(t) - h'(0)| \, \mathrm{d}t &\leq \int_0^1 L \cdot \|x - y\|^2 \cdot |t| \, \mathrm{d}t \\ &= \frac{L}{2} \|y - x\|^2 \end{aligned}$$

which proves one direction of the statement.

In order to prove the other direction, it suffices to observe that the condition in the Lemma implies

$$\big| f(y) - f(x) - \langle g(x), y - x \rangle \big| = o(\|y - x\|)$$

since $(L/2) \cdot \|y - x\|^2 = o(\|y - x\|)$ (where the asymptotic notation is with respect to $\|x - y\| \to 0$). But this exactly corresponds to one of the various equivalent definitions of the gradient of f. In other words, the condition implies that f is differentiable at x and $\nabla f(x) = g(x)$. Since this holds for all $x \in D$, and g is continuous on D, it follows that f is continuously differentiable on D with $\nabla f = g$. □

We are now ready to formally define our KKT problem.

Definition 7.3. **KKT**:

Input:

- precision parameter $\varepsilon > 0$,
- $(A, b) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m$ defining a bounded non-empty domain $D = \{x \in \mathbb{R}^n : Ax \leq b\}$,
- well-behaved arithmetic circuits $f : \mathbb{R}^n \to \mathbb{R}$ and $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$,
- Lipschitz constant $L > 0$.

Output: An ε-KKT point for the minimisation problem of f on domain D.
Formally, find $x \in D$ such that there exist $\mu_1, \ldots, \mu_m \geq 0$ such that

$$\Big\| \nabla f(x) + A^T \mu \Big\| \leq \varepsilon$$

and $\langle \mu, Ax - b \rangle = 0$.
Alternatively, we also accept one of the following violations as a solution:

- (f or ∇f is not L-Lipschitz) $x, y \in D$ such that

$$|f(x) - f(y)| > L\|x - y\| \qquad \text{or} \qquad \|\nabla f(x) - \nabla f(y)\| > L\|x - y\|,$$

- (∇f is not the gradient of f) $x, y \in D$ that contradict Taylor's theorem (Lemma 7.1), i.e.,

$$\left|f(y) - f(x) - \langle \nabla f(x), y - x\rangle\right| > \frac{L}{2}\|y - x\|^2.$$

Note that all conditions on the input of the KKT problem can be checked in polynomial time. In particular, we can use linear programming to check that the domain is bounded and non-empty. With regards to a solution $x \in D$, there is no need to include the values $\mu_1, \ldots, \mu_m$ as part of a solution. Indeed, given $x \in D$, we can check in polynomial time whether there exist such $\mu_1, \ldots, \mu_m$ by solving the following convex quadratic program:

$$\begin{aligned} \min_{\mu \in \mathbb{R}^m} \quad & \left\|\nabla f(x) + A^T\mu\right\|^2 \\ \text{s.t.} \quad & \langle \mu, Ax - b\rangle = 0 \\ & \mu \geq 0 \end{aligned}$$

If the optimal value of this program is strictly larger than ε^2, then x is not an ε-KKT point. Otherwise, it is an ε-KKT point and the optimal $\mu_1, \ldots, \mu_m$ certify this. If we use the ℓ_∞-norm or the ℓ_1-norm instead of the ℓ_2-norm for the definition of ε-KKT points, then we can check whether a point is an ε-KKT point using the same approach (except that we do not take the square of the norm, and we simply obtain a linear program). Whether we use the ℓ_2-norm, the ℓ_∞-norm or the ℓ_1-norm for the definition of ε-KKT points has no impact on the complexity of the KKT problem defined above. Indeed, is is easy to reduce the various versions to each other.

Note that ε and L are provided in binary representation in the input. This is important, since our hardness result in Theorem 8.1 relies on at least one of those two parameters being exponential in the size of the input. If both parameters are provided in unary, then the problem can be solved in polynomial time on domain $[0, 1]^n$ (see Lemma 7.5).

7.2.3 Gradient Descent problems

In this section we formally define our two versions of the Gradient Descent problem. Since we consider Gradient Descent on bounded domains D, we need to ensure that the next iterate indeed lies in D. The standard way to handle this is by using so-called *Projected* Gradient Descent, where the next iterate is computed using a standard Gradient Descent step and then projected onto D using Π_D. Formally,

$$x^{(k+1)} \leftarrow \Pi_D\left(x^{(k)} - \eta \nabla f\left(x^{(k)}\right)\right)$$

where $\eta > 0$ is the step size. Throughout, we only consider the case where the step size is fixed, i.e., the same in all iterations. Our first version of the problem

considers the stopping criterion is: stop if the next iterate improves the objective function value by less than ε.

Definition 7.4. **GD-Local-Search**:

Input:

- precision/stopping parameter $\varepsilon > 0$,
- step size $\eta > 0$,
- $(A, b) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m$ defining a bounded non-empty domain $D = \{x \in \mathbb{R}^n : Ax \leq b\}$,
- well-behaved arithmetic circuits $f : \mathbb{R}^n \to \mathbb{R}$ and $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$,
- Lipschitz constant $L > 0$.

Output: Any point where (projected) gradient descent for f on domain D with fixed step size η terminates. Formally, find $x \in D$ such that

$$f\Big(\Pi_D\big(x - \eta \nabla f(x)\big)\Big) \geq f(x) - \varepsilon.$$

Alternatively, we also accept one of the following violations as a solution:

- (f or ∇f is not L-Lipschitz) $x, y \in D$ such that

$$|f(x) - f(y)| > L\|x - y\| \qquad \text{or} \qquad \|\nabla f(x) - \nabla f(y)\| > L\|x - y\|,$$

- (∇f is not the gradient of f) $x, y \in D$ that contradict Taylor's theorem (Lemma 7.1), i.e.,

$$\big|f(y) - f(x) - \langle \nabla f(x), y - x\rangle\big| > \frac{L}{2}\|y - x\|^2.$$

Our second version of the problem considers the stopping criterion: stop if the next iterate is ε-close to the current iterate.

Definition 7.5. **GD-Fixpoint**:

Input:

- precision/stopping parameter $\varepsilon > 0$,
- step size $\eta > 0$,
- $(A, b) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m$ defining a bounded non-empty domain $D = \{x \in \mathbb{R}^n : Ax \leq b\}$,
- well-behaved arithmetic circuits $f : \mathbb{R}^n \to \mathbb{R}$ and $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$,
- Lipschitz constant $L > 0$.

Output: Any point that is an ε-approximate fixed point of (projected) gradient descent for f on domain D with fixed step size η. Formally, find $x \in D$ such that

$$\left\| x - \Pi_D\big(x - \eta \nabla f(x)\big) \right\| \leq \varepsilon.$$

Alternatively, we also accept one of the following violations as a solution:

- (f or ∇f is not L-Lipschitz) $x, y \in D$ such that

$$|f(x) - f(y)| > L\|x - y\| \qquad \text{or} \qquad \|\nabla f(x) - \nabla f(y)\| > L\|x - y\|,$$

- (∇f is not the gradient of f) $x, y \in D$ that contradict Taylor's theorem (Lemma 7.1), i.e.,

$$\big|f(y) - f(x) - \langle \nabla f(x), y - x \rangle\big| > \frac{L}{2}\|y - x\|^2.$$

The comments made about the KKT problem in the previous section also apply to these two problems. In particular, we show that even the promise versions of the two Gradient Descent problems remain PPAD $\cap$ PLS-hard. In other words, the hard instances we construct have no violations.

Remark 7.1. An interesting question is: what happens if we omit the last violation (namely, the one about Taylor's theorem) from the definitions of these problems? For GD-Local-Search it turns out that this does not change the complexity of the problem. Indeed, removing the last violation means that the functions f and ∇f can now be completely unrelated. However, for GD-Local-Search it is not hard to see that the problem remains in CLS (in fact, note that the proof of Proposition 7.5 which reduces GD-Local-Search to General-Continuous-Localopt does not use violations to Taylor's theorem). Thus, the problem remains PPAD $\cap$ PLS-complete.

On the other hand, for GD-Fixpoint and KKT it turns out that omitting the last violation does change the complexity of the problem. Indeed, note that unlike GD-Local-Search, the property that some x must satisfy in order to be a (non-violation) solution only depends on ∇f, and not at all on f. As a result, it is easy to reduce from the problem of finding an approximate Brouwer fixed point of a function $g : [0, 1]^2 \to [0, 1]^2$ to either of these two problems, by letting $f(x) = 0$, $\nabla f(x) = x - g(x)$, and setting the remaining parameters appropriately. It follows that GD-Fixpoint and KKT without the last violation are PPAD-hard, and in fact it can be shown that they are PPAD-complete. Finally, note that it is easy to see that GD-Fixpoint and KKT remain equivalent if we remove the last violation: one direction is given by the proof of Proposition 7.3 (which still works without violations to Taylor's theorem), and the other direction can be obtained by using the arguments in step 2 of the proof of Proposition 7.4.

7.3 Gradient Descent and KKT are PPAD ∩ PLS-complete

In this section, we show how the PPAD∩PLS-hardness of KKT (Theorem 8.1) implies that our other problems of interest, including our Gradient Descent problems, are PPAD ∩ PLS-complete. Namely, we prove:

Theorem 7.1. *The problems* KKT*,* GD-Local-Search*,* GD-Fixpoint *and* General-Continuous-Localopt *are* PPAD ∩ PLS*-complete, even when the domain is fixed to be the unit square* $[0,1]^2$*. This hardness result continues to hold even if one considers the promise-versions of these problems, i.e., only instances without violations.*

The hardness results in this theorem are the "best possible" in the following sense:

- Promise-problem: as mentioned in the theorem, the hardness holds even for the promise-versions of these problems. In other words, the hard instances that we construct are not pathological: they satisfy all the conditions that we would expect from the input, e.g., ∇f is indeed the gradient of f, ∇f and f are indeed L-Lipschitz-continuous, etc.

- Domain: the problems remain hard even if we fix the domain to be the unit square $[0,1]^2$, which is arguably the simplest two-dimensional bounded domain. All the problems become polynomial-time solvable if the domain is one-dimensional (Lemma 7.4).

- Exponential parameters: in all of our problems, the parameters, such as ε and L, are provided in the input in *binary* representation. This means that the parameters are allowed to be exponentially small or large with respect to the length of the input. Our hardness results make use of this, since the proof of Theorem 8.1 constructs an instance of KKT where ε is some constant, but L is exponential in the input length. By a simple transformation, this instance can be transformed into one where ε is exponentially small and L is constant (see Section 8.4.1). It is easy to see that at least one of ε or L must be exponentially large/small, for the problem to be hard on the domain $[0,1]^2$. However, this continues to hold even in high dimension, i.e., when the domain is $[0,1]^n$ (Lemma 7.5). In other words, if the parameters are given in unary, the problem is easy, even in high dimension. This is in contrast with the problem of finding a Brouwer fixed point, where moving to domain $[0,1]^n$ makes it possible to prove PPAD-hardness even when the parameters are given in unary.

Theorem 7.1 follows from Theorem 8.1, proved in Chapter 8, and a set of domain- and promise-preserving reductions as pictured in Figure 7.1, which are presented in the rest of this section as follows. In Section 7.3.2 we show that the problems KKT, GD-Local-Search and GD-Fixpoint are equivalent. Then, in Section 7.3.3 we reduce GD-Local-Search to General-Continuous-Localopt, and finally we show that General-Continuous-Localopt lies in PPAD ∩ PLS. First, we present some useful tools from convex analysis.

7.3.1 Tools from convex analysis and a generalisation of Farkas' Lemma

Let $D \subseteq \mathbb{R}^n$ be a non-empty closed convex set. Recall that the projection $\Pi_D : \mathbb{R}^n \to D$ is defined by $\Pi_D(x) = \operatorname{argmin}_{y \in D} \|x - y\|$, where $\|\cdot\|$ denotes the Euclidean norm. It is known that $\Pi_D(x)$ always exists and is unique. The following two results are standard tools in convex analysis, see, e.g., [Ber99].

Lemma 7.2. *Let D be a non-empty closed convex set in $\mathbb{R}^n$ and let $y \in \mathbb{R}^n$. Then for all $x \in D$ it holds that*

$$\langle y - \Pi_D(y), x - \Pi_D(y) \rangle \leq 0.$$

Proposition 7.1. *Let D_1 and D_2 be two disjoint non-empty closed convex sets in $\mathbb{R}^n$ and such that D_2 is bounded. Then, there exist $c \in \mathbb{R}^n \setminus \{0\}$ and $d \in \mathbb{R}$ such that $\langle c, x \rangle < d$ for all $x \in D_1$, and $\langle c, x \rangle > d$ for all $x \in D_2$.*

We will need the following generalisation of Farkas' Lemma, which we prove below. For $\varepsilon = 0$, we recover the usual statement of Farkas' Lemma.

Lemma 7.3. *Let $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^n$ and $\varepsilon \geq 0$. Then exactly one of the following two statements holds:*

1. $\exists x \in \mathbb{R}^n : Ax \leq 0$, $\langle b, x \rangle > \varepsilon \|x\|$,
2. $\exists y \in \mathbb{R}^m : \|A^T y - b\| \leq \varepsilon$, $y \geq 0$.

Proof. Let us first check that both statements cannot hold at the same time. Indeed, if this were the case, then we would obtain the following contradiction

$$\varepsilon \|x\| < \langle b, x \rangle = \langle A^T y, x \rangle + \langle b - A^T y, x \rangle \leq \langle y, Ax \rangle + \|b - A^T y\| \|x\| \leq \varepsilon \|x\|$$

where we used the fact that $\langle y, Ax \rangle \leq 0$ and the Cauchy-Schwarz inequality.

Now, let us show that if statement 2 does not hold, then statement 1 must necessarily hold. Let $D_1 = \{A^T y : y \geq 0\}$ and $D_2 = \{x : \|x - b\| \leq \varepsilon\}$. Note that since statement 2 does not hold, it follows that D_1 and D_2 are disjoint. Furthermore,

it is easy to check that D_1 and D_2 satisfy the conditions of Proposition 7.1. Thus, there exist $c \in \mathbb{R}^n \setminus \{0\}$ and $d \in \mathbb{R}$ such that $\langle c, x\rangle < d$ for all $x \in D_1$, and $\langle c, x\rangle > d$ for all $x \in D_2$. In particular, we have that for all $y \geq 0$, $\langle Ac, y\rangle = \langle c, A^T y\rangle < d$. From this it follows that $Ac \leq 0$, since if $[Ac]_i > 0$ for some i, then $\langle Ac, y\rangle \geq d$ for $y = \frac{|d|}{[Ac]_i} e_i$.

In order to show that $x := c$ satisfies the first statement, it remains to prove that $\langle b, c\rangle > \varepsilon\|c\|$. Note that by setting $y = 0$, we get that $0 = \langle c, A^T 0\rangle < d$. Let $z = b - \varepsilon\frac{c}{\|c\|}$. Since $z \in D_2$, it follows that $\langle c, z\rangle > d > 0$. Since $\langle c, z\rangle = \langle c, b\rangle - \varepsilon\|x\|$, statement 1 indeed holds. □

7.3.2 KKT and the Gradient Descent problems are equivalent

The equivalence between KKT, GD-Local-Search and GD-Fixpoint is proved by providing a "triangle" of reductions as shown in Figure 7.1. Namely, we show that GD-Local-Search reduces to GD-Fixpoint (Proposition 7.2), GD-Fixpoint reduces to KKT (Proposition 7.3), and KKT reduces to GD-Local-Search (Proposition 7.4). All the reductions are domain- and promise-preserving.

Proposition 7.2. GD-Local-Search *reduces to* GD-Fixpoint *using a domain- and promise-preserving reduction.*

Proof. Let $(\varepsilon, \eta, A, b, f, \nabla f, L)$ be an instance of GD-Local-Search. The reduction simply constructs the instance $(\varepsilon', \eta, A, b, f, \nabla f, L)$ of GD-Fixpoint, where $\varepsilon' = \varepsilon/L$. This reduction is trivially domain-preserving and it is also promise-preserving, because any violation of the constructed instance is immediately also a violation of the original instance. Clearly, the reduction can be computed in polynomial time, so it remains to show that any (non-violation) solution of the constructed instance can be mapped back to a solution or violation of the original instance.

Consider any solution $x \in D$ of the GD-Fixpoint instance, i.e.,

$$\|x - y\| = \|x - \Pi_D(x - \eta\nabla f(x))\| \leq \varepsilon'.$$

where $y = \Pi_D(x - \eta\nabla f(x))$. If x, y do not satisfy the L-Lipschitzness of f, then we have obtained a violation. Otherwise, it must be that

$$|f(x) - f(y)| \leq L\|x - y\| \leq L\varepsilon' = \varepsilon.$$

In particular, it follows that

$$f(y) \geq f(x) - \varepsilon$$

which means that x is a solution of the original GD-Local-Search instance. □

Proposition 7.3. GD-Fixpoint *reduces to* KKT *using a domain- and promise-preserving reduction.*

Proof. Let $(\varepsilon, \eta, A, b, f, \nabla f, L)$ be an instance of GD-Fixpoint. The reduction simply constructs the instance $(\varepsilon', A, b, f, \nabla f, L)$ of KKT, where $\varepsilon' = \varepsilon/\eta$. This reduction is trivially domain-preserving and it is also promise-preserving, because any violation of the constructed instance is immediately also a violation of the original instance. Clearly, the reduction can be computed in polynomial time, so it remains to show that any (non-violation) solution of the constructed instance can be mapped back to a solution or violation of the original instance.

In more detail, we will show that any ε'-KKT point must be an ε-approximate fixed point of the gradient descent dynamics. Consider any ε'-KKT point of the KKT instance, i.e., a point $x \in D$ such that there exists $\mu \geq 0$ with $\langle \mu, Ax - b \rangle = 0$ and $\|\nabla f(x) + A^T \mu\| \leq \varepsilon'$.

Let $y = \Pi_D(x - \eta \nabla f(x))$. We want to show that $\|x - y\| \leq \varepsilon$. Since y is the projection of $x - \eta \nabla f(x)$ onto D, by Lemma 7.2 it follows that for all $z \in D$

$$\langle x - \eta \nabla f(x) - y, z - y \rangle \leq 0.$$

Letting $z := x$, this implies that

$$\begin{aligned} \|x - y\|^2 \leq \eta \langle \nabla f(x), x - y \rangle &= \eta \langle \nabla f(x) + A^T \mu, x - y \rangle - \eta \langle A^T \mu, x - y \rangle \\ &\leq \eta \langle \nabla f(x) + A^T \mu, x - y \rangle \\ &\leq \eta \|\nabla f(x) + A^T \mu\| \cdot \|x - y\| \end{aligned}$$

where we used the Cauchy-Schwarz inequality and the fact that $\langle A^T \mu, x - y \rangle \geq 0$, which follows from

$$\langle A^T \mu, x - y \rangle = \langle \mu, A(x - y) \rangle = \langle \mu, Ax - b \rangle - \langle \mu, Ay - b \rangle \geq 0$$

since $\langle \mu, Ax - b \rangle = 0$, $\mu \geq 0$ and $Ay - b \leq 0$ (because $y \in D$).

We can now show that $\|x - y\| \leq \varepsilon$. If $\|x - y\| = 0$, this trivially holds. Otherwise, divide both sides of the inequality obtained above by $\|x - y\|$, which yields

$$\|x - y\| \leq \eta \|\nabla f(x) + A^T \mu\| \leq \eta \cdot \varepsilon' = \varepsilon. \qquad \square$$

Proposition 7.4. KKT *reduces to* GD-Local-Search *using a domain- and promise-preserving reduction.*

Proof. Let $(\varepsilon, A, b, f, \nabla f, L)$ be an instance of KKT. The reduction simply constructs the instance $(\varepsilon', \eta, A, b, f, \nabla f, L)$ of GD-Local-Search, where $\varepsilon' = \frac{\varepsilon^2}{8L}$ and $\eta = \frac{1}{L}$. This reduction is trivially domain-preserving and it is also promise-preserving, because any violation of the constructed instance is immediately also a violation of the original instance. Clearly, the reduction can be computed in polynomial time, so it remains to show that any (non-violation) solution of the constructed instance can be mapped back to a solution or violation of the original instance.

Consider any $x \in D$ that is a solution of the GD-Local-Search instance and let $y = \Pi_D(x - \eta \nabla f(x))$. Then, it must be that $f(y) \geq f(x) - \varepsilon'$. We begin by showing that this implies that $\|x - y\| \leq \frac{\varepsilon}{2L}$, or we can find a violation of the KKT instance.

Step 1: Bounding $\|x - y\|$. If x and y do not satisfy Taylor's theorem (Lemma 7.1), then we immediately obtain a violation. If they do satisfy Taylor's theorem, it holds that

$$\langle \nabla f(x), x - y \rangle - \frac{L}{2}\|y - x\|^2 \leq f(x) - f(y) \leq \varepsilon'.$$

Now, since y is the projection of $x - \eta \nabla f(x)$ onto D, by Lemma 7.2 it follows that $\langle x - \eta \nabla f(x) - y, z - y \rangle \leq 0$ for all $z \in D$. In particular, by letting $z := x$, we obtain that

$$\langle \nabla f(x), x - y \rangle \geq \frac{1}{\eta}\langle x - y, x - y \rangle = L\|y - x\|^2$$

where we used the fact that $\eta = 1/L$. Putting the two expressions together we obtain that

$$\frac{L}{2}\|y - x\|^2 = L\|y - x\|^2 - \frac{L}{2}\|y - x\|^2 \leq \varepsilon'$$

which yields that $\|x - y\| \leq \sqrt{2\varepsilon'/L} = \frac{\varepsilon}{2L}$.

Step 2: Obtaining an ε-KKT point. Next, we show how to obtain an ε-KKT point or a violation of the KKT instance. Note that if $y - x = -\eta \nabla f(x)$, then we immediately have that $\|\nabla f(x)\| = \|x - y\|/\eta \leq \varepsilon/2$, i.e., x is an ε-KKT point. However, because of the projection Π_D used in the computation of y, in general we might not have that $y - x = -\eta \nabla f(x)$ and, most importantly, x might not be an ε-KKT point. Nevertheless, we show that y will necessarily be an ε-KKT point.

Since y is the projection of $x - \eta \nabla f(x)$ onto D, by Lemma 7.2 it follows that for all $z \in D$

$$\langle x - \eta \nabla f(x) - y, z - y \rangle \leq 0.$$

From this it follows that for all $z \in D$

$$\langle -\nabla f(x), z - y \rangle \leq \frac{1}{\eta}\langle y - x, z - y \rangle \leq \frac{1}{\eta}\|x - y\| \cdot \|z - y\| \leq \frac{\varepsilon}{2}\|z - y\|$$

where we used the Cauchy-Schwarz inequality, $\eta = 1/L$ and $\|x - y\| \leq \varepsilon/2L$. Next, unless x and y yield a violation to the L-Lipschitzness of ∇f, it must hold that $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \leq \varepsilon/2$. Thus, we obtain that for all $z \in D$

$$\begin{aligned} \langle -\nabla f(y), z - y\rangle &= \langle -\nabla f(x), z - y\rangle + \langle \nabla f(x) - \nabla f(y), z - y\rangle \\ &\leq \frac{\varepsilon}{2}\|z - y\| + \|\nabla f(x) - \nabla f(y)\| \cdot \|z - y\| \\ &\leq \varepsilon\|z - y\| \end{aligned} \tag{7.3}$$

where we used the Cauchy-Schwarz inequality.

Let $I = \{i \in [m] : [Ay - b]_i = 0\}$, i.e., the indices of the constraints that are tight at y. Denote by $A_I \in \mathbb{R}^{(m-|I|)\times n}$ the matrix obtained by only keeping the rows of A that correspond to indices in I. Consider any $p \in \mathbb{R}^n$ such that $A_I p \leq 0$. Then, there exists a sufficiently small $\alpha > 0$ such that $z = y + \alpha p \in D$. Indeed, note that $[Az - b]_i = [Ay - b]_i + \alpha[Ap]_i$ and thus

- for $i \in I$, we get that $[Az - b]_i \leq 0$, since $[Ay - b]_i = 0$ and $[Ap]_i \leq 0$,
- for $i \notin I$, we have that $[Ay-b]_i < 0$. If $[Ap]_i \leq 0$, then we obtain $[Az-b]_i \leq 0$ as above. If $[Ap]_i > 0$, then it also holds that $[Az-b]_i \leq 0$, as long as $\alpha \leq -\frac{[Ay-b]_i}{[Ap]_i}$.

Thus, it suffices to pick $\alpha = \min\left\{-\frac{[Ay-b]_i}{[Ap]_i} : i \notin I, [Ap]_i > 0\right\}$. Note that this indeed ensures that $\alpha > 0$.

Since $z = y + \alpha p \in D$, using (7.3) we get that

$$\langle -\nabla f(y), p\rangle = \frac{1}{\alpha}\langle -\nabla f(y), z - y\rangle \leq \frac{\varepsilon}{\alpha}\|z - y\| = \varepsilon\|p\|.$$

As a result, we have shown that the statement

$$\exists p \in \mathbb{R}^n : A_I p \leq 0, \langle -\nabla f(y), p\rangle > \varepsilon\|p\|$$

does not hold. By the stronger version of Farkas' Lemma, which we proved above (Lemma 7.3), it follows that there exists $\nu \in \mathbb{R}^{|I|}_{\geq 0}$ such that $\|A_I^T\nu + \nabla f(y)\| \leq \varepsilon$. Let $\mu \in \mathbb{R}^m_{\geq 0}$ be such that μ agrees with ν on indices $i \in I$, i.e., $\mu_I = \nu$, and $\mu_i = 0$ for $i \notin I$. Then we immediately obtain that $A_I^T\nu = A^T\mu$ and thus $\|A^T\mu + \nabla f(y)\| \leq \varepsilon$. Since we also have that $\langle \mu, Ay - b\rangle = \langle \mu_I, [Ay - b]_I\rangle = 0$ (because $[Ay - b]_I = 0$), it follows that y indeed is an ε-KKT point of f on domain D. □

7.3.3 From GD-Local-Search to PPAD ∩ PLS

In this section we show that GD-Local-Search reduces to General-Continuous-Localopt (Proposition 7.5), and then that General-Continuous-Localopt lies in PPAD ∩ PLS (Proposition 7.6).

Proposition 7.5. GD-Local-Search *reduces to* General-Continuous-Localopt *using a domain- and promise-preserving reduction.*

Proof. This essentially follows from the fact that the local search version of Gradient Descent is a special case of continuous local search, which is captured by the General-Continuous-Localopt problem. Let $(\varepsilon, A, b, \eta, f, \nabla f, L)$ be an instance of GD-Local-Search. The reduction simply constructs the instance $(\varepsilon, A, b, p, g, L')$ of General-Continuous-Localopt, where $p(x) = f(x)$, $g(x) = x - \eta \nabla f(x)$ and $L' = \max\{\eta L + 1, L\}$. We can easily construct an arithmetic circuit computing g, given the arithmetic circuit computing ∇f. It follows that the reduction can be computed in polynomial time. In particular, since we extend ∇f by using only the gates $-$ and $\times\zeta$, the circuit for g is also well-behaved.

Let us now show that any solution to the General-Continuous-Localopt instance yields a solution to the GD-Local-Search instance. First of all, by construction of g, it immediately follows that any local optimum solution of the General-Continuous-Localopt instance is also a non-violation solution to the GD-Local-Search instance.

Next, we show that any pair of points $x, y \in D$ that violate the $(\eta L + 1)$-Lipschitzness of g, also violate the L-Lipschitzness of ∇f. Indeed, if x, y do not violate the L-Lipschitzness of ∇f, then

$$\|g(x) - g(y)\| \leq \|x - y\| + \eta \|\nabla f(x) - \nabla f(y)\| \leq (\eta L + 1)\|x - y\|.$$

In particular, any violation to the L'-Lipschitzness of g yields a violation to the L-Lipschitzness of ∇f.

Finally, any violation to the L'-Lipschitzness of p immediately yields a violation to the L-Lipschitzness of f. Since any violation to General-Continuous-Localopt yields a violation to GD-Local-Search, the reduction is also promise-preserving. □

Proposition 7.6. General-Continuous-Localopt *lies in* PPAD ∩ PLS.

Proof. This essentially follows by the same arguments that were used by Daskalakis and Papadimitriou [DP11] to show that CLS lies in PPAD∩PLS. The only difference is that here the domain is allowed to be more general. Consider any instance $(\varepsilon, A, b, p, g, L)$ of General-Continuous-Localopt.

The containment of General-Continuous-Localopt in PPAD follows from a reduction to the problem of finding a fixed point guaranteed by Brouwer's fixed point theorem, which is notoriously PPAD-complete. Indeed, let $x^* \in D$ be any ε/L-approximate fixed point of the function $x \mapsto \Pi_D(g(x))$, i.e., such that $\|\Pi_D(g(x^*)) - x^*\| \leq \varepsilon/L$. Then, unless x^* and $\Pi_D(g(x^*))$ yield a violation of L-Lipschitzness of

p, it follows that $p(\Pi_D(g(x^*))) \geq p(x^*) - \varepsilon$, i.e., x^* is a solution of the General-Continuous-Localopt instance. Formally, the reduction works by constructing the instance $(\varepsilon', A, b, g, L)$ of General-Brouwer, where $\varepsilon' = \varepsilon/L$. The formal definition of General-Brouwer can be found in Appendix C, where it is also proved that the problem lies in PPAD.

The containment of CLS in PLS was proved by Daskalakis and Papadimitriou [DP11] by reducing Continuous-Localopt to a problem called Real-Localopt, which they show to lie in PLS. Real-Localopt is defined exactly as Continuous-Localopt, except that the function g is not required to be continuous. In order to show the containment of General-Continuous-Localopt in PLS, we reduce to the appropriate generalisation of Real-Localopt, which we simply call General-Real-Localopt. Formally, the reduction is completely trivial, since any instance of General-Continuous-Localopt is also an instance of General-Real-Localopt, and solutions can be mapped back as is. The formal definition of General-Real-Localopt can be found in Appendix C, where it is also proved that the problem lies in PLS. □

7.3.4 Minor observations on KKT and our other problems of interest

Lemma 7.4. General-Continuous-Localopt *with fixed dimension* $n = 1$ *can be solved in polynomial time. As a result, this also holds for* KKT*,* GD-Local-Search *and* GD-Fixpoint*.*

Proof. This is a straightforward consequence of the fact that finding Brouwer fixed points in one dimension is easy. Consider any instance $(\varepsilon, A, b, p, g, L)$ of General-Continuous-Localopt with $n = 1$. It is easy to see that any ε/L-approximate fixed point of $x \mapsto \Pi_D(g(x))$ immediately yields a solution to the General-Continuous-Localopt instance.

Thus, we proceed as follows. First of all, from A and b we can directly determine $t_1, t_2 \in \mathbb{R}$ such that $D = [t_1, t_2]$. Note that the bit-size of t_1 and t_2 is polynomial in the input size. Then define a grid of points on the interval $[t_1, t_2]$ such that the distance between consecutive points is ε/L^2. Finally, using binary search, find two consecutive points x_1 and x_2 such that $g(x_1) \geq x_1$ and $g(x_2) \leq x_2$. One of these two points has to be an ε/L-approximate fixed point of $x \mapsto \Pi_D(g(x))$ (or we obtain a violation of Lipschitz-continuity). Binary search takes polynomial time, because the number of points is at most exponential in the input size.

Since the other three problems reduce to General-Continuous-Localopt using domain-preserving reductions (see Section 7.3), it follows that they can also be solved in polynomial time when $n = 1$. □

Lemma 7.5. KKT *on domain* $[0,1]^n$ *can be solved in polynomial time in* $1/\varepsilon$, L *and the sizes of the circuits for* f *and* ∇f.

Proof. This follows from the fact that the problem can be solved by Gradient Descent in polynomial time in those parameters. Let $(\varepsilon, f, \nabla f, L)$ be an instance of KKT with domain $[0,1]^n$. First, compute $f(0)$ in polynomial time in size(f). If f is indeed L-Lipschitz-continuous, then it follows that $f(x) \in I = [f(0) - \sqrt{n}L, f(0) + \sqrt{n}L]$ for all $x \in [0,1]^n$. If we ever come across a point where this does not hold, we immediately obtain a violation of L-Lipschitz-continuity of f. So, for the rest of this proof we simply assume that $f(x) \in I$ for all $x \in [0,1]^n$.

Note that the length of interval I is $2\sqrt{n}L$, which is polynomial in L and n. By using the reduction in the proof of Proposition 7.4, we can solve our instance by solving the instance $(\varepsilon', \eta, f, \nabla f, L)$ of GD-Local-Search, where $\varepsilon' = \frac{\varepsilon^2}{8L}$ and $\eta = \frac{1}{L}$. The important observation here is that this instance of GD-Local-Search can be solved by applying Gradient Descent with step size η and with any starting point, in at most $\frac{|I|}{\varepsilon'} = \frac{16\sqrt{n}L^2}{\varepsilon^2}$ steps. Indeed, every step must improve the value of f by ε', otherwise we have found a solution. It is easy to see that each step of Gradient Descent can be done in polynomial time in size(∇f), n and $\log L$. Since the number of steps is polynomial in $1/\varepsilon$, L and n, the problem can be solved in polynomial time in $1/\varepsilon$, L, n, size(f) and size(∇f). Finally note that $n \leq$ size(f) (because f has n input gates). □

7.4 Consequences for Continuous Local Search

In this section, we explore the consequences of Theorem 8.1 (and Theorem 7.1) for the class CLS, defined by Daskalakis and Papadimitriou [DP11] to capture problems that can be solved by "continuous local search" methods. In Section 7.4.2 we also consider a seemingly weaker version of CLS, which we call Linear-CLS, and show that it is in fact the same as CLS. Finally, we define a Gradient Descent problem where we do not have access to the gradient of the function (which might, in fact, not even be differentiable) and instead use "finite differences" to compute an approximate gradient. We show that this problem remains PPAD ∩ PLS-complete.

7.4.1 Consequences for CLS

The class CLS was defined by Daskalakis and Papadimitriou [DP11] as a more natural counterpart to PPAD ∩ PLS. Indeed, Daskalakis and Papadimitriou noted that all the known PPAD ∩ PLS-complete problems were unnatural, namely uninteresting combinations of a PPAD-complete and a PLS-complete problem. As a result, they defined CLS, a subclass of PPAD ∩ PLS, which is a more natural combination of PPAD

and PLS, and conjectured that CLS is a *strict* subclass of PPAD ∩ PLS. They were able to prove that various interesting problems lie in CLS, thus further strengthening the conjecture that CLS is a more natural subclass of PPAD ∩ PLS, and more likely to capture the complexity of interesting problems.

It follows from our results that, surprisingly, CLS is actually equal to PPAD ∩ PLS.

Theorem 7.2. CLS = PPAD ∩ PLS.

Recall that in Theorem 7.1, we have shown that General-Continuous-Localopt with domain $[0,1]^2$ is PPAD ∩ PLS-complete. Theorem 7.2 follows from the fact that this problem lies in CLS, almost by definition. Before proving this in Proposition 7.7 below, we explore some further consequences of our results for CLS.

An immediate consequence is that the two previously known CLS-complete problems (see Section 2.3.4) are in fact PPAD ∩ PLS-complete.

Theorem 7.3. Banach *and* MetametricContraction *are* PPAD∩PLS*-complete.*

Furthermore, our results imply that the definition of CLS is "robust" in the following sense:

- Dimension: the class CLS was defined by Daskalakis and Papadimitriou [DP11] as the set of all TFNP problems that reduce to 3D-Continuous-Localopt, i.e., Continuous-Localopt with $n = 3$. Even though it is easy to see that kD-Continuous-Localopt reduces to $(k+1)$D-Continuous-Localopt (Lemma 7.6), it is unclear how to construct a reduction in the other direction. Indeed, similar reductions exist for the Brouwer problem, but they require using a discrete equivalent of Brouwer, namely End-of-Line, as an intermediate step. Since no such discrete problem was known for CLS, this left open the possibility of a hierarchy of versions of CLS, depending on the dimension, i.e., 2D-CLS ⊂ 3D-CLS ⊂ 4D-CLS .... We show that even the two-dimensional version is PPAD ∩ PLS-hard, and thus the definition of CLS is indeed independent of the dimension used. In other words,

  $$\text{2D-CLS} = \text{CLS} = n\text{D-CLS}.$$

  Note that this is tight, since 1D-Continuous-Localopt can be solved in polynomial time (Lemma 7.4), i.e., 1D-CLS = FP.

- Domain: some interesting problems can be shown to lie in CLS, but the reduction produces a polytopal domain, instead of the standard hypercube $[0,1]^n$. In other words, they reduce to General-Continuous-Localopt, which we have defined as a generalisation of Continuous-Localopt. Since General-Continuous-Localopt is PPAD ∩ PLS-complete (Theorem 7.1), it follows that CLS can equivalently be defined as the set of all TFNP problems that reduce to General-Continuous-Localopt.

- Promise: the problem Continuous-Localopt, which defines CLS, is a problem with violation solutions. One can instead consider promise-CLS, which is defined as the set of all TFNP problems that reduce to a promise version of Continuous-Localopt. In the promise version of Continuous-Localopt, we restrict our attention to instances that satisfy the promise, i.e., where the functions p and g are indeed L-Lipschitz-continuous. The class promise-CLS could possibly be weaker than CLS, since the reduction is required to always map to instances of Continuous-Localopt without violations. However, it follows from our results that promise-CLS = CLS, since the promise version of Continuous-Localopt is shown to be PPAD ∩ PLS-hard, even on domain $[0,1]^2$ (Theorem 7.1).

- Turing reductions: since PPAD and PLS are closed under Turing reductions [BJ12], it is easy to see that this also holds for PPAD ∩ PLS, and thus by our result also for CLS.

- Circuits: CLS is defined using the problem Continuous-Localopt where the functions are represented by general arithmetic circuits. If one restricts the type of arithmetic circuit that is used, this might yield a weaker version of CLS. Linear arithmetic circuits (Section 2.2.2) are a natural class of circuits that arise when reducing from various natural problems. We define Linear-CLS as the set of problems that reduce to Continuous-Localopt with linear arithmetic circuits. In Section 7.4.2 we show that 2D-Linear-CLS = CLS.

Before moving on to Section 7.4.2 and Linear-CLS, we provide the last reduction in the chain of reductions proving Theorem 7.2.

Proposition 7.7. General-Continuous-Localopt *with fixed domain* $[0,1]^2$ *reduces to* 2D-Continuous-Localopt *using a promise-preserving reduction. In particular, the problem lies in* CLS.

Proof. Given an instance (ε, p, g, L) of General-Continuous-Localopt with fixed domain $[0,1]^2$, we construct the instance (ε, p, g', L) of 2D-Continuous-Localopt, where $g'(x) = \Pi_D(g(x))$. Note that since $D = [0,1]^2$, the projection Π_D can easily be computed as $[\Pi_D(x)]_i = \min\{1, \max\{0, x_i\}\}$ for all $x \in \mathbb{R}^2$ and $i \in [2]$. In particular, since we extend g by using only the gates $-$, $\times\zeta$, min, max and rational constants, the circuit for g' is also well-behaved.

Any non-violation solution of the constructed instance is also a solution of the original instance. Any violation of the constructed instance is immediately mapped back to a violation of the original instance. In particular, it holds that $\|g'(x) - g'(y)\| \leq \|g(x) - g(y)\|$ for all $x, y \in [0,1]^2$, since projecting two points cannot increase the

distance between them. This implies that any violation of the L-Lipschitzness of g' is also a violation of the L-Lipschitzness of g. Note that by Lemma 7.7 we do not need to ensure that the codomain of p is in $[0, 1]$. Finally, it is easy to see that 2D-Continuous-Localopt lies in CLS, since it immediately reduces to 3D-Continuous-Localopt (Lemma 7.6). □

7.4.2 Linear-CLS and Gradient Descent with finite differences

The class CLS was defined by Daskalakis and Papadimitriou [DP11] using the Continuous-Localopt problem which uses arithmetic circuits with gates in $\{+, -, \min, \max, \times, <\}$ and rational constants. In this section we show that even if we restrict ourselves to linear arithmetic circuits (i.e., only the gates in $\{+, -, \min, \max, \times\zeta\}$ and rational constants are allowed), the Continuous-Localopt problem and CLS remain just as hard as the original versions.

Definition 7.6. **Linear-Continuous-Localopt**:

Input:

- precision/stopping parameter $\varepsilon > 0$,
- linear arithmetic circuits $p : [0, 1]^n \to [0, 1]$ and $g : [0, 1]^n \to [0, 1]^n$.

Output: An approximate local optimum of p with respect to g. Formally, find $x \in [0, 1]^n$ such that

$$p(g(x)) \geq p(x) - \varepsilon.$$

For $k \in \mathbb{N}$, we let kD-Linear-Continuous-Localopt denote the problem Linear-Continuous-Localopt where n is fixed to be equal to k. Note that the definition of Linear-Continuous-Localopt does not require violation solutions, since every linear arithmetic circuit is automatically Lipschitz-continuous with a Lipschitz-constant that can be represented with a polynomial number of bits (Lemma 2.2). In particular, Linear-Continuous-Localopt reduces to Continuous-Localopt and thus to General-Continuous-Localopt.

We define the class 2D-Linear-CLS as the set of all TFNP problems that reduce to 2D-Linear-Continuous-Localopt. We show that:

Theorem 7.4. 2D-Linear-CLS = PPAD ∩ PLS.

Note that, just as for CLS, the one-dimensional version can be solved in polynomial time, i.e., 1D-Linear-CLS = FP. The containment 2D-Linear-CLS ⊆ PPAD ∩ PLS immediately follows from the fact that 2D-Linear-CLS ⊆ CLS ⊆ PPAD ∩ PLS. The other, more interesting, containment in Theorem 7.4 can be proved by directly

reducing 2D-Continuous-Localopt to 2D-Linear-Continuous-Localopt. This reduction mainly relies on a more general result which says that any arithmetic circuit can be arbitrarily well approximated by a linear arithmetic circuit on a bounded domain (Theorem 2.1).

Instead of reducing 2D-Continuous-Localopt to 2D-Linear-Continuous-Localopt, we prove Theorem 7.4 by a different route that also allows us to introduce a problem which might be of independent interest. To capture the cases where the gradient is not available or perhaps too expensive to compute, we consider a version of Gradient Descent where the *finite differences* approach is used to compute an approximate gradient, which is then used as usual to obtain the next iterate. Formally, given a finite difference spacing parameter $h > 0$, the approximate gradient $\widetilde{\nabla}_h f(x)$ at some point $x \in [0,1]^n$ is computed as

$$\left[\widetilde{\nabla}_h f(x)\right]_i = \frac{f(x + h \cdot e_i) - f(x - h \cdot e_i)}{2h}$$

for all $i \in [n]$. The computational problem is defined as follows. Note that even though we define the problem on the domain $[0,1]^n$, it can be defined on more general domains as in our other problems.

Definition 7.7. **GD-Finite-Diff**:

Input:

- precision/stopping parameter $\varepsilon > 0$,
- step size $\eta > 0$,
- finite difference spacing parameter $h > 0$,
- linear arithmetic circuit $f : \mathbb{R}^n \to \mathbb{R}$.

Output: Any point where (projected) gradient descent for f on domain $D = [0,1]^n$ using finite differences to approximate the gradient and fixed step size η terminates. Formally, find $x \in [0,1]^n$ such that

$$f(\Pi_D(x - \eta \widetilde{\nabla}_h f(x))) \geq f(x) - \varepsilon$$

where for all $i \in [n]$

$$\left[\widetilde{\nabla}_h f(x)\right]_i = \frac{f(x + h \cdot e_i) - f(x - h \cdot e_i)}{2h}.$$

GD-Finite-Diff immediately reduces to Linear-Continuous-Localopt by setting $p := f$ and $g := \Pi_D(x - \eta \widetilde{\nabla}_h f(x))$. It is easy to construct a linear arithmetic circuit computing g, given a linear arithmetic circuit computing f. Note, in particular,

that the projection Π_D can be computed by a linear circuit since $D = [0,1]^n$. Indeed, $[\Pi_D(x)]_i = \min\{1, \max\{0, x_i\}\}$ for all $i \in [n]$ and $x \in \mathbb{R}^n$. Finally, the restriction of the codomain of p to $[0,1]$ can be handled exactly as in the proof of Lemma 7.7.

In particular, the reduction from GD-Finite-Diff to Linear-Continuous-Localopt is domain-preserving and thus Theorem 7.4 immediately follows from the following theorem.

Theorem 7.5. GD-Finite-Diff *is* PPAD ∩ PLS*-complete, even with fixed domain* $[0,1]^2$.

This result is interesting by itself, because the problem GD-Finite-Diff is arguably quite natural, but also because it is the first problem that is complete for PPAD ∩ PLS (and CLS) that has a *single* arithmetic circuit in the input. Note that our other problems which we prove to be PPAD ∩ PLS-complete, as well as the previously known CLS-complete problems, all have two arithmetic circuits in the input.

Proof. As explained above, GD-Finite-Diff immediately reduces to Linear-Continuous-Localopt and thus to General-Continuous-Localopt, which lies in PPAD ∩ PLS by Proposition 7.6. Thus, it remains to show that GD-Finite-Diff is PPAD ∩ PLS-hard when we fix $n = 2$. This is achieved by reducing from GD-Local-Search on domain $[0,1]^2$, which is PPAD ∩ PLS-hard by Theorem 7.1. In fact, we can even simplify the reduction by only considering GD-Local-Search instances that have some additional structure, but remain PPAD ∩ PLS-hard. Namely, consider an instance $(\varepsilon, \eta, f, \nabla f, L)$ of GD-Local-Search on domain $D = [0,1]^2$ such that:

- ∇f is the gradient of f,
- f and ∇f are L-Lipschitz-continuous on $[-1,2]^2$.

To see that the problem remains PPAD ∩ PLS-hard even with these restrictions, note that the restrictions are satisfied by the hard instances constructed for the KKT problem in the proof of Theorem 8.1, and that the reduction from KKT to GD-Local-Search in Proposition 7.4 also trivially preserves them. In particular, even though the proof of Theorem 8.1 only mentions that f and ∇f are L-Lipschitz-continuous on $[0,1]^2$, the same arguments also show that they are L-Lipschitz-continuous on $[-1,2]^2$ (where L has been scaled by some fixed constant).

Let us now reduce from the instance $(\varepsilon, \eta, f, \nabla f, L)$ of GD-Local-Search to GD-Finite-Diff. We construct the instance $(\varepsilon', \eta, h, F)$ of GD-Finite-Diff where $\varepsilon' = \varepsilon/4$, $h = \min\{1, \frac{\varepsilon}{8\eta L^2}\}$ and F is a linear arithmetic circuit that is obtained as follows. Let $\delta = \min\{\varepsilon/4, Lh^2/2\}$. By Theorem 2.1 and Remark B.1, we can construct a linear arithmetic circuit $F : \mathbb{R}^2 \to \mathbb{R}$ in polynomial time in size(f), $\log L$ and $\log(1/\delta)$

such that $|f(x) - F(x)| \leq \delta$ for all $x \in [-1, 2]^2$. Note that the second possibility in Theorem 2.1 cannot occur, since f is guaranteed to be L-Lipschitz-continuous on $[-1, 2]^2$.

Consider any solution of that instance of GD-Finite-Diff, i.e., a point $x \in [0, 1]^2$ such that $F(\Pi_D(x - \eta\widetilde{\nabla}_h F(x))) \geq F(x) - \varepsilon/4$. Let us show that x is a solution to the original GD-Local-Search instance, i.e., that $f(\Pi_D(x - \eta\nabla f(x))) \geq f(x) - \varepsilon$.

We have that for $i \in \{1, 2\}$

$$
\begin{aligned}
&\left|\left[\widetilde{\nabla}_h f(x)\right]_i - \left[\nabla f(x)\right]_i\right| \\
&= \left|\frac{f(x + h \cdot e_i) - f(x - h \cdot e_i)}{2h} - \left[\nabla f(x)\right]_i\right| \\
&\leq \frac{1}{2h}\left(\left|f(x + h \cdot e_i) - f(x) - h\left[\nabla f(x)\right]_i\right| + \left|-f(x - h \cdot e_i) + f(x) - h\left[\nabla f(x)\right]_i\right|\right) \\
&= \frac{1}{2h}\left(\left|f(x + h \cdot e_i) - f(x) - \left\langle \nabla f(x), (x + h \cdot e_i) - x\right\rangle\right|\right. \\
&\quad \left. + \left|-f(x - h \cdot e_i) + f(x) + \left\langle \nabla f(x), (x - h \cdot e_i) - x\right\rangle\right|\right) \\
&\leq \frac{1}{2h}\left(\frac{L}{2}\left\|h \cdot e_i\right\|^2 + \frac{L}{2}\left\|-h \cdot e_i\right\|^2\right) = \frac{Lh}{2}
\end{aligned}
$$

where we used Taylor's theorem (Lemma 7.1). Note that $x \pm h \cdot e_i \in [-1, 2]^2$, since $h \leq 1$. Furthermore, it is easy to see that $\left|[\widetilde{\nabla}_h F(x)]_i - [\widetilde{\nabla}_h f(x)]_i\right| \leq \delta/h$, since F approximates f up to error δ on all of $[-1, 2]^2$. It follows that $\left\|\widetilde{\nabla}_h F(x) - \nabla f(x)\right\| \leq \sqrt{2}(\delta/h + Lh/2) \leq 2Lh$. From this it follows that

$$
\begin{aligned}
&\left|f\Big(\Pi_D\big(x - \eta\nabla f(x)\big)\Big) - f\Big(\Pi_D\big(x - \eta\widetilde{\nabla}_h F(x)\big)\Big)\right| \\
&\leq L \cdot \left\|\Pi_D\big(x - \eta\nabla f(x)\big) - \Pi_D\big(x - \eta\widetilde{\nabla}_h F(x)\big)\right\| \\
&\leq L \cdot \left\|\big(x - \eta\nabla f(x)\big) - \big(x - \eta\widetilde{\nabla}_h F(x)\big)\right\| \\
&\leq \eta L \cdot \left\|\widetilde{\nabla}_h F(x) - \nabla f(x)\right\| \\
&\leq 2\eta L^2 h.
\end{aligned}
$$

Finally, note that $|f(x) - F(x)| \leq \delta \leq \varepsilon/4$ and

$$
\left|f\Big(\Pi_D\big(x - \eta\widetilde{\nabla}_h F(x)\big)\Big) - F\Big(\Pi_D\big(x - \eta\widetilde{\nabla}_h F(x)\big)\Big)\right| \leq \delta \leq \varepsilon/4.
$$

Thus, since $F\Big(\Pi_D\big(x - \eta\widetilde{\nabla}_h F(x)\big)\Big) \geq F(x) - \varepsilon/4$, it follows that

$$
f\Big(\Pi_D\big(x - \eta\nabla f(x)\big)\Big) \geq f(x) - 3\varepsilon/4 - 2\eta L^2 h
$$

i.e., x is a solution to the original GD-Local-Search instance, since $2\eta L^2 h \leq \varepsilon/4$. □

7.4.3 Minor observations on Continuous-Localopt

In this section, we prove two simple results about Continuous-Localopt.

Lemma 7.6. *For all integers $k_2 > k_1 > 0$, k_1*D-Continuous-Localopt *reduces to k_2*D-Continuous-Localopt *using a promise-preserving reduction.*

Proof. For $x \in \mathbb{R}^{k_2}$, we write $x = (x_1, x_2)$, where $x_1 \in \mathbb{R}^{k_1}$ and $x_2 \in \mathbb{R}^{k_2-k_1}$. Let (ε, p, g, L) be an instance of k_1D-Continuous-Localopt. The reduction constructs the instance (ε, p', g', L) of k_2D-Continuous-Localopt, where

$$p'(x) = p'(x_1, x_2) = p(x_1) \qquad \text{and} \qquad g'(x) = g'(x_1, x_2) = (g(x_1), 0).$$

Clearly, the arithmetic circuits for p' and g' can be constructed in polynomial time and are well-behaved.

Since $|p'(x) - p'(y)| = |p(x_1) - p(y_1)| \leq L\|x_1 - y_1\| \leq L\|x - y\|$, it is clear that any violation $x, y \in [0,1]^{k_2}$ of L-Lipschitzness for p' also yields a violation $x_1, y_1 \in [0,1]^{k_1}$ for p. Similarly, since

$$\|g'(x) - g'(y)\| = \|(g(x_1), 0) - (g(y_1), 0)\| = \|g(x_1) - g(y_1)\| \leq L\|x_1 - y_1\| \leq L\|x - y\|,$$

any violation x, y of L-Lipschitzness for g' also yields a violation x_1, y_1 for g. Thus, any violation of the constructed instance is always mapped back to a violation of the original instance, and the reduction is indeed promise-preserving.

Finally, note that any proper solution $x \in [0,1]^{k_2}$ of the constructed instance, i.e., such that $p'(g'(x)) \geq p'(x) - \varepsilon$, immediately yields a solution $x_1 \in [0,1]^{k_1}$ of the original instance. □

Lemma 7.7. Continuous-Localopt *with codomain $[0,1]$ for function p is equivalent to* Continuous-Localopt *without this restriction.*

Proof. It is clear that the version with the restriction trivially reduces to the version without the restriction. Thus, it remains to show the other direction, namely that Continuous-Localopt without the codomain restriction reduces to the restricted version.

Let (ε, p, g, L) be an instance of Continuous-Localopt with domain $[0,1]^n$ and without a codomain restriction for p. The reduction constructs the instance $(\varepsilon', p', g, L')$ of Continuous-Localopt with domain $[0,1]^n$, where $\varepsilon' = \frac{\varepsilon}{2nL}$, $L' = \max\{L, \frac{1}{2n}\}$ and

$$p'(x) = \min\left\{1, \max\left\{0, \frac{1}{2} + \frac{p(x) - p(z_c)}{2nL}\right\}\right\}$$

where $z_c = (1/2, 1/2, \ldots, 1/2)$ is the centre of $[0,1]^n$. Note that the arithmetic circuit computing p' can be computed in polynomial time given the circuit for p, and that

the modification of p will require using gates $\times\zeta$, but no general multiplication gates. Thus, the circuit for p' is also well-behaved. Note, in particular, that the value $p(z_c)$ can be computed in polynomial time in the size of the circuit for p. It follows that the reduction can be computed in polynomial time.

First of all, let us show that any point $x \in [0,1]^n$ such that $p'(x) \neq \frac{1}{2} + \frac{p(x)-p(z_c)}{2nL}$ will immediately yield a violation of the L-Lipschitzness of p. Indeed, if x and z_c satisfy the L-Lipschitzness of p, then this means that

$$|p(x) - p(z_c)| \leq L\|x - z_c\| \leq nL$$

since $x, z_c \in [0,1]^n$. As a result, it follows that $\frac{1}{2} + \frac{p(x)-p(z_c)}{2nL} \in [0,1]$ and thus $p'(x) = \frac{1}{2} + \frac{p(x)-p(z_c)}{2nL}$.

In the rest of this proof we assume that we always have $p'(x) = \frac{1}{2} + \frac{p(x)-p(z_c)}{2nL}$, since we can immediately extract a violation if we ever come across a point x where this does not hold. Let us now show that any solution of the constructed instance immediately yields a solution of the original instance. Clearly, any violation of the L'-Lipschitzness of g is trivially also a violation of L-Lipschitzness.

Next assume that $x, y \in [0,1]^n$ are a violation of L'-Lipschitzness of p'. Let us show by contradiction that x, y must be a violation of L-Lipschitzness for p. Indeed, assume that x, y satisfy the L-Lipschitzness for p, then

$$|p'(x) - p'(y)| = \left|\frac{p(x) - p(y)}{2nL}\right| \leq \frac{1}{2n}\|x - y\|$$

which is a contradiction to x, y being a violation of L'-Lipschitzness of p'.

Finally, consider any proper solution of the constructed instance, i.e., $x \in [0,1]^n$ such that $p'(g(x)) \geq p'(x) - \varepsilon'$. Then it follows straightforwardly that $p(g(x)) \geq p(x) - 2nL\varepsilon'$, which implies that x is a solution to the original instance, since $2nL\varepsilon' = \varepsilon$. Note that the reduction is also promise-preserving, since we always map violations of the constructed instance back to violations of the original instance. □

7.5 Conclusion and Future Directions

Our results may help to identify the complexity of the following problems that are known to lie in PPAD ∩ PLS:

- Mixed-Congestion: The problem of finding a *mixed* Nash equilibrium of a congestion game. It is known that finding a *pure* Nash equilibrium is PLS-complete [FPT04]. Babichenko and Rubinstein [BR21] have recently applied our main result to obtain PPAD ∩ PLS-completeness for Mixed-Congestion. It would be interesting to extend this to *network* congestion games, where the strategies are represented implicitly.

- POLYNOMIAL-KKT: The special case of the KKT problem where the function is a polynomial, provided explicitly in the input (exponents in unary). A consequence of the above-mentioned reduction by Babichenko and Rubinstein [BR21] is that the problem is PPAD ∩ PLS-complete for polynomials of degree 5. It is an interesting open problem to extend this hardness result to lower degree polynomials.

- CONTRACTION: Find a fixed point of a function that is contracting with respect to some ℓ_p-norm.

- TARSKI: Find a fixed point of an order-preserving function, as guaranteed by Tarski's theorem [EPRY20; FS21; DQY20].

- COLOURFULCARATHÉODORY: A problem based on a theorem in convex geometry [MMSS17].

The first three problems on this list were known to lie in CLS [DP11], while the other two were only known to lie in PPAD ∩ PLS.

The collapse between CLS and PPAD ∩ PLS raises the question of whether the class EOPL (for End of Potential Line), a subclass of CLS, is also equal to PPAD ∩ PLS. The class EOPL, or more precisely its subclass UEOPL (with U for unique), is known to contain various problems of interest that have unique solutions such as Unique Sink Orientation (USO), the P-matrix Linear Complementarity Problem (P-LCP), Simple Stochastic Games (SSG) and Parity Games [FGMS20]. We conjecture that EOPL ≠ PPAD ∩ PLS. Unlike CLS, EOPL has a more standard combinatorial definition that is simultaneously a special case of END-OF-LINE and LOCALOPT. While PPAD ∩ PLS captures problems that have a PPAD-type proof of existence and a PLS-type proof of existence, EOPL seems to capture problems that have a single proof of existence which is simultaneously of PPAD- and PLS-type. The first step towards confirming this conjecture would be to provide an oracle separation between EOPL and PPAD ∩ PLS, in the sense of Beame et al. [BCE+98].

Ishizuka [Ish21] has recently provided a somewhat artificial problem that is complete for PPA ∩ PLS. This raises the interesting question of whether PPA ∩ PLS, and other intersections of well-studied classes, also admit natural complete problems, or if PPAD ∩ PLS is in fact an isolated case.

Chapter 8

KKT is PPAD ∩ PLS-hard

In this chapter, we prove the following theorem, which is the main technical contribution underpinning all of the results presented in the previous chapter, including the collapse CLS = PPAD ∩ PLS.

Theorem 8.1. KKT *is* PPAD ∩ PLS*-hard, even when the domain is fixed to be the unit square* $[0,1]^2$*. The hardness continues to hold even if one considers the promise-version of the problem, i.e., only instances without violations.*

8.1 Proof Overview

In order to prove Theorem 8.1 we provide a polynomial-time many-one reduction from the PPAD ∩ PLS-complete problem Either-Solution(End-of-Line,Iter) (see Section 2.3.5) to the KKT problem on domain $[0,1]^2$.

Given an instance I^{EOL} of End-of-Line and an instance I^{ITER} of Iter, we construct an instance $I^{\text{KKT}} = (\varepsilon, f, \nabla f, L)$ of the KKT problem on domain $[0,1]^2$ such that from any ε-KKT point of f, we can efficiently obtain a solution to either I^{EOL} or I^{ITER}. The function f and its gradient ∇f are first defined on an exponentially small grid on $[0,1]^2$, and then extended within every small square of the grid by using bicubic interpolation. This ensures that the function is continuously differentiable on the whole domain. The most interesting part of the reduction is how the function is defined on the grid points, by using information from I^{EOL}, and then, where necessary, also from I^{ITER}.

Embedding I^{EOL}. The domain is first subdivided into $2^n \times 2^n$ *big squares*, where $[2^n]$ is the set of vertices in I^{EOL}. The big squares on the diagonal (shaded in Figure 8.1) represent the vertices of I^{EOL} and the function f is constructed so as to embed the directed edges in the graph of I^{EOL}. If the edge (v_1, v_2) in I^{EOL} is a forward edge, i.e, $v_1 < v_2$, then there will be a "green path" going from the big square of v_1 to the big square of v_2. On the other hand, if the edge (v_1, v_2) in I^{EOL} is a

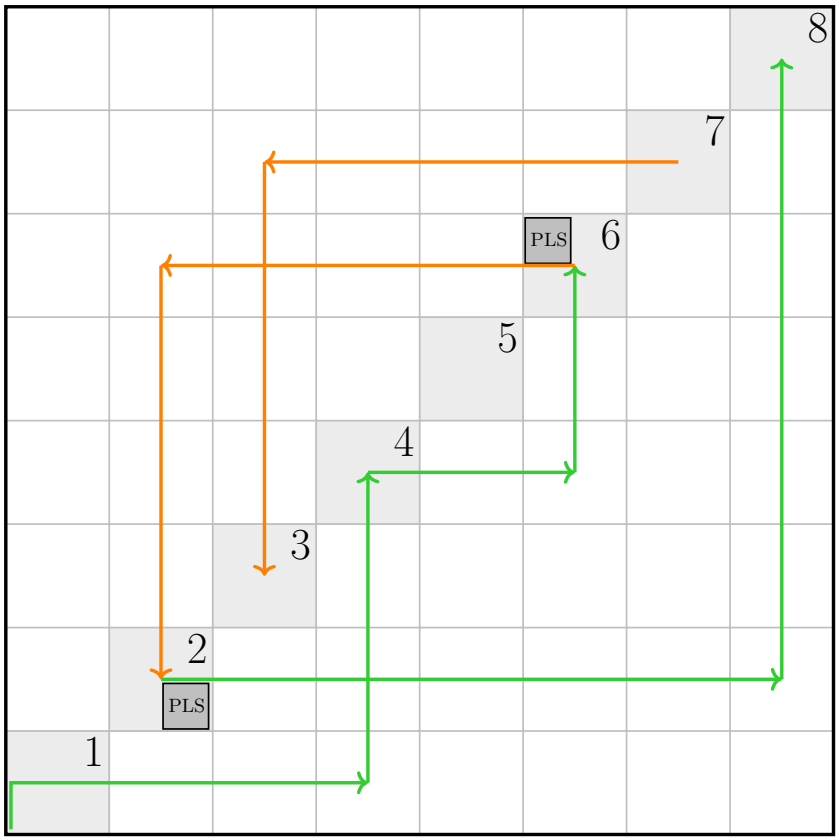


Figure 8.1: A high-level illustration of our construction. The shaded squares on the diagonal correspond to vertices of the graph represented by I^{EOL}, in this case corresponding to the graph in Figure 2.2. The green and orange arrows encode the directed edges of the graph. The positions where I^{ITER} is encoded, i.e., the PLS-Labyrinths, are shown as boxes labelled "PLS". They are located at points where the embedding of I^{EOL} would introduce false solutions, and their purpose is to hide those false solutions by co-locating any such solution with a solution to I^{ITER}.

backward edge, i.e., $v_1 > v_2$, then there will be an "orange path" going from the big square of v_1 to the big square of v_2. These paths are shown in Figure 8.1 for the corresponding example instance of Figure 2.2.

The function f is constructed such that when we move along a green path the value of f decreases. Conversely, when we move along an orange path the value of f increases. Outside the paths, f is defined so as to decrease towards the origin $(0, 0) \in [0, 1]^2$, where the green path corresponding to the source of I^{EOL} starts. As a result, we show that an ε-KKT point can only occur in a big square corresponding to a vertex v of I^{EOL} such that (a) v is a solution of I^{EOL}, or (b) v is *not* a solution of I^{EOL}, but its two neighbours (in the I^{EOL} graph) are both greater than v, or alternatively both less than v. Case (b) exactly corresponds to the case where a green path "meets" an orange path. In that case, it is easy to see that an ε-KKT point is unavoidable.

The PLS-Labyrinth. In order to resolve the issue with case (b) above, we use the following idea: hide the (unavoidable) ε-KKT point in such a way that locating it

requires solving I^{ITER}! This is implemented by introducing a gadget, that we call the PLS-Labyrinth, at the point where the green and orange paths meet (within some big square). An important point is that the PLS-Labyrinth only works properly when it is positioned at such a meeting point. If it is positioned elsewhere, then it will either just introduce additional unneeded ε-KKT points, or even introduce ε-KKT points that are easy to locate. Indeed, if we were able to position the PLS-Labyrinth wherever we wanted, this would presumably allow us to show PLS-hardness, which we do not expect to hold. In Figure 8.1, the positions where a PLS-Labyrinth is introduced are shown as grey boxes labelled "PLS".

Every PLS-Labyrinth is subdivided into exponentially many *medium squares* such that the medium squares on the diagonal (shaded in Figure 8.2) correspond to the nodes of I^{ITER}. The point where the green and orange paths meet, which lies just outside the PLS-Labyrinth, creates an "orange-blue path" which then makes its way to the centre of the medium square for node 1 of I^{ITER}. Similarly, for every node u of I^{ITER} that is a candidate to be a solution (i.e., with $C(u) > u$), there is an orange-blue path starting from the orange path (which runs along the PLS-Labyrinth) and going to the centre of the medium square corresponding to u. Sinks of orange-blue paths introduce ε-KKT points, and so for those u that are not solutions of I^{ITER}, the orange-blue path of u turns into a "blue path" that goes and merges into the orange-blue path of $C(u)$. This ensures that sinks of orange-blue paths (that do not turn into blue paths) exactly correspond to the solutions of I^{ITER}. An interesting point to note is that sources of blue paths do *not* introduce ε-KKT points. This allows us to handle crossings between paths in a straightforward way. Figure 8.2 shows an overview of the PLS-Labyrinth that encodes the ITER example of Figure 2.3.

Bicubic interpolation. Within our construction, we specify how the objective function f behaves within the "small squares" of $[0,1]^2$. At this stage, we have values of f and ∇f at the corners of the small squares, and we then need to smoothly interpolate within the interior of the square. We use bicubic interpolation to do this. It constructs a smooth polynomial over the small square given values for f and ∇f at the square's corners.

We must prove that using bicubic interpolation does not introduce any ε-KKT points within any small square, unless that small square corresponds to a solution of I^{ITER} or I^{EOL}. Each individual small square leads to a different class of polynomials, based on the colour-coding of the grid point, and the direction of the gradient at each grid point. Our construction uses 101 distinct small squares, and we must prove that no unwanted solutions are introduced in any of them. By making use of various symmetries we are able to group these 101 squares into just four different cases for which we can directly verify that the desired statement holds: an ε-KKT point can only appear in a small square that yields a solution of I^{ITER} or I^{EOL}.

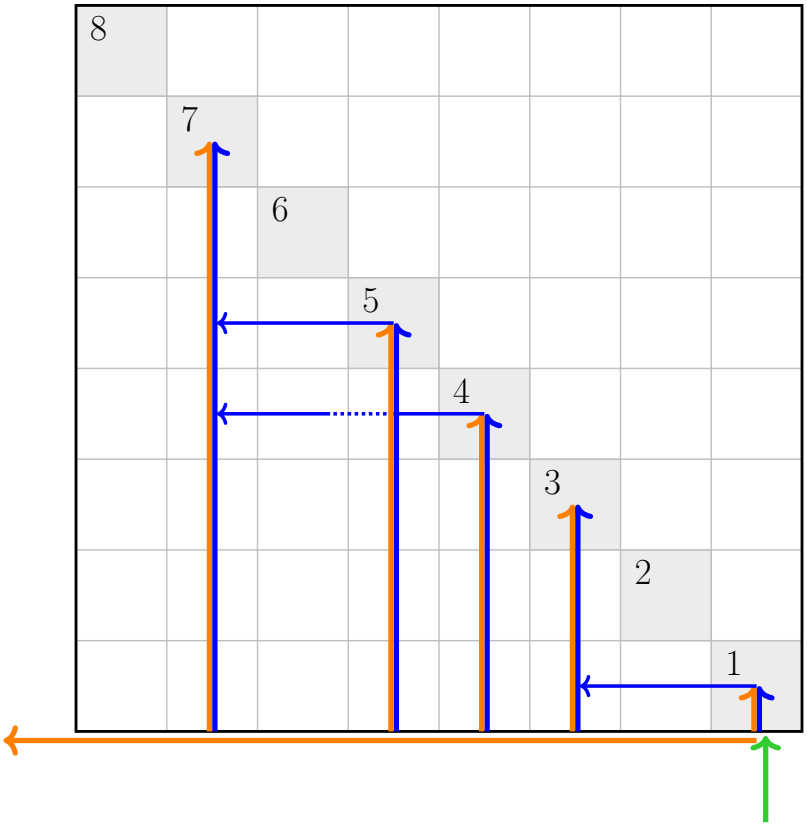


Figure 8.2: High-level illustration of the PLS-Labyrinth corresponding to the ITER example of Figure 2.3. Shaded squares on the diagonal correspond to the nodes of ITER. Colours of lines determine how f is constructed at these points. The horizontal blue lines (pointing left) correspond to the 3 edges in Figure 2.3 that go out from non-solutions, and we do not use similar lines going out from solutions (nodes 3 and 7).

8.2 Defining the Function on the Grid

Consider any instance of END-OF-LINE with 2^n vertices and any instance of ITER with 2^m nodes. We work on the domain $[0, N]^2$ with a grid $G = \{0, 1, 2, \ldots, N\}^2$, where $N = 2^n \cdot 2^{m+4}$. We explain at the end of the chapter how the domain can easily be scaled down to $[0, 1]^2$.

Overview of the embedding. We divide the domain $[0, N]^2$ into $2^n \times 2^n$ big squares. For any $v_1, v_2 \in [2^n]$, let $B(v_1, v_2)$ denote the big square

$$\left[(v_1 - 1)\frac{N}{2^n}, v_1\frac{N}{2^n}\right] \times \left[(v_2 - 1)\frac{N}{2^n}, v_2\frac{N}{2^n}\right].$$

We use the following interpretation: the vertex $v \in [2^n]$ of the END-OF-LINE instance is embedded at the centre of the big square $B(v, v)$. Thus, the vertices are arranged along the main diagonal of the domain. In particular, the trivial source $1 \in [2^n]$

is located at the centre of the big square that lies in the bottom-left corner of the domain and contains the origin.

We seek to embed the edges of the End-of-Line instance in our construction. For every directed edge (v_1, v_2) of the End-of-Line instance, we are going to embed a directed path in the grid G that goes from the centre of $B(v_1, v_1)$ to the centre of $B(v_2, v_2)$. The type of path used and the route taken by the path will depend on whether the edge (v_1, v_2) is a "forward" edge or a "backward" edge. In more detail:

- if $v_1 < v_2$ ("forward" edge), then we will use a so-called *green* path that can only travel to the right and upwards. The path starts at the centre of $B(v_1, v_1)$ and moves to the right until it reaches the centre of $B(v_2, v_1)$. Then, it moves upwards until it reaches its destination: the centre of $B(v_2, v_2)$.

- if $v_1 > v_2$ ("backward" edge), then we will use a so-called *orange* path that can only travel to the left and downwards. The path starts at the centre of $B(v_1, v_1)$ and moves to the left until it reaches the centre of $B(v_2, v_1)$. Then, it moves downwards until it reaches its destination: the centre of $B(v_2, v_2)$.

Figure 8.3 illustrates the high-level idea of the embedding with an example.

For points of the grid G that are part of the "environment," namely that do not lie on a path, the function f will simply be defined by $(x, y) \mapsto x + y$. Thus, if there are no paths at all, the only local minimum of f will be at the origin. However, a green path starts at the origin and this will ensure that there is no minimum there. This green path will correspond to the outgoing edge of the trivial source $1 \in [2^n]$ of the End-of-Line instance.

The green paths will be constructed such that if one moves along a green path the value of f decreases, which means that we are improving the objective function value. Furthermore, the value of f at any point on a green path will be below the value of f at any point in the environment. Conversely, the orange paths will be constructed such that if one moves along an orange path the value of f increases, so the objective function value becomes worse. Additionally, the value of f at any point on an orange path will be above the value of f at any point in the environment.

As a result, if any path starts or ends in the environment, there will be a local minimum or maximum at that point (and thus a KKT point). The only exception is the path corresponding to the outgoing edge of the trivial vertex $1 \in [2^n]$. The start of that path will not create a local minimum or maximum. Thus, in the example of Figure 8.3, there will certainly be KKT points in $B(3, 3)$, $B(7, 7)$ and $B(8, 8)$, but not in $B(1, 1)$.

Recall that every vertex $v \in [2^n]$ has at most one incoming edge and at most one outgoing edge. Thus, for any vertex $v \neq 1$, one of the following cases occurs:

- v is an isolated vertex. In this case, the big square $B(v, v)$ will not contain any path and will fully be in the environment, thus not containing any KKT point. Example: vertex 5 in Figure 8.3.

- v has one outgoing edge and no incoming edge. In this case, the big square $B(v, v)$ will contain the start of a green or orange path. There will be a KKT point at the start of the path, which is fine, since v is a (non-trivial) source of the End-of-Line instance. Example: vertex 7 in Figure 8.3.

- v has one incoming edge and no outgoing edge. In this case, the big square $B(v, v)$ will contain the end of a green or orange path. There will be a KKT point at the end of the path, which is again fine, since v is a sink of the End-of-Line instance. Example: vertices 3 and 8 in Figure 8.3.

- v has one outgoing and one incoming edge. In this case, there are two sub-cases:

  - If both edges yield paths of the same colour, then we will be able to "connect" the two paths at the centre of $B(v, v)$ and avoid introducing a KKT point there. Example: vertex 4 in Figure 8.3.

  - If one of the paths is green and the other one is orange, then there will be a local maximum or minimum in $B(v, v)$ (and thus a KKT point). It is not too hard to see that this is in fact unavoidable. This is where we use the main new "trick" of our reduction: we "hide" the exact location of the KKT point inside $B(v, v)$ in such a way, that finding it requires solving a PLS-complete problem, namely the Iter instance. This is achieved by introducing a new gadget at the point where the two paths meet. We call this the PLS-Labyrinth gadget.

The construction of the green and orange paths is described in detail in Section 8.2.3. The PLS-Labyrinth gadget is described in detail in Section 8.2.4.

8.2.1 Pre-processing

Consider any instance $((S, P), C)$ of Either-Solution(End-of-Line,Iter), i.e., $S, P : [2^n] \to [2^n]$ is an instance of End-of-Line and $C : [2^m] \to [2^m]$ is an instance of Iter. Without loss of generality, we can assume that these instances satisfy the following:

1. The successor and predecessor circuits S, P agree on all edges. Formally, for all $v \in [2^n]$, it holds that

   - if $S(v) \neq v$, then $P(S(v)) = v$, and

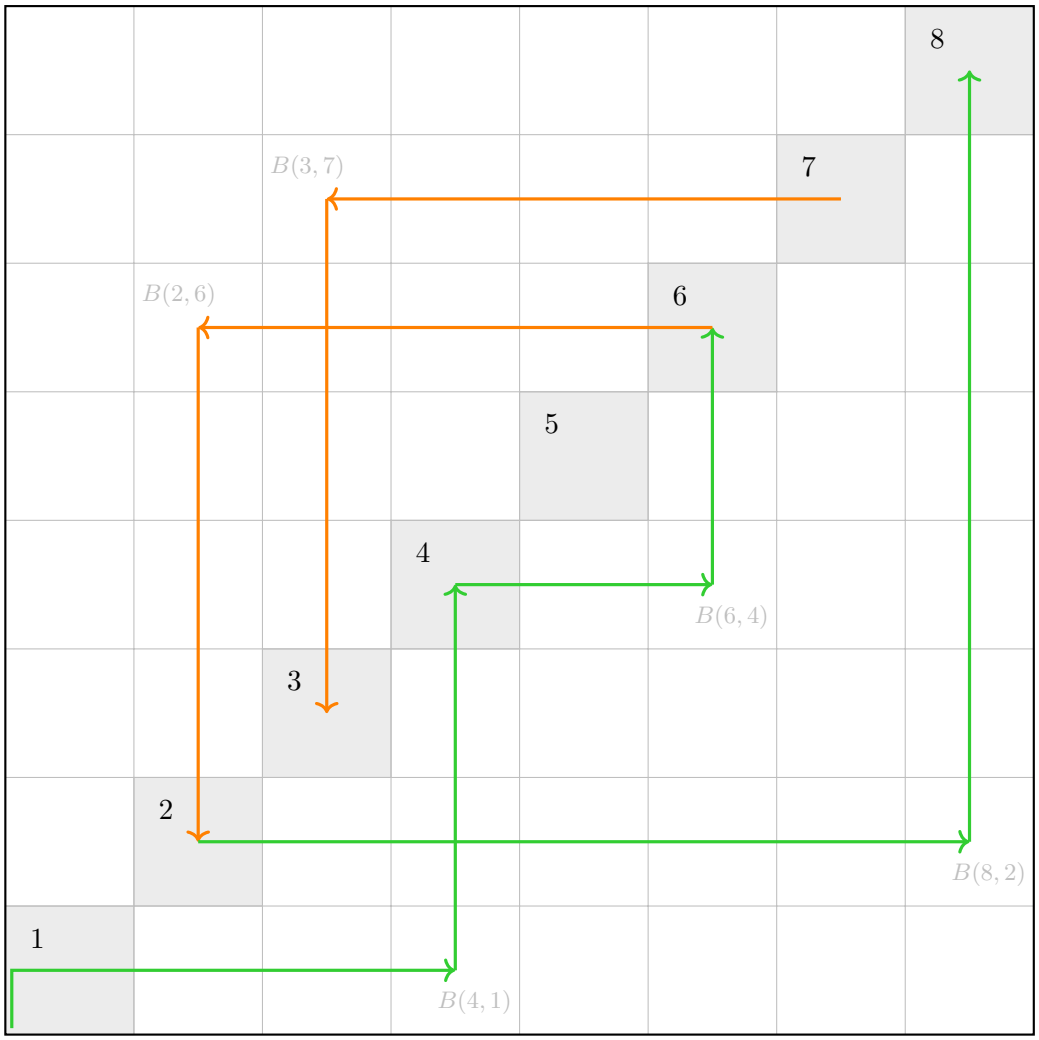


Figure 8.3: Example of the high-level idea for the embedding of an End-of-Line instance in the domain. In this example we are embedding an End-of-Line instance with the set of vertices $[8]$ (i.e., $n = 3$) and the directed edges: $(1, 4)$, $(2, 8)$, $(4, 6)$, $(6, 2)$ and $(7, 3)$ (see Figure 2.2). The domain is divided into 8×8 big squares, and the big squares corresponding to the vertices of the End-of-Line graph are coloured in grey. The solutions of this End-of-Line instance are the vertices 3, 7 and 8.

- if $P(v) \neq v$, then $S(P(v)) = v$.

This property can be ensured by a simple pre-processing step. We modify the circuit S, so that before outputting $S(v)$, it first checks whether $(S(v) \neq v) \wedge (P(S(v)) \neq v)$, and, if this holds, outputs v instead of $S(v)$. It is easy to see that this new circuit for S can be constructed in polynomial time in the size of S and P. We also perform the analogous modification for P. It is easy to check that this does not introduce any new solutions.

2. For all $u \in [2^m]$ we have $C(u) \geq u$. We can ensure that this holds by modifying the circuit C, so that before outputting $C(u)$, it checks whether $C(u) < u$, and, if this is the case, outputs u instead of $C(u)$. Again, the modification can be

done in polynomial time and does not introduce new solutions, nor does it stop the problem from being total.

8.2.2 The value regimes

Recall that we want to specify the value of f and $-\nabla f$ (*the direction of steepest descent*) at all points on the grid $G = \{0, 1, 2, \ldots, N\}^2$, where $N = 2^n \cdot 2^{m+4}$. In order to specify the value of f, it is convenient to define *value regimes*. Namely, if a point $(x, y) \in G$ is in:

- the red value regime, then $f(x, y) := x - y + 4N + 20$.
- the orange value regime, then $f(x, y) := -x - y + 4N + 10$.
- the black value regime, then $f(x, y) := x + y$.
- the green value regime, then $f(x, y) := -x - y - 10$.
- the blue value regime, then $f(x, y) := x - y - 2N - 20$.

Note that at any point on the grid, the value regimes are ordered: red $>$ orange $>$ black $>$ green $>$ blue. Furthermore, it is easy to check that the gap between any two regimes at any point is at least 10. Figure 8.4 illustrates the main properties of the value regimes.

Figure 8.4: The value regimes. On the left, the colours are ordered according to increasing value, from left to right. On the right, we indicate for each value regime, the direction in which it improves, i.e., decreases, in the x-y-plane.

The black value regime will be used for the environment. Thus, unless stated otherwise, every grid point is coloured in black, i.e., belongs to the black value regime. Furthermore, unless stated otherwise, at every black grid point (x, y), the direction of steepest descent, i.e., $-\nabla f(x, y)$, will point to the left.[1] The only exceptions to this are grid points that lie in paths, or grid points that lie on the left boundary of the domain (i.e., $x = 0$).

[1]Notice that that is not exactly the same as the negative gradient of the "black regime function" $(x, y) \mapsto x + y$, which would point south-west. Nevertheless, as we show later, this is enough to ensure that the bicubic interpolation that we use does not introduce any points with zero gradient in a region of the environment.

8.2.3 Embedding the End-of-Line instance: The green and orange paths

Our construction specifies for each grid point a colour (which represents the value of f at that point) and an arrow that represents the direction of $-\nabla f$ at that point. A general "rule" that we follow throughout our construction is that the function values should be consistent with the arrows. For example, if some grid point has an arrow pointing to the right, then the adjacent grid point to the right should have a lower function value, while the adjacent grid point to the left should have a higher function value. This rule is not completely sufficient by itself to avoid KKT points, but it is already a very useful guide.

Recall that the grid $G = \{0, 1, 2, \ldots, N\}^2$ subdivides every big square $B(v_1, v_2)$ into $2^{m+4} \times 2^{m+4}$ small squares. The width of the paths we construct will be two small squares. This corresponds to a width of three grid points.

Green paths. When a green path moves to the right, the two lower grid points will be coloured in green, and the grid point at the top will be in black. Figure 8.5a shows a big square that is traversed by a green path from left to right. Such a big square is said to be of type G1. The black arrows indicate the direction of $-\nabla f$ at every grid point.

When a green path moves upwards, the two right-most grid points will be coloured in green, and the grid point on the left will be in black. Figure 8.5b shows a big square of type G2, namely one that is traversed by a green path from the bottom to the top.

Recall that a green path implementing an edge (v_1, v_2) (where $v_1 < v_2$) comes into the big square $B(v_2, v_1)$ from the left and leaves at the top. Thus, the path has to "turn." Figure 8.5c shows how this turn is implemented. The big square $B(v_2, v_1)$ is said to be of type G3.

If a vertex $v \in [2^n]$ has one incoming edge (v_1, v) and one outgoing edge (v, v_2) such that $v_1 < v < v_2$, then both edges will be implemented by green paths. The green path corresponding to (v_1, v) will enter $B(v, v)$ from the bottom and stop at the centre of $B(v, v)$. The green path corresponding to (v, v_2) will start at the centre of $B(v, v)$ and leave the big square on the right. In order to avoid introducing any KKT points in $B(v, v)$ (since v is not a solution of the End-of-Line instance), we will connect the two paths at the centre of $B(v, v)$. This will be achieved by a simple turn, as shown in Figure 8.5d. The big square $B(v, v)$ is said to be of type G4.

If a vertex $v \in [2^n] \setminus \{1\}$ has one outgoing edge (v, v_2) such that $v < v_2$, and no incoming edge, then this will yield a green path starting at the centre of $B(v, v)$ and going to the right, as shown in Figure 8.5e. The big square $B(v, v)$ is said to be of type G5 in that case. It is not hard to see that there will be a KKT point at the source of that green path. On the other hand, if a vertex $v \in [2^n] \setminus \{1\}$ has one

incoming edge (v_1, v) such that $v_1 < v$, and no outgoing edge, then this will yield a green path coming from the bottom and ending at the centre of $B(v, v)$, as shown in Figure 8.5f. The big square $B(v, v)$ is said to be of type G6 in that case. Again, there will be a KKT point at the sink of that green path.

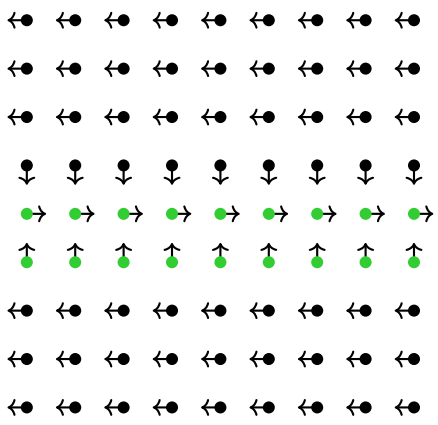
Orange paths. The structure of orange paths is, in a certain sense, symmetric to the structure of green paths. When an orange path moves to the left, the two upper grid points will be coloured in orange, and the grid point at the bottom will be in black. Figure 8.6a shows a big square that is traversed by an orange path from right to left. Such a big square is said to be of type O1.

When an orange path moves downwards, the two left-most grid points will be coloured in orange, and the grid point on the right will be in black. Figure 8.6b shows a big square of type O2, namely one that is traversed by an orange path from top to bottom. Note that the arrows on an orange path essentially point in the *opposite* direction compared to the direction of the path. This is because we want the value to increase (i.e., worsen) when we follow an orange path.

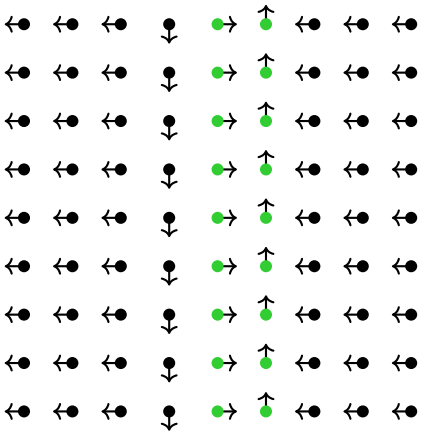
An orange path implementing an edge (v_1, v_2) (where $v_1 > v_2$) comes into the big square $B(v_2, v_1)$ from the right and leaves at the bottom. This turn is implemented as shown in Figure 8.6c. The big square $B(v_2, v_1)$ is said to be of type O3.

If a vertex $v \in [2^n]$ has one incoming edge (v_1, v) and one outgoing edge (v, v_2) such that $v_1 > v > v_2$, then both edges will be implemented by orange paths. The orange path corresponding to (v_1, v) will enter $B(v, v)$ from the top and stop at the centre of $B(v, v)$. The orange path corresponding to (v, v_2) will start at the centre of $B(v, v)$ and leave the big square on the left. As above, we avoid introducing a KKT point by connecting the two paths at the centre of $B(v, v)$. This is achieved by the turn shown in Figure 8.6d. The big square $B(v, v)$ is said to be of type O4.

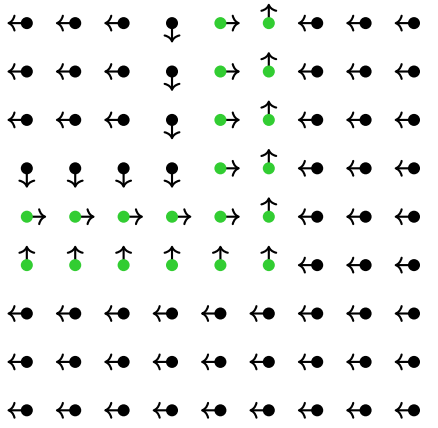
If a vertex $v \in [2^n] \setminus \{1\}$ has one outgoing edge (v, v_2) such that $v > v_2$, and no incoming edge, then this will yield an orange path starting at the centre of $B(v, v)$ and going to the left, as shown in Figure 8.6e. The big square $B(v, v)$ is said to be of type O5 in that case. It is not hard to see that there will be a KKT point at the source of that orange path. On the other hand, if a vertex $v \in [2^n] \setminus \{1\}$ has one incoming edge (v_1, v) such that $v_1 > v$, and no outgoing edge, then this will yield an orange path coming from the top and ending at the centre of $B(v, v)$, as shown in Figure 8.6f. The big square $B(v, v)$ is said to be of type O6 in that case. Again, there will be a KKT point at the sink of that orange path.



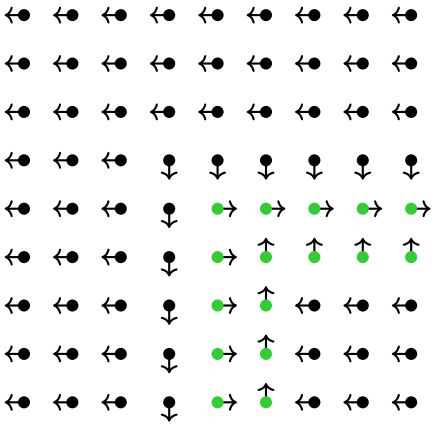
(a) [G1] Green path traversing big square from left to right.



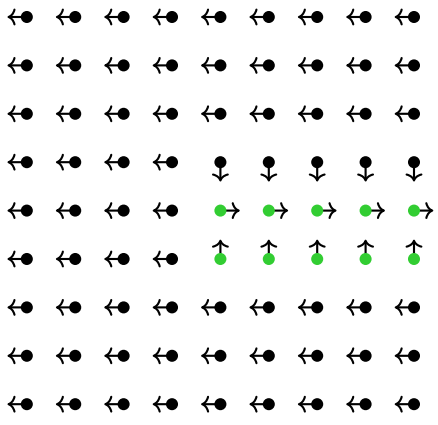
(b) [G2] Green path traversing big square from bottom to top.



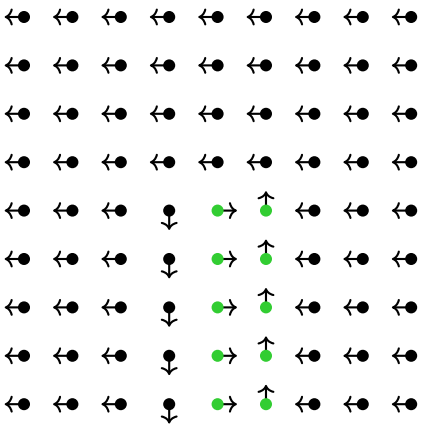
(c) [G3] Green path entering big square from the left, turning, and leaving at the top.



(d) [G4] Green path entering big square from the bottom, turning, and leaving on the right.

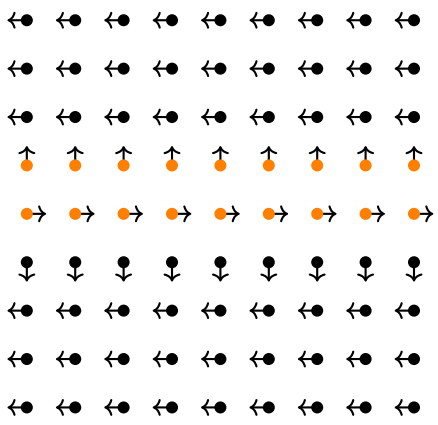


(e) [G5] Source: green path starting at the centre of big square and leaving on the right.

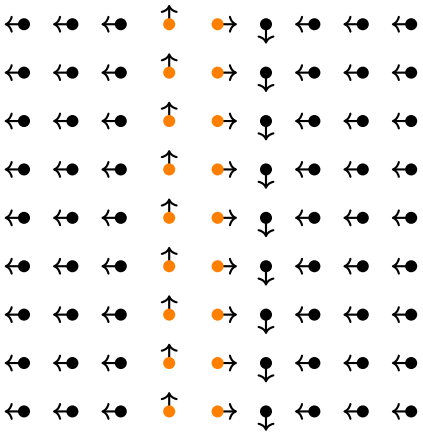


(f) [G6] Sink: green path entering big square from the bottom and ending at the centre.

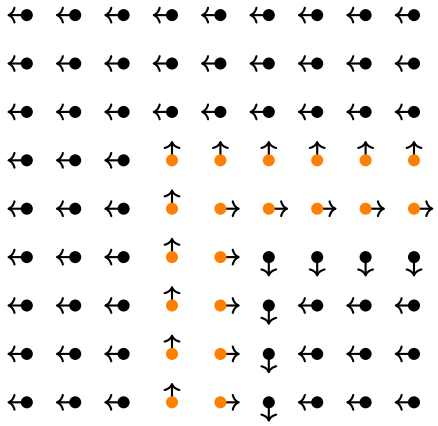
Figure 8.5: Construction of the green paths. The figures show various types of big squares containing different portions of green paths. In these illustrations, the big squares are assumed to have size 8×8 instead of $2^{m+4} \times 2^{m+4}$.



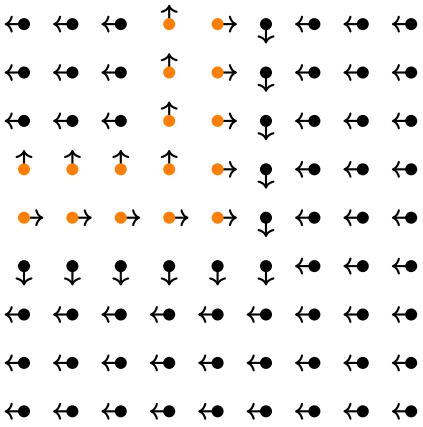
(a) [O1] Orange path traversing big square from right to left.



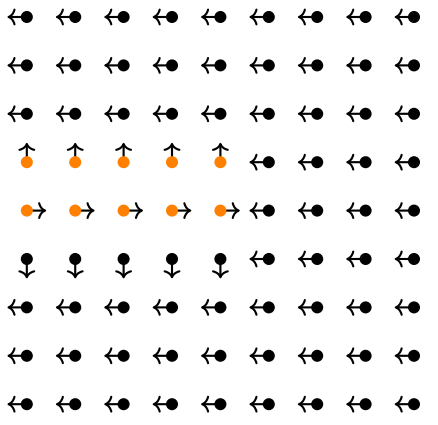
(b) [O2] Orange path traversing big square from top to bottom.



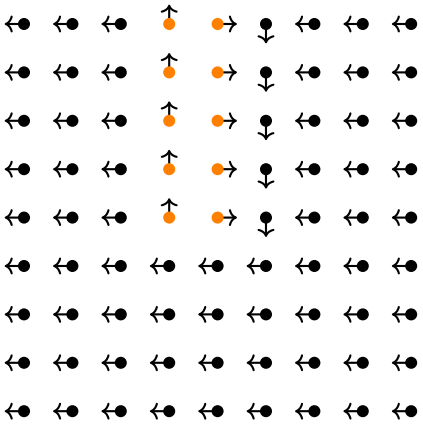
(c) [O3] Orange path entering big square from the right, turning, and leaving at the bottom.



(d) [O4] Orange path entering big square from the top, turning, and leaving on the left.



(e) [O5] Source: orange path starting at the centre of big square and leaving on the left.



(f) [O6] Sink: orange path entering big square from the top and ending at the centre.

Figure 8.6: Construction of the orange paths. The figures show various types of big squares containing different portions of orange paths. In these illustrations, the big squares are assumed to have size 8×8 instead of $2^{m+4} \times 2^{m+4}$.

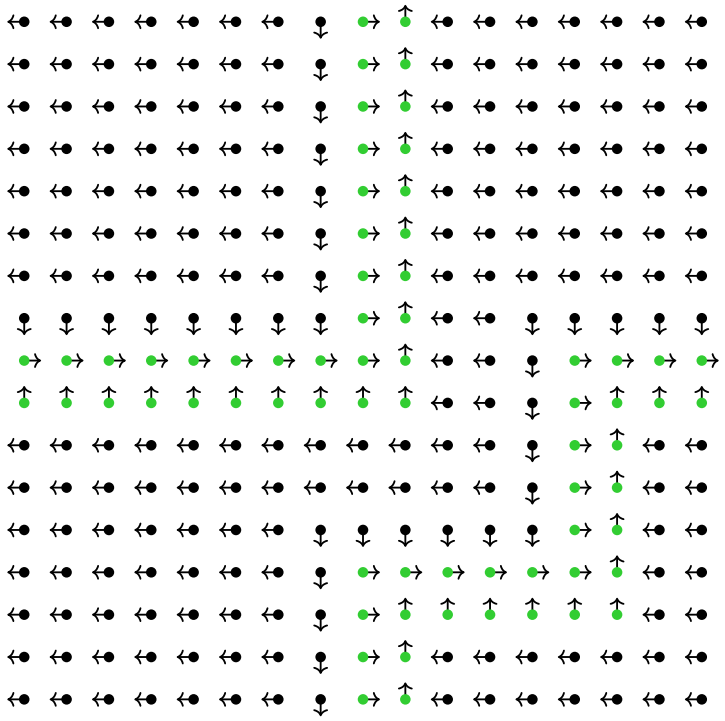
Crossings. Note that, by construction, green paths only exist below the diagonal, and orange paths only exist above the diagonal. Thus, there is no point where an orange path crosses a green path. However, there might exist points where green paths cross, or orange paths cross. First of all, note that it is impossible to have more than two paths traversing a big square, and thus any crossing involves exactly two paths. Furthermore, no crossing can occur in big squares where a "turn" occurs, since, in that case, the turn connects the two paths.

The only way for two green paths to cross is the case where a green path traverses a big square from left to right, and a second green path traverses the same big square from bottom to top. In that case, we say that the big square is of type G7. This problem always occurs when one tries to embed an End-of-Line instance in a two-dimensional domain. Chen and Deng [CD09] proposed a simple, yet ingenious, trick to resolve this issue. The idea is to locally re-route the two paths so that they no longer cross. This modification has the following two crucial properties: a) it is completely local, and b) it does not introduce any new solution (in our case a KKT point). Figure 8.7a shows how this modification is implemented for crossing green paths, i.e., what our construction does for big squares of type G7.

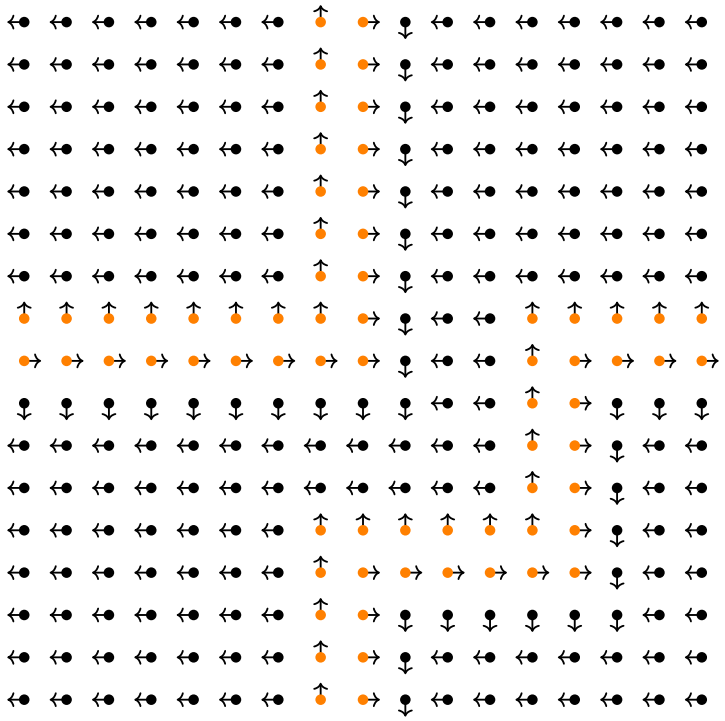
The same issue might arise for orange paths. By the same arguments as above, this can only happen when an orange path traverses a big square from right to left, and a second orange path traverses the same big square from top to bottom. In that case, we say that the big square is of type O7. Figure 8.7b shows how the issue is locally resolved in that case, i.e., what our construction does for big squares of type O7.

Boundary and origin squares. Any big square that is not traversed by any path (including all big squares $B(v, v)$ where v is an isolated vertex of the End-of-Line instance), will have all its grid points coloured in black, and $-\nabla f$ pointing to the left. These big squares, which are said to be of type E1, are as represented in Figure 8.8a. The only exceptions to this rule are the big squares $B(1, v)$ for all $v \in [2^n] \setminus \{1\}$. In those big squares, which are said to be of type E2, the grid points on the left boundary have $-\nabla f$ pointing downwards, instead of to the left. The rest of the grid points have $-\nabla f$ pointing to the left as before. Note that none of these big squares is ever traversed by a path, so they are always as shown in Figure 8.8b.

The big square $B(1, 1)$ is special and we say that it is of type S. Since it corresponds to the trivial source of the End-of-Line instance, it has one outgoing edge (which necessarily corresponds to a green path) and no incoming edge. Normally, this would induce a KKT point at the centre of $B(1, 1)$ (as in Figure 8.5e). Furthermore, recall that, by the definition of the black value regime, there must also be a KKT point at the origin, if it is coloured in black. By a careful construction (which is very similar to the one used by Hubácek and Yogev [HY20] for Continuous-Localopt)

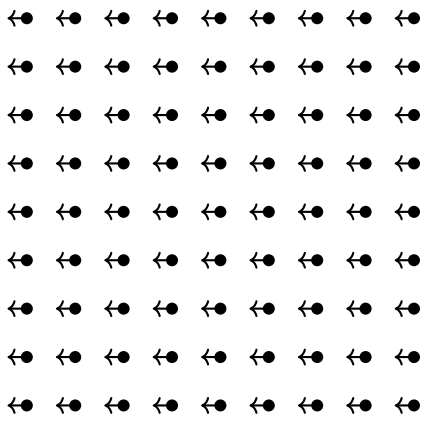


(a) [G7] Crossing of green paths.

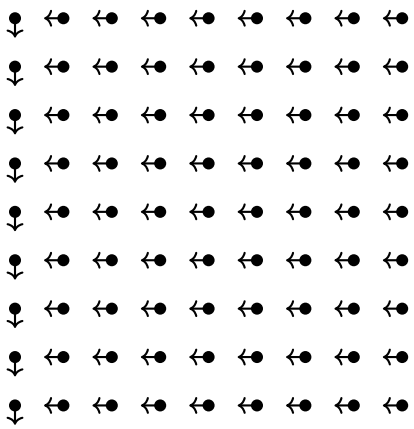


(b) [O7] Crossing of orange paths.

Figure 8.7: Crossing gadgets for green and orange paths. In these two illustrations, the big squares are assumed to have size 16×16 instead of $2^{m+4} \times 2^{m+4}$.



(a) [E1] Big square not traversed by any path.



(b) [E2] Big square on left boundary of domain.

Figure 8.8: Big squares not traversed by any path. In these two illustrations, the big squares are assumed to have size 8×8 instead of $2^{m+4} \times 2^{m+4}$.

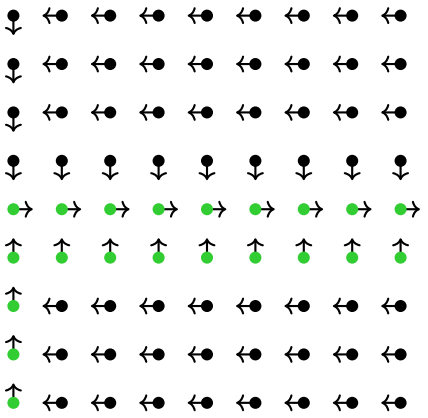


Figure 8.9: [S] Construction for big square $B(1,1)$ (for size 8×8 instead of $2^{m+4} \times 2^{m+4}$).

we can ensure that these two KKT points neutralise each other. In other words, instead of two KKT points, there is no KKT point at all in $B(1,1)$. The construction for $B(1,1)$ is shown in Figure 8.9.

Figure 8.10 shows the whole construction for a small example where $n = 1$ and big squares have size 8×8 (instead of $2^{m+4} \times 2^{m+4}$).

Green and orange paths meeting. Our description of the construction is almost complete, but there is one crucial piece missing. Indeed, consider any vertex v that has one incoming edge (v_1, v) and one outgoing edge (v, v_2) such that: A) $v_1 < v$ and $v_2 < v$, or B) $v_1 > v$ and $v_2 > v$. As it stands, a green path and an orange path meet at the centre of $B(v,v)$ which means that there is a local minimum or maximum at the centre of $B(v,v)$, and thus a KKT point. However, v is not a solution to the End-of-Line instance. Even though we cannot avoid having a KKT point in $B(v,v)$, we can

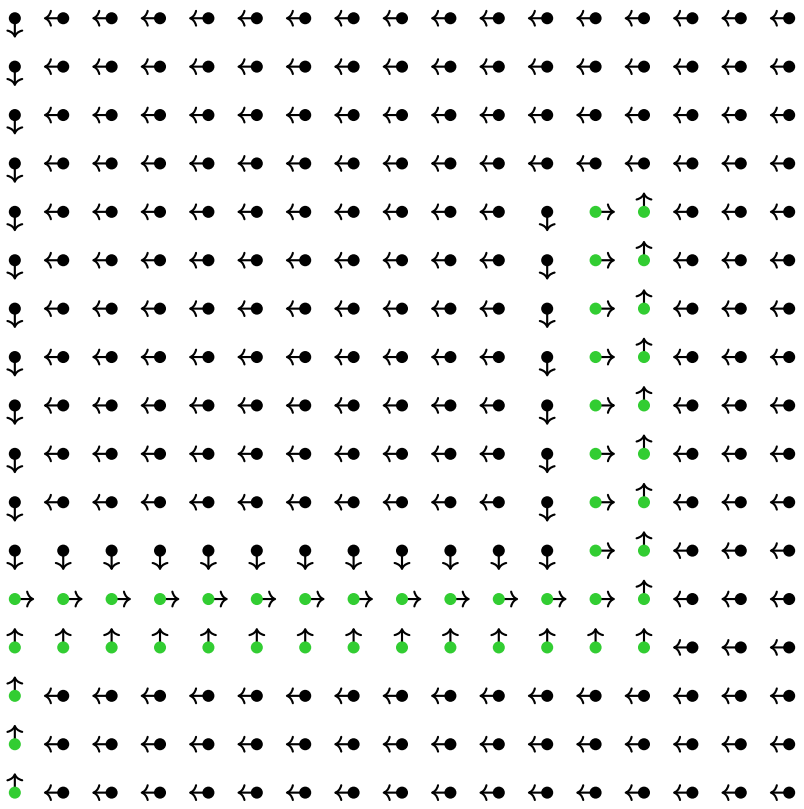


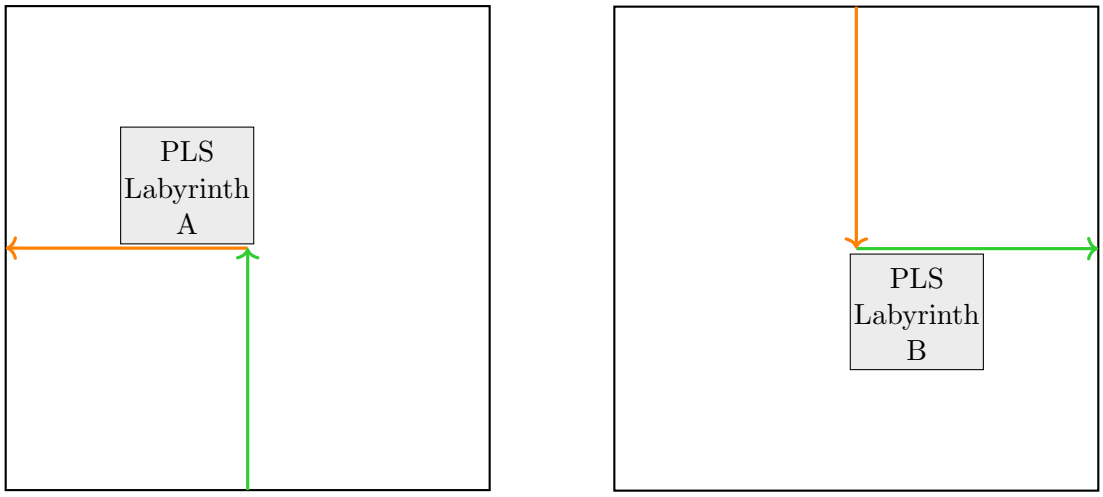
Figure 8.10: [X] Full construction for a small example, in particular showing the whole boundary. Here $n = 1$ and big squares have size 8×8 (instead of $2^{m+4} \times 2^{m+4}$).

"hide" it, so that finding it requires solving the Iter instance. This is implemented by constructing a PLS-Labyrinth gadget at the point where the green and orange paths meet. Figures 8.11a and 8.11b show where this PLS-Labyrinth gadget is positioned inside a big square of type LA (namely when case A above occurs) and a big square of type LB (namely when case B above occurs) respectively. The PLS-Labyrinth gadget can only be positioned at a point where a green path and an orange path meet. In particular, it cannot be used to "hide" a KKT point occurring at a source or sink of a green or orange path, i.e., at a solution of the End-of-Line instance.

In our construction, every big square is of type G1, G2, ..., G7, O1, O2, ..., O7, E1, E2, S, LA or LB. Note that we can efficiently determine the type of a given big square, if we have access to the End-of-Line circuits S and P.

8.2.4 Embedding the Iter instance: The PLS-Labyrinth

PLS-Labyrinth. We begin by describing the PLS-Labyrinth gadget for case A, i.e., v has one incoming edge (v_1, v) and one outgoing edge (v, v_2) such that $v_1 < v$ and $v_2 < v$. In particular, $B(v, v)$ is of type LA. The PLS-Labyrinth gadget comprises $2^{m+2} \times 2^{m+2}$ small squares and is positioned in the big square $B(v, v)$ as shown in



(a) [LA] Position of PLS-Labyrinth gadget in big square of type LA.

(b) [LB] Position of PLS-Labyrinth gadget in big square of type LB.

Figure 8.11: Position of PLS-Labyrinth gadget in big squares of type LA and LB.

Figure 8.11a. Note, in particular, that the bottom side of the gadget is adjacent to the orange path, and the bottom-right corner of the gadget lies just above the point where the green and orange paths intersect (which occurs at the centre of $B(v, v)$). Finally, observe that since $B(v, v)$ has $2^{m+4} \times 2^{m+4}$ small squares, there is enough space for the PLS-Labyrinth gadget.

For convenience, we subdivide the PLS-Labyrinth gadget into $2^m \times 2^m$ medium squares. Thus, every medium square is made out of 4×4 small squares. We index the medium squares as follows: for $u_1, u_2 \in [2^m]$, let $M(u_1, u_2)$ denote the medium square that is the u_2th from the bottom and the u_1th from the *right*. Thus, $M(1, 1)$ corresponds to the medium square that lies at the bottom-right of the gadget (and is just above the intersection of the paths). Our construction will create the following paths inside the PLS-Labyrinth gadget:

- For every $u \in [2^m]$ such that $C(u) > u$, there is an orange-blue path starting at $M(u, 1)$ and moving upwards until it reaches $M(u, u)$.
- For every $u \in [2^m]$ such that $C(u) > u$ and $C(C(u)) > C(u)$, there is a blue path starting at $M(u, u)$ and moving to the left until it reaches $M(C(u), u)$.

Figure 8.12 shows a high-level overview of how the Iter instance is embedded in the PLS-Labyrinth. Note that if $C(u) > u$ and $C(C(u)) > C(u)$, then the blue path starting at $M(u, u)$ will move to the left until $M(C(u), u)$ where it will reach the orange-blue path moving up from $M(C(u), 1)$ to $M(C(u), C(u))$ (which exists since $C(C(u)) > C(u)$). Thus, every blue path will always "merge" into some orange-blue path. On the other hand, some orange-blue paths will stop in the environment without merging into any other path. Consider any $u \in [2^m]$ such that $C(u) > u$.

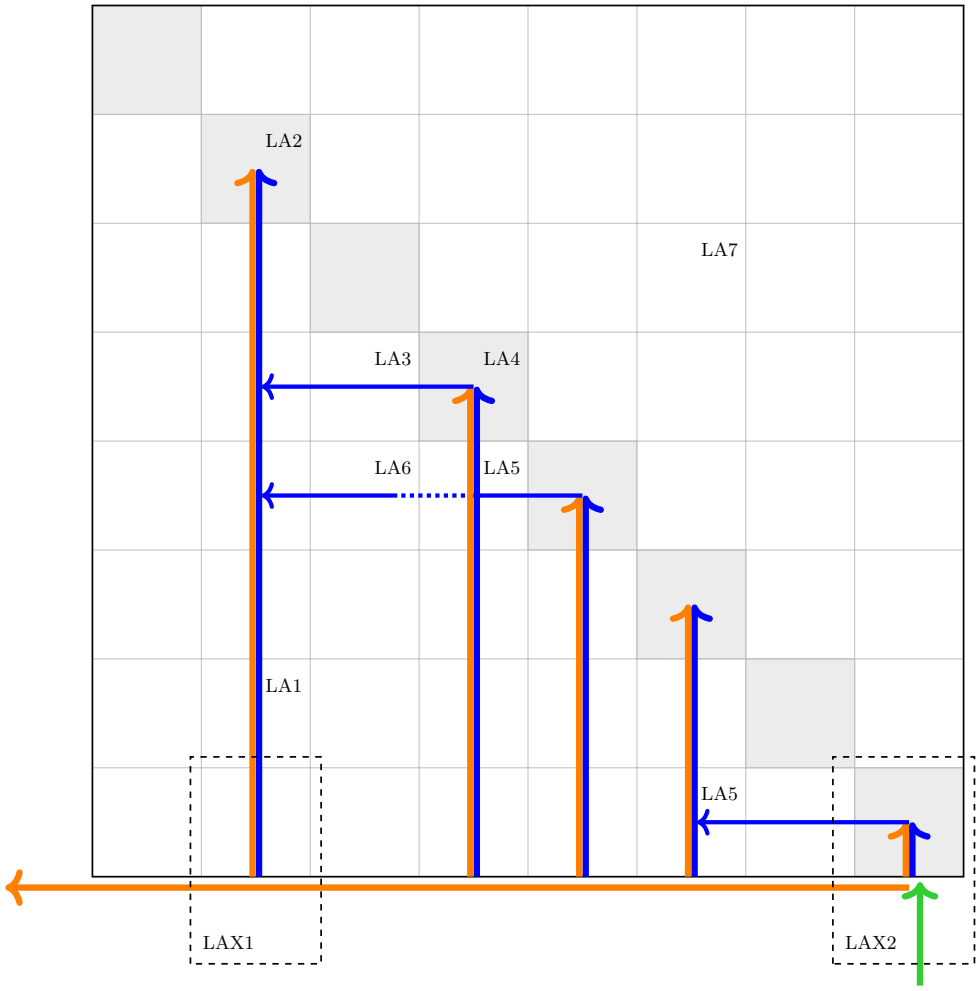


Figure 8.12: Map of the PLS-Labyrinth for case A corresponding to the ITER example of Figure 2.3. Shaded squares are the medium squares corresponding to the nodes of ITER. The horizontal blue lines (pointing left) correspond to the 3 edges in Figure 2.3 that go out from non-solutions, and we do not use similar lines going out from solutions (nodes 3 and 7). We have also indicated the parts LA1-LA6, and LAX1-LAX2, that are constructed in Figure 8.13.

The orange-blue path for u stops at $M(u,u)$. If $C(C(u)) > C(u)$, then there is a blue path starting there, so the orange-blue path "merges" into the blue path. However, if $C(C(u)) \leq C(u)$, i.e., $C(C(u)) = C(u)$, there is no blue path starting at $M(u,u)$ and the orange-blue path just stops in the environment. Thus, the only place in the PLS-Labyrinth where a path can stop in the environment is in a medium square $M(u,u)$ such that $C(u) > u$ and $C(C(u)) = C(u)$. This corresponds exactly to the solutions of the ITER instance C. In our construction, we will ensure that KKT points can indeed only occur at points where a path stops without merging into any other path.

Orange-blue paths. An orange-blue path moves from $M(u,1)$ upwards to $M(u,u)$ (for some $u \in [2^m]$ such that $C(u) > u$) and has a width of two small squares. The left-

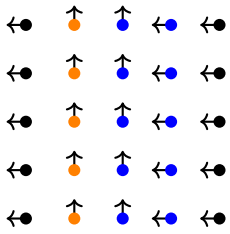
most point is coloured in orange and the two points on the right are blue. Figure 8.13a shows a medium square that is being traversed by the orange-blue path, i.e., a medium square $M(u, w)$ where $w < u$. We say that such a medium square $M(u, w)$ is of type LA1. When the orange-blue path reaches $M(u, u)$, it either "turns" to the left and creates the beginning of a blue path (medium square of type LA4, Figure 8.13d), or it just stops there (medium square of type LA2, Figure 8.13b). The case where the orange-blue path just stops, occurs when there is no blue path starting at $M(u, u)$. Note that, in that case, u is a solution of the ITER instance, and so it is acceptable for a medium square of type LA2 to contain a KKT point.

The orange-blue path begins in $M(u, 1)$, which lies just above the orange path. In fact, the beginning of the orange-blue path is adjacent to the orange path as shown in Figure 8.13g. This is needed, since if the orange-blue path started in the environment, the point coloured orange would yield a local maximum and thus a KKT point.

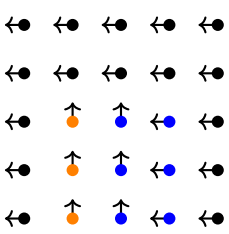
The beginning of the orange-blue path for $u = 1$ is special, since, in a certain sense, this path is created by the intersection of the green and orange paths. Figure 8.13h shows how the intersection is implemented and how exactly it is adjacent to $M(1, 1)$. Note that $M(1, 1)$ is just a standard "turn," i.e., a medium square of type LA4.

Blue paths. A blue path starts in $M(u, u)$ for some $u \in [2^m]$ such that $C(u) > u$ and $C(C(u)) > C(u)$. It moves from right to left and has a width of two small squares. All three points on the path are coloured blue and the direction of steepest descent points to the left. Figure 8.13c shows a medium square traversed by a blue path. Such a medium square is said to be of type LA3. As mentioned above, the blue path starts at $M(u, u)$ which is of type LA4 (a "turn"). When the blue path reaches $M(C(u), u)$, it merges into the orange-blue path going from $M(C(u), 1)$ to $M(C(u), C(u))$. This merging is straightforward and is implemented as shown in Figure 8.13f. The medium square $M(C(u), u)$ is then said to be of type LA5.

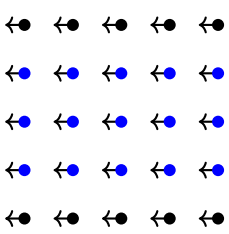
Crossings. Note that two orange-blue paths cannot cross, and similarly two blue paths can also not cross. However, a blue path going from $M(u, u)$ to $M(C(u), u)$ can cross many other orange-blue paths, before it reaches and merges into its intended orange-blue path. Fortunately, these crossings are much easier to resolve than earlier. Indeed, when a blue path is supposed to cross an orange-blue path, it can simply merge into it and restart on the other side. The important thing to note here is that, while a blue path cannot stop in the environment (without creating a KKT point), it can *start* in the environment. Figure 8.13e shows how this is implemented. In particular, we use a medium square of type LA5 for the merging, and a medium square of type LA6 for the re-start of the blue path.



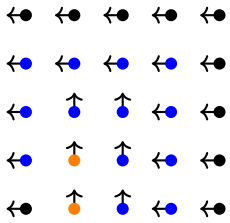
(a) [LA1] Orange-blue path traversing medium square.



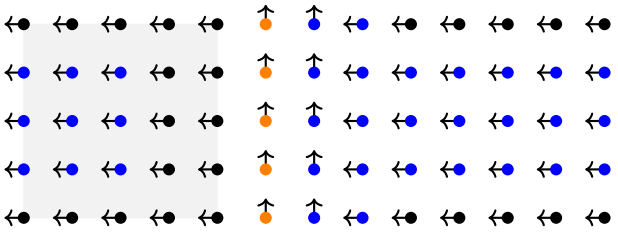
(b) [LA2] Orange-blue path ending in medium square.



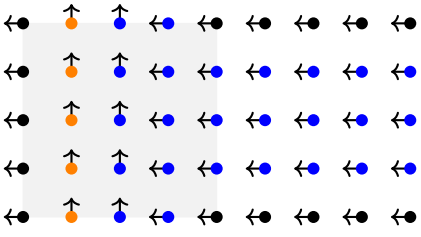
(c) [LA3] Blue path traversing medium square.



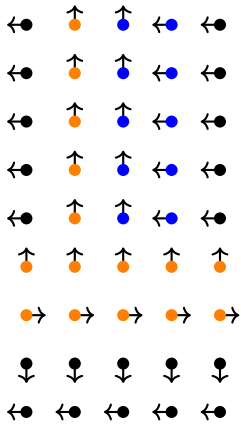
(d) [LA4] Orange-blue path turning in medium square and creating start of blue path.



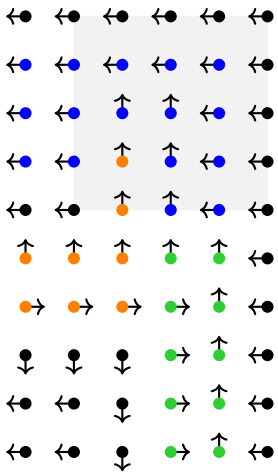
(e) [LA6] Blue path crossing over orange-blue path. Medium square of type LA6 indicated in grey.



(f) [LA5] Blue path merging into orange-blue path. Medium square of type LA5 indicated in grey.



(g) [LAX1] Start of orange-blue path, adjacent to orange path.



(h) [LAX2] Start of orange-blue path for $u = 1$, adjacent to the intersection of green and orange path. $M(1, 1)$ is indicated in grey.

Figure 8.13: Construction of blue and orange-blue paths in the PLS-Labyrinth gadget inside a big square of type LA.

Note that if the blue path has to cross more than one orange-blue path in immediate succession, then it will simply merge into the first one it meets, and restart after the last one (i.e., as soon as it reaches a medium square that is not traversed by an orange-blue path).

Finally, we say that a medium square is of type LA7, if it does not contain any path at all. Medium squares of type LA7 are like the environment, i.e., all the grid points are coloured black and the arrows of steepest descent point to the left. In our construction, every medium square in the PLS-Labyrinth gadget is of type LA1, LA2, ..., LA6, or LA7. It is easy to check that the type of a given medium square can be determined efficiently, given access to the ITER circuit C.

The PLS-Labyrinth gadget for case B is, in a certain sense, symmetric to the one presented above. Indeed, it suffices to perform a point reflection (in other words, a rotation by 180 degrees) with respect to the centre of $B(v, v)$, and a very simple transformation of the colours. With regards to the final interpolated function, this corresponds to rotating $B(v, v)$ by 180 degrees around its centre and multiplying the output of the function by -1. Let $\phi : B(v, v) \to B(v, v)$ denote rotation by 180 degrees around the centre of $B(v, v)$. Then, the direction of steepest descent at some grid point $(x, y) \in B(v, v)$ in case B is simply the same as the direction of steepest descent at $\phi(x, y)$ in case A. The colour of (x, y) in case B is obtained from the colour of $\phi(x, y)$ in case A as follows:

- black remains black,
- green becomes orange, and vice-versa,
- blue becomes red, and vice-versa.

Figure 8.14 shows a high-level overview of the PLS-Labyrinth gadget for case B. We obtain corresponding medium squares of type LB1, LB2, ..., LB7. The analogous illustrations for case B are shown in Figure 8.15.

8.3 Extending the Function to the Rest of the Domain

Up to this point we have defined the function f and the direction of its gradient at all grid points of G. In order to extend f to the whole domain $[0, N]^2$, we use *bicubic interpolation* (see, e.g., [Rus95] or the corresponding Wikipedia article[2]). Note that the more standard and simpler *bilinear* interpolation (used in particular by Hubácek

[2] https://en.wikipedia.org/wiki/Bicubic_interpolation

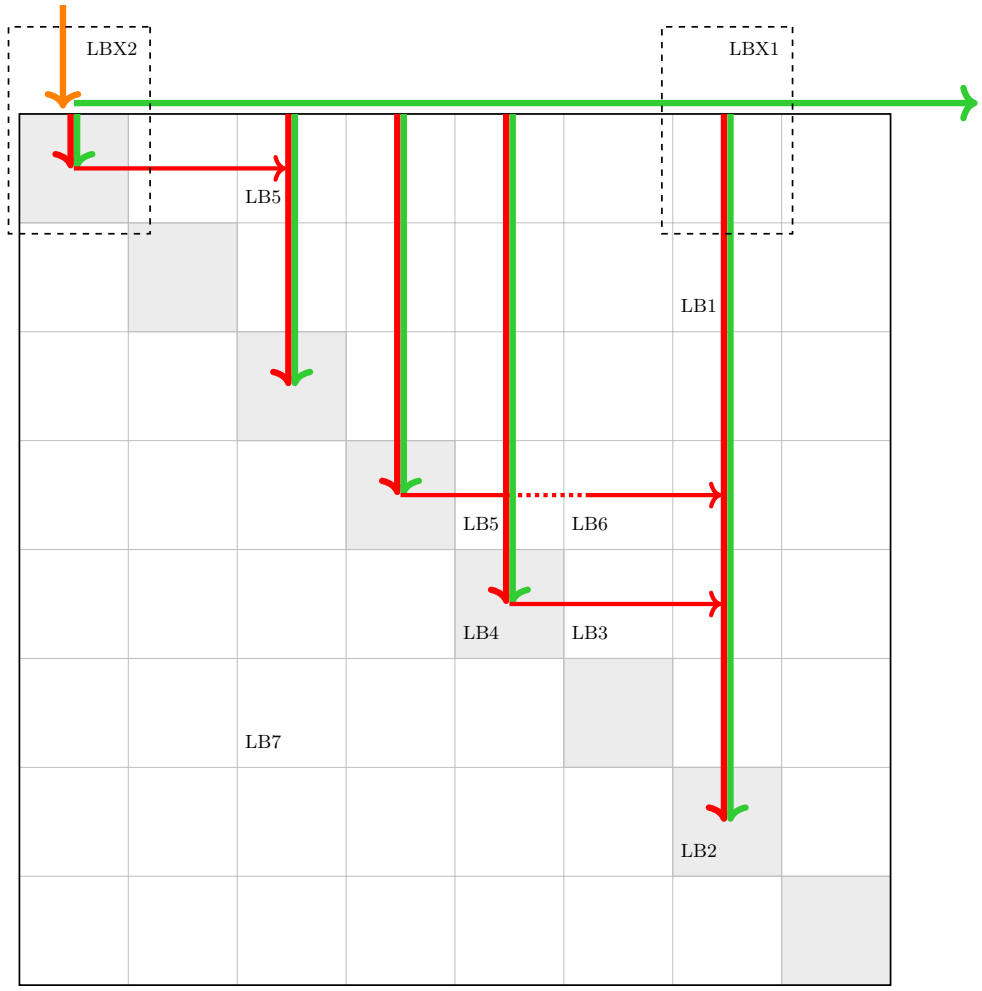


Figure 8.14: Map of the PLS-Labyrinth for case B corresponding to the ITER example of Figure 2.3. Shaded squares are the medium squares corresponding to the nodes of ITER. We have also indicated the parts LB1-LB6, and LBX1-LBX2, that are constructed in Figure 8.15.

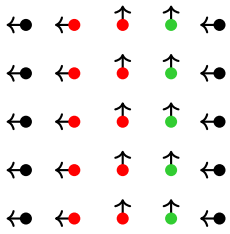
and Yogev [HY20]) yields a continuous function, but not necessarily a continuously differentiable function. On the other hand, bicubic interpolation ensures that the function will indeed be continuously differentiable over the whole domain $[0, N]^2$.

We use bicubic interpolation in every small square of the grid G. Consider any small square and let $(x, y) \in [0, 1]^2$ denote the local coordinates of a point inside the square. Then, the bicubic interpolation inside this square will be a polynomial of the form:

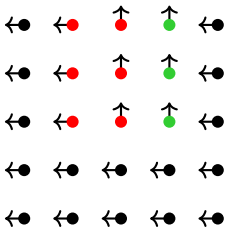
$$f(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j \tag{8.1}$$

where the coefficients a_{ij} are computed as follows

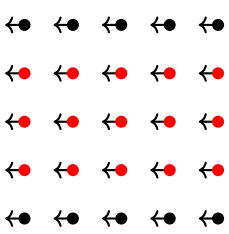
$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{8.2}$$



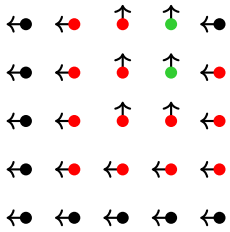
(a) [LB1] Red-green path traversing medium square.



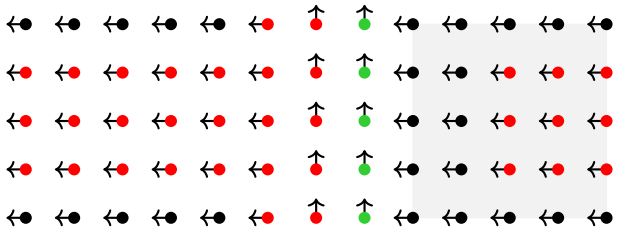
(b) [LB2] Red-green path ending in medium square.



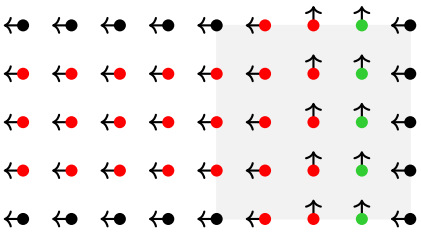
(c) [LB3] Red path traversing medium square.



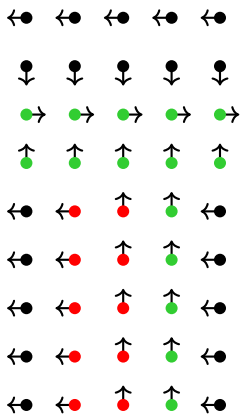
(d) [LB4] Red-green path turning in medium square and creating start of red path.



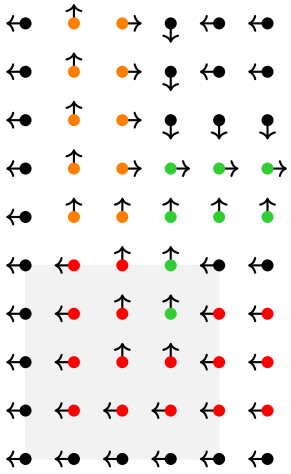
(e) [LB6] Red path crossing over red-green path. Medium square of type LB6 indicated in grey.



(f) [LB5] Red path merging into red-green path. Medium square of type LB5 indicated in grey.



(g) [LBX1] Start of red-green path, adjacent to green path.



(h) [LBX2] Start of red-green path for $u = 1$, adjacent to the intersection of orange and green path. $M(1, 1)$ is indicated in grey.

Figure 8.15: Construction of red and red-green paths in the PLS-Labyrinth gadget inside a big square of type LB.

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} f(0,0) & f(0,1) & f_y(0,0) & f_y(0,1) \\ f(1,0) & f(1,1) & f_y(1,0) & f_y(1,1) \\ f_x(0,0) & f_x(0,1) & f_{xy}(0,0) & f_{xy}(0,1) \\ f_x(1,0) & f_x(1,1) & f_{xy}(1,0) & f_{xy}(1,1) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Here f_x and f_y denote the partial derivatives with respect to x and y respectively. Similarly, f_{xy} denotes the second order partial derivative with respect to x and y. It remains to explain how we set the values of f, f_x, f_y and f_{xy} at the four corners of the square:

- The values $f(0,0), f(0,1), f(1,0)$ and $f(1,1)$ are set according to the value regimes in our construction.
- The values of $f_x(0,0)$, $f_x(0,1)$, $f_x(1,0)$, $f_x(1,1)$, $f_y(0,0)$, $f_y(0,1)$, $f_y(1,0)$ and $f_y(1,1)$ are set based on the direction of steepest descent $(-\nabla f)$ in our construction, with a length multiplier of $\delta = 1/2$. For example, if the arrow of steepest descent at $(0,1)$ is pointing to the left, then we set $f_x(0,1) = \delta$ and $f_y(0,1) = 0$. If it is pointing up, then we set $f_x(0,1) = 0$ and $f_y(0,1) = -\delta$.
- We always set $f_{xy}(0,0) = f_{xy}(0,1) = f_{xy}(1,0) = f_{xy}(1,1) = 0$.

By using this interpolation procedure in each small square, we obtain a function $f : [0,N]^2 \to \mathbb{R}$. In fact, we can even extend the function to points $(x,y) \in \mathbb{R}^2 \setminus [0,N]^2$ by simply using the interpolated polynomial obtained for the small square that is closest to (x,y). This will be done automatically by our construction of the arithmetic circuit computing f and it will ensure that the gradient is well-defined even on the boundary of $[0,N]^2$.

Lemma 8.1. *The function $f : \mathbb{R}^2 \to \mathbb{R}$ we obtain by bicubic interpolation has the following properties:*

- *It is continuously differentiable on $\mathbb{R}^2$;*
- *f and its gradient ∇f are Lipschitz-continuous on $[0,N]^2$ with Lipschitz-constant $L = 2^{18}N$;*
- *Well-behaved arithmetic circuits computing f and ∇f can be constructed in polynomial time (in the size of the circuits S, P and C).*

Proof. Regarding the first point, see, e.g., [Rus95].

Lipschitz-continuity. In order to prove the second point, we first show that f and ∇f are L-Lipschitz-continuous in every small square of the grid. Consider any small square. In our construction, the values of f, f_x, f_y, f_{xy} used in the computation of the coefficients a_{ij} are clearly all upper bounded by $2^3 N$ in absolute value. Thus, using Equation (8.2), it is easy to check that $|a_{ij}| \leq 2^{10}N$ for all $i,j \in \{0,1,2,3\}$.

Furthermore, note that the partial derivatives of f inside the small square can be expressed as:

$$\frac{\partial f}{\partial x}(x,y) = \sum_{i=1}^{3}\sum_{j=0}^{3} i \cdot a_{ij} x^{i-1} y^j \qquad \frac{\partial f}{\partial y}(x,y) = \sum_{i=0}^{3}\sum_{j=1}^{3} j \cdot a_{ij} x^i y^{j-1} \tag{8.3}$$

using the local coordinates $(x,y) \in [0,1]^2$ inside the small square. Finally, it is easy to check that the monomials $x^i y^j$, $i,j \in \{0,1,2,3\}$, are all 6-Lipschitz continuous over $[0,1]^2$. Putting everything together and using Equation (8.1) and Equation (8.3), it follows that f and ∇f are Lipschitz-continuous (w.r.t. the ℓ_2-norm) with Lipschitz constant $L = 2^{18}N$ inside the small square. Note that the change from local coordinates to standard coordinates is just a very simple translation that does not impact the Lipschitzness of the functions.

Since f and ∇f are L-Lipschitz-continuous inside every small square and continuous over all of $[0,N]^2$, it follows that they are in fact L-Lipschitz-continuous over the whole domain $[0,N]^2$. Indeed, consider any points $z_1, z_2 \in [0,N]^2$. Then, there exists $\ell \in \mathbb{N}$ such that the segment $z_1 z_2$ can be subdivided into $z_1 w_1 w_2 \dots w_\ell z_2$ so that each of the segments $z_1 w_1$, $w_1 w_2, \dots,$ $w_{\ell-1} w_\ell$, $w_\ell z_2$ lies within a small square. For ease of notation, we let $w_0 := z_1$ and $w_{\ell+1} := z_2$. Then, we can write

$$\|\nabla f(z_1) - \nabla f(z_2)\| \le \sum_{i=0}^{\ell} \|\nabla f(w_i) - \nabla f(w_{i+1})\| \le L \sum_{i=0}^{\ell} \|w_i - w_{i+1}\| = L\|z_1 - z_2\|$$

where we used the fact that $\sum_{i=0}^{\ell} \|w_i - w_{i+1}\| = \|z_1 - z_2\|$, since $w_0 w_1 w_2 \dots w_\ell w_{\ell+1}$ is just a partition of the segment $z_1 z_2$. The exact same argument also works for f.

Arithmetic circuits. Before showing how to construct the arithmetic circuits for f and ∇f, we first construct a Boolean circuit B that will be used as a sub-routine. The Boolean circuit B receives as input a point (x,y) on the grid $G = \{0,1,\dots,N\}^2$ and outputs the colour (i.e., value regime) and steepest descent arrow at that point. It is not too hard to see that the circuit B can be constructed in time that is polynomial in the sizes of the circuits S, P and C. In more detail, it performs the following operations:

1. Compute End-of-Line-vertices $v_1, v_2 \in [2^n]$ such that (x,y) lies in the big square $B(v_1, v_2)$.

2. Using the End-of-Line circuits S and P determine the exact type of $B(v_1, v_2)$, namely one of the following: G1-G7, O1-O7, E1, E2, S, LA or LB.

3. If the type of $B(v_1, v_2)$ is not LA or LB, then we know the exact structure of $B(v_1, v_2)$ and can easily return the colour and arrow at (x,y).

4. If the type of $B(v_1, v_2)$ is LA or LB, then first determine whether (x, y) lies in the PLS-Labyrinth inside $B(v_1, v_2)$ or not.

5. If (x, y) does not lie in the PLS-Labyrinth, then we can easily determine the colour and arrow at (x, y), since we know the exact structure of $B(v_1, v_2)$ except the inside of the PLS-Labyrinth.

6. If (x, y) lies in the PLS-Labyrinth, then we can compute ITER-vertices $u_1, u_2 \in [2^m]$ such that (x, y) lies in the medium square $M(u_1, u_2)$ of the PLS-Labyrinth inside $B(v_1, v_2)$.

7. Using the ITER circuit C determine the type of $M(u_1, u_2)$, namely one of the following: LA1-LA7, LB1-LB7. Given the type of $M(u_1, u_2)$, we then know the exact structure of $M(u_1, u_2)$ and can in particular determine the colour and arrow at (x, y).

The arithmetic circuits for f and ∇f are then constructed to perform the following operations on input $(x, y) \in [0, N]^2$:

1. Using the comparison gate $<$ and binary search, compute the bits representing $(\widehat{x}, \widehat{y}) \in \{0, 1, \ldots, N-1\}^2$: a grid point such that (x, y) lies in the small square that has $(\widehat{x}, \widehat{y})$ as its bottom left corner.

2. Simulate the Boolean circuit B using arithmetic gates to compute (a bit representation) of the colour and arrow at the four corners of the small square, namely $(\widehat{x}, \widehat{y})$, $(\widehat{x}+1, \widehat{y})$,$(\widehat{x}, \widehat{y}+1)$ and $(\widehat{x}+1, \widehat{y}+1)$.

3. Using this information and the formulas for the value regimes, compute the 16 terms for f, f_x, f_y and f_{xy} needed to determine the bicubic interpolation. Then, compute the coefficients a_{ij} by performing the matrix multiplication in Equation (8.2).

4. In the arithmetic circuit for f, apply Equation (8.1) to compute the value of $f(x, y)$. In the arithmetic circuit for ∇f, apply Equation (8.3) to compute the value of $\nabla f(x, y)$. Note that in the interpolation Equations (8.1) and (8.3), we have to use the local coordinates $(x - \widehat{x}, y - \widehat{y}) \in [0, 1]^2$ instead of (x, y).

The two arithmetic circuits can be computed in polynomial time in n, m and in the sizes of S, P, C. Since n and m are upper bounded by the sizes of S and C respectively, they can be constructed in polynomial time in the sizes of S, P, C. Furthermore, note that the two circuits are well-behaved. In fact, they only use a *constant* number of true multiplication gates. To see this, note that true multiplication gates are only used for the matrix multiplication in step 3 and for step 4. In particular, steps 1 and 2 do not need to use any true multiplication gates at all (see, e.g., [DGP09; CDT09]). □

8.4 Correctness

To show the correctness of the construction, we prove the following lemma, which states that 0.01-KKT points of f only lie at solutions for the End-of-Line instance or the Iter instance.

Lemma 8.2. *Let $\varepsilon = 0.01$. We have that (x, y) is an ε-KKT point of f on the domain $[0, N]^2$ only if (x, y) lies in a "solution region," namely:*

- *(x, y) lies in a big square $B(v, v)$, such that $v \in [2^n] \setminus \{1\}$ is a source or sink of the* End-of-Line *instance S, P, or*
- *(x, y) lies in a medium square $M(u, u)$ of some* PLS*-Labyrinth gadget, such that $u \in [2^m]$ is a solution to the* Iter *instance C, i.e., $C(u) > u$ and $C(C(u)) = C(u)$.*

Proof. Even though we have defined ε-KKT points in Section 7.2.1 with respect to the ℓ_2-norm, here it is more convenient to consider the ℓ_∞-norm instead. Note that any ε-KKT point w.r.t. the ℓ_2-norm is also an ε-KKT point w.r.t. the ℓ_∞-norm. Thus, if Lemma 8.2 holds for ε-KKT points w.r.t. the ℓ_∞-norm, then it automatically also holds for ε-KKT points w.r.t. the ℓ_2-norm.

For the domain $[0, N]^2$, it is easy to see that a point $x \in [0, N]^2$ is an ε-KKT point (with respect to the ℓ_∞-norm) if and only if

- for all $i \in \{1, 2\}$ with $x_i \neq 0$: $[\nabla f(x)]_i \leq \varepsilon$
- for all $i \in \{1, 2\}$ with $x_i \neq N$: $-[\nabla f(x)]_i \leq \varepsilon$.

Intuitively, these conditions state that if x is not on the boundary of $[0, N]^2$, then it must hold that $\|\nabla f(x)\|_\infty \leq \varepsilon$. If x is on the boundary of $[0, 1]^2$, then "$-\nabla f(x)$ must point straight outside the domain, up to an error of ε."

In order to prove Lemma 8.2, we will show that any small square that does not lie in a solution region, does not contain any ε-KKT point. The behaviour of the function in a given small square depends on the information we have about the four corners, namely the colours and arrows at the four corners, but also on the position of the square in our instance, since the value defined by a colour depends on the position. For our proof, it is convenient to consider a square with the (colour and arrow) information about its four corners, but without any information about its position. Indeed, if we can show that a square does not contain any ε-KKT point using only this information, then this will always hold, wherever the square is positioned. As a result, we obtain a finite number of squares (with colour and arrow information) that we need to check. Conceptually, this is a straightforward task: for each small square we get a set of

cubic polynomials that could be generated by bicubic interpolation for that square, and we must prove that no polynomial in that set has an ε-KKT point within the square. However, this still leaves us with 101 distinct small squares to verify. To resolve this, we make extensive use of symmetries to drastically reduce the number of cases that need to be verified.

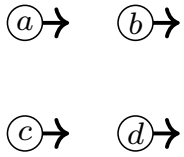
We begin by describing the transformations that we use to group "symmetric" squares together. Throughout, we consider a square where each of its four corners has an associated value in $\mathbb{Z}$ and an arrow pointing in a cardinal direction. The first transformation is *reflection with respect to the axis* $y = 1/2$. Applying this transformation to a square has the following effect: the two corners at the top of the square now find themselves at the bottom of the square (and vice-versa) and the sign of the y-coordinate of each arrow is flipped. Using Equations (8.1) and (8.2) one can check that taking the bicubic interpolation of this reflected square yields the same result as taking the interpolation of the original square and then applying the reflection to the interpolated function (which corresponds to considering $(x, y) \mapsto f(x, 1 - y)$). We can summarize this by saying that bicubic interpolation *commutes* with this transformation.

The second transformation we use is *reflection with respect to the axis* $y = x$, i.e., the diagonal through the square. This corresponds to swapping the corners $(0, 1)$ and $(1, 0)$ of the square, and additionally also swapping the x- and y-coordinate of the arrows at all four corners. Again, using Equations (8.1) and (8.2) one can directly verify that this transformation also commutes with bicubic interpolation (where applying the transformation to the interpolated function f corresponds to considering $(x, y) \mapsto f(y, x)$). These two transformations are already enough to obtain any rotation or reflection of the square. We will only need one more transformation: *negation*. This corresponds to negating the values and arrows at the four corners, where "negating an arrow" just means replacing it by an arrow in the opposite direction. Using Equations (8.1) and (8.2), it is immediately clear that negation commutes with bicubic interpolation.

Since all three transformations commute with bicubic interpolation, this continues to hold for any more involved transformation that is constructed from these basic three. Furthermore, it is easy to see that the basic transformations do not introduce ε-KKT points. Indeed, if a function does not have any ε-stationary points, then applying any reflection or taking the negation cannot change that property. As a result, if two squares are "symmetric" (i.e., one can be obtained from the other by applying a combination of the three basic transformations), then it is enough to verify that just one of these two squares does not contain any ε-KKT points when we take the bicubic interpolation.

Using this notion of symmetry, the squares that need to be verified can be grouped into just four different groups, namely Groups 1 to 4, as shown in Figure 8.16. The remaining squares lie in Group 0, which contains all squares that always lie close to actual solutions and thus do not need to be verified. They are shown in Figure 8.17. We now prove that the squares in Groups 1 to 4 do not contain ε-KKT points.

Group 1. This group contains the squares where all the arrows point in the same direction. These squares are all symmetric to a square of the following form:



Conditions:
$a \geq b + 1$
$c \geq d + 1$

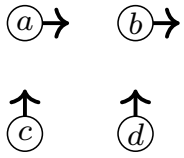
where a, b, c, d are the values at the four corners of the square as shown. We now prove that bicubic interpolation does not introduce any ε-KKT point in such a square.

Recall that $\varepsilon = 0.01$ and that we use $\delta = 1/2$ as the length multiplier for the arrows. Also recall that the arrows point in the *opposite* direction of the gradient. Thus, our goal here will be to prove that the bicubic interpolation f satisfies $\frac{\partial f}{\partial x}(x, y) < -\varepsilon$ for all points $(x, y) \in [0, 1]^2$ in the square, which immediately implies that the square does not contain any ε-KKT point. Using Equations (8.2) and (8.3) and then simplifying, we obtain that for all $x, y \in [0, 1]$:

$$\begin{aligned}\frac{\partial f}{\partial x}(x, y) &= -\frac{1}{2} - 3x(1-x)\Big(1 + 2(1 - y^2(3-2y))(c-d-1) \\ &\qquad\qquad + 2y^2(3-2y)(a-b-1)\Big) \\ &\leq -\frac{1}{2} < -\varepsilon\end{aligned}$$

where we used the fact that $y^2(3-2y) \in [0, 1]$ for all $y \in [0, 1]$, as well as the conditions $c - d - 1 \geq 0$ and $a - b - 1 \geq 0$.

Group 2. This group contains all the squares that are symmetric to a square of the following form:



Conditions:
$a \geq b + 1$
$c \geq a + 1$
$d \geq b + 1$
$c \geq d - 1$

In order to prove that such squares do not contain any ε-KKT points, we distinguish two cases depending on the value of y: when $y \in [0, 2/3]$ we show that $\frac{\partial f}{\partial y}(x, y) < -\varepsilon$,

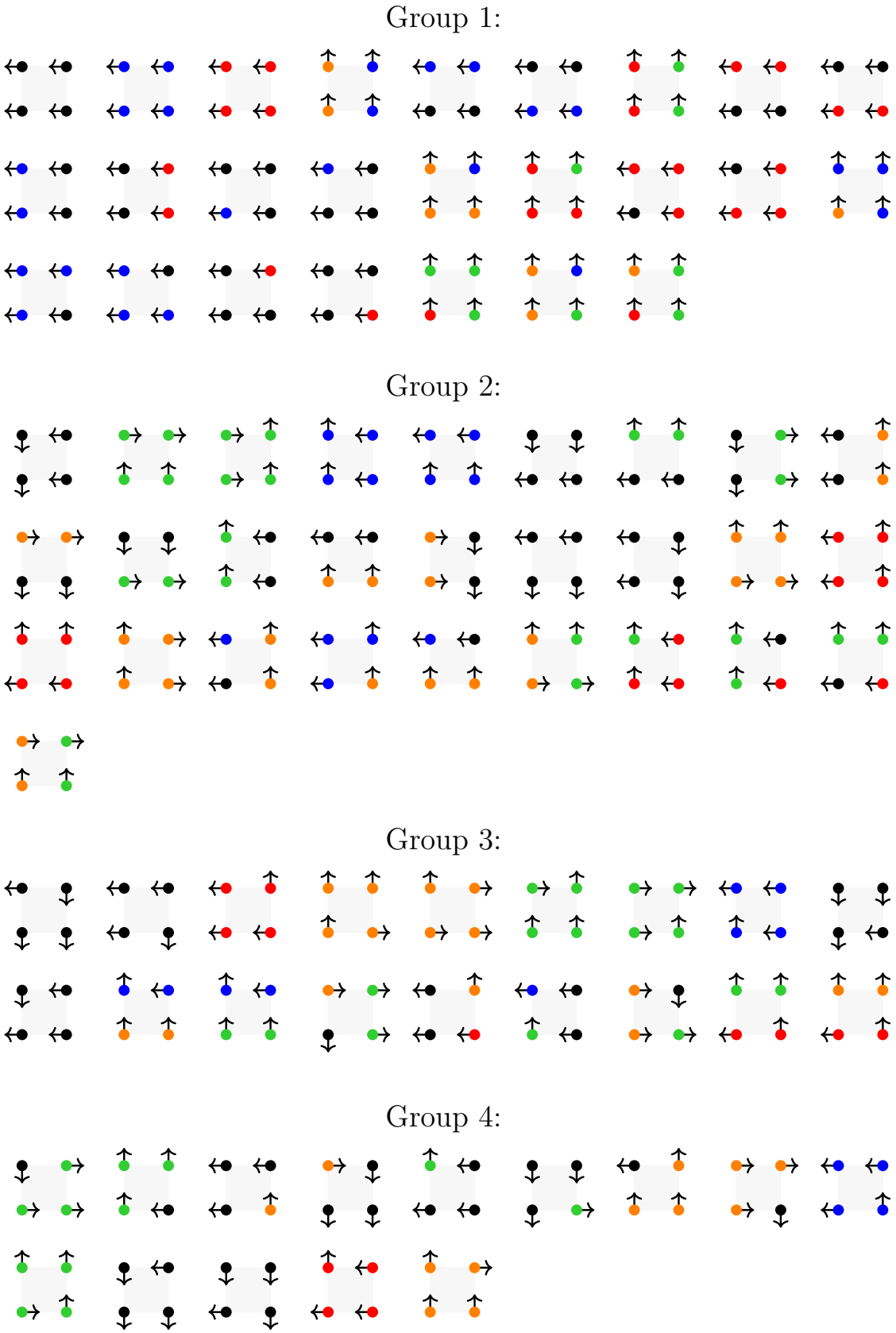


Figure 8.16: Complete list of all squares that need to be verified, partitioned into four groups.

Group 0:

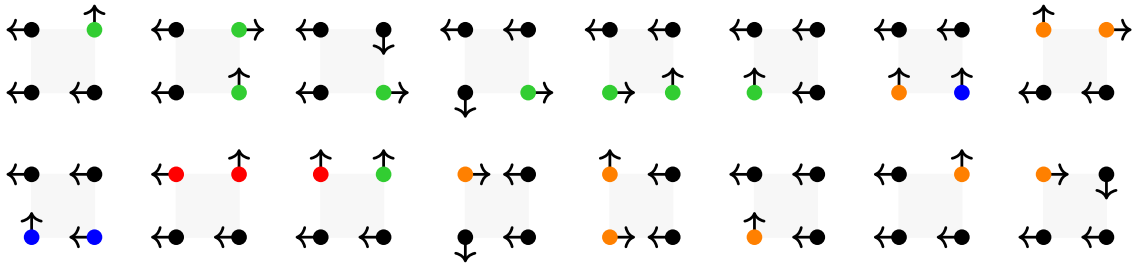


Figure 8.17: List of squares that lie close to actual solutions, and thus do *not* need to be verified.

and when $y \in [2/3, 1]$ we show that $\frac{\partial f}{\partial x}(x, y) < -\varepsilon$. Consider first the case where $y \leq 2/3$. Using Equations (8.2) and (8.3) and simplifying we obtain

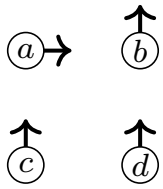
$$\begin{aligned}\frac{\partial f}{\partial y}(x,y) &= -\frac{1}{2}(1-y)\Bigg(1+3y\Bigg(1+2x+4\Big(c-a-\frac{1}{2}\Big)\Big(1-x^2(3-2x)\Big) \\ &\qquad\qquad + 2(2d-2b-2)x^2(3-2x)\Bigg)\Bigg) \\ &\leq -\frac{1}{2}(1-y) \leq -\frac{1}{6} < -\varepsilon\end{aligned}$$

where we used the fact that $x^2(3-2x) \in [0,1]$ for all $x \in [0,1]$, as well as $c-a-1/2 \geq 0$ and $2d-2b-2 \geq 0$ to obtain the first inequality, and then $y \leq 2/3$. Next, consider the case where $y \geq 2/3$. Using Equations (8.2) and (8.3) and simplifying we obtain

$$\begin{aligned}\frac{\partial f}{\partial x}(x,y) &= -\frac{1}{2}y^2(3-2y) - 3x(1-x)\Big((2a-2b-1)y^2(3-2y) \\ &\qquad\qquad + 2(c-d)\left(1-y^2(3-2y)\right)\Big) \\ &\leq -\frac{1}{2}y^2(3-2y) - 3x(1-x)\left(3y^2(3-2y)-2\right) \\ &\leq -\frac{1}{2}y^2(3-2y) \leq -\frac{1}{3} < -\varepsilon\end{aligned}$$

where we used $2a-2b-1 \geq 1$ and $c-d \geq -1$ to obtain the first inequality (as well as the fact that $y^2(3-2y) \in [0,1]$ for all $y \in [0,1]$ as before), and for the second and third inequality the fact that for $y \in [2/3, 1]$ we have $y^2(3-2y) \geq 2/3$.

Group 3. This group contains all the squares that are symmetric to a square of the following form:



Conditions:
$a \geq b+1$
$c \geq a+1$
$d \geq b+1$
$c \geq d-1$

Consider first the case where $(x,y) \in [0,1/2] \times [2/3,1]$. Using Equations (8.2) and (8.3) and simplifying we obtain

$$\begin{aligned}\frac{\partial f}{\partial x}(x,y) &= -\frac{1}{2}(1-x)\Big(y^2(3-2y) + 3x\Big(y^2 - 2y^2(3-2y) + 4(a-b)y^2(3-2y) \\ &\qquad + 4(c-d)\left(1-y^2(3-2y)\right)\Big)\Big) \\ &\leq -\frac{1}{2}(1-x)\Big(y^2(3-2y) + 3x(y^2 + 6y^2(3-2y) - 4)\Big) \\ &\leq -\frac{1}{2}(1-x)y^2(3-2y) \leq -\frac{1}{6} < -\varepsilon\end{aligned}$$

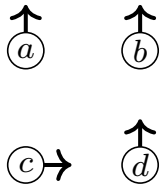
where we used $a-b \geq 1$, $c-d \geq -1$ and $y^2(3-2y) \in [0,1]$ for the first inequality, and then the fact that $y^2(3-2y) \geq 2/3$ for $y \in [2/3,1]$, as well as $x \leq 1/2$, for the second and third inequality.

Next, consider the case where $(x,y) \notin [0,1/2] \times [2/3,1]$, i.e., $x > 1/2$ or $y < 2/3$. Using Equations (8.2) and (8.3) and simplifying we obtain

$$\begin{aligned}\frac{\partial f}{\partial y}(x,y) &= -\frac{1}{2} + \frac{y}{2}\Big(y\big(1-x^2(3-2x)\big) - (1-y)\Big(-4 + 3x(2-x) - 5x^2(3-2x) \\ &\qquad + 12(d-b-1)x^2(3-2x) + 12(c-a-1)\left(1-x^2(3-2x)\right)\Big)\Big) \\ &\leq -\frac{1}{2} + \frac{y}{2}\Big(y\left(1-x^2(3-2x)\right) - (1-y)\big(8 + 3x(2-x) - 5x^2(3-2x)\big)\Big) \\ &\leq -\frac{1}{2} + \frac{y^2}{2}\left(1-x^2(3-2x)\right) \\ &\leq -\frac{1}{2} + \max\left\{\frac{2}{9}, \frac{3}{8}\right\} < -\varepsilon\end{aligned}$$

where we used $d-b-1 \geq 0$, $c-a-1 \geq 0$ and $x^2(3-2x) \in [0,1]$ for the first and second inequalities. For the third inequality we used the fact that $x > 1/2$ or $y < 2/3$, where we note that $1-x^2(3-2x) \in [0,1]$ when $x \in [0,1]$, and $1-x^2(3-2x) \leq 3/4$ when $x \in [1/2,1]$.

Group 4. This group contains all the squares that are symmetric to a square of the following form:



Conditions:
$c \geq a + 1$
$d \geq b + 1$
$c \geq d + 1$
$a \geq b - 1$

Consider first the case where $(x, y) \in [0, 1/2] \times [1, 1/3]$. Using Equations (8.2) and (8.3) and simplifying we obtain

$$\begin{aligned}\frac{\partial f}{\partial x}(x,y) &= -\frac{1}{2}(1-x)\Big(1 - y^2(3-2y) + 3x\Big(-1 + y(2-y) + 4(a-b)y^2(3-2y) \\ &\qquad + 4(c-d)\left(1 - y^2(3+2y)\right)\Big)\Big) \\ &\leq -\frac{1}{2}(1-x)\Big(1 - y^2(3-2y) + 3x\big(3 + y(2-y) - 8y^2(3-2y)\big)\Big) \\ &\leq -\frac{1}{2}(1-x)\left(1 - y^2(3-2y)\right) \leq -\frac{1}{6} < -\varepsilon\end{aligned}$$

where we used $a - b \geq -1$, $c - d \geq 1$ and $y^2(3-2y) \in [0, 1]$ for the first inequality, and $y^2(3-2y) \leq 1/3$ for $y \in [0, 1/3]$ for the second inequality. For the third inequality, we used the fact that $x \leq 1/2$ and $1 - y^2(3-2y) \geq 2/3$ for $y \in [0, 1/3]$.

Next, consider the case where $(x, y) \notin [0, 1/2] \times [1, 1/3]$, i.e., $x > 1/2$ or $y > 1/3$. Using Equations (8.2) and (8.3) and simplifying we obtain

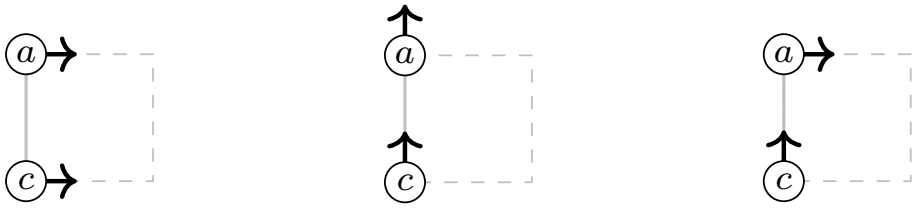
$$\begin{aligned}\frac{\partial f}{\partial y}(x,y) &= -\frac{1}{2} + \frac{1}{2}(1-y)\big(1 - x^2(3-2x)\big) - \frac{3}{2}y(1-y)\Big(3 - x(2-x) \\ &\qquad + 4(d-b-1)x^2(3-2x) + 4(c-a-1)\left(1 - x^2(3-2x)\right)\Big) \\ &\leq -\frac{1}{2} + \frac{1}{2}(1-y)\left(1 - x^2(3-2x)\right) - \frac{3}{2}y(1-y)\big(3 - x(2-x)\big) \\ &\leq -\frac{1}{2} + \frac{1}{2}(1-y)\left(1 - x^2(3-2x)\right) \\ &\leq -\frac{1}{2} + \max\left\{\frac{3}{8}, \frac{1}{3}\right\} < -\varepsilon\end{aligned}$$

where we used $d - b - 1 \geq 0$, $c - a - 1 \geq 0$ and $x^2(3-2x) \in [0, 1]$ for the first inequality, and the fact that $3 - x(2-x) \geq 0$ for the second inequality. For the third inequality we used the fact that $x > 1/2$ or $y > 1/3$, where we again note that $1 - x^2(3-2x) \in [0, 1]$ when $x \in [0, 1]$, and $1 - x^2(3-2x) \leq 3/4$ when $x \in [1/2, 1]$.

Boundary. Up to this point we have shown that there are no unintended ε-KKT points in the interior of the domain $[0, N]^2$ of our instance. It remains to show that no ε-KKT points appear on the boundary of the domain. Intuitively, this corresponds to showing that $-\nabla f$ never points "straight outside the domain" when we are on the boundary.

First of all, it is easy to see that there is no ε-KKT point at any of the four corners of the domain $[0, N]^2$. This follows from the fact that at any such corner the arrow never points outside the domain (see Figure 8.10). Since these corners of the domain are also corners of their respective squares, $-\nabla f$ (where f is obtained by bicubic interpolation in every square) will automatically be equal to the arrow at that corner (scaled by $\delta = 1/2$), which ensures that it is not an ε-KKT point.

It remains to consider points that lie on the boundary of the domain, except at the corners. Using the first two transformations introduced earlier it is easy to check that any square that lies on the boundary (see Figure 8.10) is symmetric to one of the three following cases:



Case (i) Case (ii) Case (iii)

where for cases (ii) and (iii) we also have that $c \geq a + 1$. In each of these three illustrations the square touches the boundary of the domain with its left edge (which is not dashed). In order to show that there are no ε-KKT points on that left edge, it suffices to show that for any (x, y) on that edge (i.e., $x = 0$, $y \in [0, 1]$) we have $\frac{\partial f}{\partial y}(x, y) \notin [-\varepsilon, \varepsilon]$ or $\frac{\partial f}{\partial x}(x, y) < -\varepsilon$.

- For case (i), we have
$$\frac{\partial f}{\partial x}(0, y) = -\frac{1}{2}y^2(3 - 2y) - \frac{1}{2}\left(1 - y^2(3 - 2y)\right) = -\frac{1}{2} < -\varepsilon.$$
- For case (ii), we have
$$\frac{\partial f}{\partial y}(0, y) = -\frac{1}{2} - 3(2c - 2a - 1)(1 - y)y \leq -\frac{1}{2} < -\varepsilon$$
where we used $2c - 2a - 1 \geq 0$.
- For case (iii), if $y \geq 1/2$ then we have
$$\frac{\partial f}{\partial x}(0, y) = -\frac{1}{2}y^2(3 - 2y) \leq -\frac{1}{4} < -\varepsilon$$
since $y^2(3 - 2y) \geq 1/2$ when $y \in [1/2, 1]$. On the other hand, if $y \leq 1/2$ then we have
$$\frac{\partial f}{\partial y}(0, y) = -\frac{1}{2}(1 - y)\left(1 + 3(4c - 4a - 1)y\right) \leq -\frac{1}{2}(1 - y) \leq -\frac{1}{4} < -\varepsilon$$
where we used $4c - 4a - 1 \geq 0$.

It follows that there are no ε-KKT points on the boundary of the domain. This completes the proof of Lemma 8.2. □

8.4.1 Re-scaling

The last step of the reduction is to re-scale the function f so that it is defined on $[0,1]^2$ instead of $[0,N]^2$. Thus, the final function, which we denote $\widehat{f}$ here, is defined by

$$\widehat{f}(x,y) = (1/N) \cdot f(N \cdot x, N \cdot y).$$

The properties of f proved in Lemma 8.1 naturally also hold for $\widehat{f}$, in the following sense. Clearly, $\widehat{f}$ is also continuously differentiable. Furthermore, it holds that $\nabla\widehat{f}(x,y) = \nabla f(N \cdot x, N \cdot y)$. Thus, we can easily construct well-behaved arithmetic circuits for $\widehat{f}$ and $\nabla\widehat{f}$ in polynomial time given well-behaved circuits for f and ∇f, which, in turn, can be efficiently constructed according to Lemma 8.1. Furthermore, since ∇f is L-Lipschitz-continuous, it is easy to see that $\nabla\widehat{f}$ is $\widehat{L}$-Lipschitz-continuous with $\widehat{L} = N \cdot L = 2^{18}N^2 = 2^{2n+2m+26}$. Finally, note that since f is L-Lipschitz-continuous, $\widehat{f}$ is too, and in particular it is also $\widehat{L}$-Lipschitz-continuous.

All these properties imply that the instance of KKT we construct does not admit any violation solutions. In other words, it satisfies all the expected promises. Finally, note that any ε-KKT point of $\widehat{f}$ on $[0,1]^2$ immediately yields an ε-KKT point of f on $[0,N]^2$. Thus, the correctness of the reduction follows from Lemma 8.2.

Note that we can re-scale the instance depending on the parameter regime we are interested in. The instance $(\varepsilon, \widehat{f}, \nabla\widehat{f}, \widehat{L})$ we have constructed is clearly equivalent to the instance $(\alpha\varepsilon, \alpha\widehat{f}, \nabla(\alpha\widehat{f}), \alpha\widehat{L})$ for any $\alpha > 0$. For example, by letting $\alpha = 1/\widehat{L}$, we obtain hard instances with Lipschitz-constant 1, and with inversely exponential precision parameter.

Part III

PPA-k

Chapter 9

The Classes PPA-k: Existence from Arguments Modulo k

The TFNP subclasses PPA-p were defined by Papadimitriou almost 30 years ago in his seminal paper [Pap94]. Recall that the existence of a solution to a PPA problem is guaranteed by a parity argument, i.e., an argument modulo 2. The classes PPA-p are a generalisation of this. For every prime p, the existence of a solution to a PPA-p problem is guaranteed by an argument modulo p. In particular, PPA-2 = PPA. Surprisingly, these classes have received very little attention. As far as we know, they have only been studied in the following:

- Papadimitriou [Pap94] defined the classes PPA-p and proved that a problem called Chevalley-mod-p lies in PPA-p and a problem called Cubic-Subgraph lies in PPA-3.
- In an online thread on Stack Exchange [Jeř17], Jeřábek provided two other equivalent ways to define PPA-3. The problems and proofs can be generalised to any prime p.
- In his thesis [Joh11], Johnson defined the classes PMOD^k for any $k \geq 2$, which were intended to capture the complexity of counting arguments modulo k. He proved various oracle separation results involving his classes and other TFNP classes. While the PPA-p classes are not mentioned by Johnson, using Jeřábek's results [Jeř17] it is easy to show that PMOD^p = PPA-p for any prime p. In Section 9.4, we characterise PMOD^k in terms of the classes PPA-p when k is not prime. In particular, we show that PMOD^k only partially captures existence arguments modulo k.

In this chapter, we use the natural generalisation of Papadimitriou's definition of the classes PPA-p to define PPA-k for any $k \geq 2$. We then provide a characterisation of PPA-k in terms of the classes PPA-p. In particular, we show that PPA-k is completely

determined by the set of prime factors of k. In order to gain a better understanding of the inner structure of the class PPA-k, we also define new subclasses that we denote PPA-$k[\#\ell]$ and investigate how they relate to the other classes. We show that PPA-$k[\#\ell]$ is completely determined by the set of prime factors of $k/\gcd(k,\ell)$.

Furthermore, we provide various equivalent complete problems that can be used to define PPA-k and PPA-$k[\#\ell]$ (Section 9.2). While these problems are not "natural," we believe that they provide additional tools that can be very useful when proving that natural problems are complete for these classes. In Section 9.5, we provide an additional tool for showing that problems lie in these classes: we prove that PPA-p^r (p prime, $r \geq 1$) and PPA-$k[\#\ell]$ ($k \geq 2$) are closed under Turing reductions. On the other hand, we provide evidence that PPA-k might not be closed under Turing reductions when k is not a prime power.

Finally, in Section 9.4 we investigate the classes PMOD^k defined by Johnson [Joh11] and provide a full characterisation in terms of the classes PPA-k. In particular, we show that PMOD^k = PPA-k if k is a prime power. However, when k is not a prime power, we provide evidence that PMOD^k does not capture the full strength of existence arguments modulo k, unlike PPA-k. This characterisation of PMOD^k in terms of PPA-k leads to some oracle separation results involving PPA-k and other TFNP classes (using Johnson's oracle separation results).

We note that a significant fraction of our results were also obtained by Göös, Kamath, Sotiraki and Zampetakis in concurrent and independent work [GKSZ20]. In their work, they have also provided the first "natural" complete problem for the classes PPA-p (a variant of Chevalley-mod-p), namely the first complete problem that does not involve circuits or other computational devices in its description.

9.1 Definition of the Classes

9.1.1 PPA-k : Polynomial Argument modulo k

For any prime p, Papadimitriou [Pap94] defined the class PPA-p as the set of all TFNP problems that many-one reduce to the following problem, that we call Bipartite-mod-p: We are given an undirected bipartite graph (implicitly represented by a circuit) and a vertex with degree $\neq 0 \mod p$ (which we call the *trivial solution*). The goal is to find another such vertex. This problem lies in TFNP: if all other vertices had degree $= 0 \mod p$, then the sum of the degrees of all vertices on each side would have a different value modulo p, which is impossible.

The problem remains well-defined and total if p is not a prime, and so we will instead define it for any $k \geq 2$. Let us now provide a formal definition of the problem. A vertex of the bipartite graph is represented as a bit-string in $\{0,1\} \times \{0,1\}^n$, where

the first bit indicates whether the vertex lies on the "left" or "right" side of the bipartite graph. The graph will be represented by a Boolean circuit that outputs a set of potential neighbours, just as we did for LEAF. Instead of at most two neighbours, here we allow at most k neighbours (see Remark 9.1 for why this is enough). Note that we can syntactically enforce that the graph is bipartite, i.e., that a vertex $0x$ can only have neighbours of the type $1y$ and vice versa. Formally, the problem is defined as follows for any $k \geq 2$, where we use the Set-notation introduced in Section 2.2.1.

Definition 9.1. **BIPARTITE-MOD-k:**

Input: Boolean circuit $C : \{0,1\} \times \{0,1\}^n \to \mathsf{Set}_{\leq k}(\{0,1\} \times \{0,1\}^n)$ representing a bipartite graph on the vertex set $(\{0\} \times \{0,1\}^n, \{1\} \times \{0,1\}^n)$ with $|C(00^n)| \in \{1, \ldots, k-1\}$.
Output: Either

- $x \neq 00^n$ such that $|C(x)| \notin \{0, k\}$, or
- x, y such that $y \in C(x)$ but $x \notin C(y)$.

Here the trivial solution is the vertex 00^n. The first type of solution corresponds to a vertex with degree $\neq 0 \mod k$. The second type of solution corresponds to an edge that is not well-defined. We can always ensure that all edges are well-defined by doing some pre-processing. Indeed, in polynomial time we can construct a circuit C' such that all solutions are of the first type and yield a solution for C. On input $0x$ the circuit C' first computes $C(0x) = \{1y_1, \ldots, 1y_m\}$ and then for each i removes $1y_i$ from this set, if $0x \notin C(1y_i)$.

Remark 9.1. Note that in this problem statement we require that all degrees lie in $\{0, 1, \ldots, k\}$. This is easily seen to be equivalent to the more general formulation where vertices can have more than k neighbours. Indeed, any vertex that has more than k edges can be split into multiple copies such that all the copies have 0 or k edges, except for one copy which is allowed to have any number of edges in $\{0, 1, \ldots, k\}$. A solution of the original instance is then easily recovered from a solution of this modified instance. Note that since the set of neighbours is given as the output of a circuit, it will have length bounded by some polynomial in the input size and so this argument can indeed be applied.

Definition 9.2 (PPA-k [Pap94])**.** For any $k \geq 2$, the class PPA-k is defined as the set of all TFNP problems that many-one reduce to BIPARTITE-MOD-k.

As a warm-up let us show the following:

Proposition 9.1 ([Pap94])**.** PPA-2 = PPA

Proof. Recall that PPA can be defined using the canonical complete problem Leaf [Pap94; BCE+98]: given an undirected graph where every vertex has degree at most 2, and a leaf (i.e., degree = 1), find another leaf. This immediately yields PPA-2 $\subseteq$ PPA, since Bipartite-mod-2 is just a special case of Leaf where the graph is bipartite.

Given an instance of Leaf with graph $G = (\{0,1\}^n, E)$ we construct an instance of Bipartite-mod-2 on the vertex set $\{0,1\} \times \{0,1\}^{2n}$ as follows. For any $u \in \{0,1\}^n$ we have a vertex $x_u := 0u0^n$ on the left side of the bipartite graph. For any edge $\{u,v\} \in E$ (u, v ordered lexicographically) we have a vertex $y_{uv} := 1uv$ on the right side of the bipartite graph and we create the edges $\{x_u, y_{uv}\}$ and $\{x_v, y_{uv}\}$. All other vertices in $\{0,1\} \times \{0,1\}^{2n}$ are isolated. In polynomial time we can construct a circuit that computes the neighbours of any vertex. Furthermore, $w \in \{0,1\}^n$ is a leaf, if and only if x_w has degree 1. Finally, all vertices on the right-hand side have degree 0 or 2. □

9.1.2 PPA-$k[\#\ell]$: Fixing the degree of the trivial solution

In the definition of the PPA-k-complete problem Bipartite-mod-k (Definition 9.1) the degree of the trivial solution 00^n can be any number in $\{1, \ldots, k-1\}$. In this section we define more refined classes where the degree of the trivial solution is fixed. In Section 9.3, these classes will be very useful to describe how the PPA-k classes relate to each other. These definitions are inspired by the corresponding "counting principles" studied in Beame et al. [BIK+96] that were also defined in a refined form in order to describe how they relate to each other. We believe that these refined classes will also be useful to capture the complexity of natural problems. Note that for $k = 2$, the degree of the trivial solution will always be 1 and thus the question does not even appear in the study of PPA.

Definition 9.3. Let $k \geq 2$ and $1 \leq \ell \leq k-1$. The problem Bipartite-mod-$k[\#\ell]$ is defined as Bipartite-mod-k (Definition 9.1) but with the additional condition $|C(00^n)| = \ell$.

Note that this problem remains in TFNP, since the condition can be checked efficiently.

Definition 9.4 (PPA-$k[\#\ell]$)**.** Let $k \geq 2$ and $1 \leq \ell \leq k-1$. The class PPA-$k[\#\ell]$ is defined as the set of all TFNP problems that many-one reduce to Bipartite-mod-$k[\#\ell]$.

If k is some prime p, then these classes are not interesting. Indeed, it holds that PPA-$p[\#\ell]$ = PPA-p for all $1 \leq \ell \leq p-1$. This can be shown using the following technique: take multiple copies of the instance and "glue" the trivial solutions together. If p is prime, then any other degree of the glued trivial solution can be obtained (by taking the right number of copies). In fact this technique yields the stronger result:

Lemma 9.1. *If* $\gcd(k, \ell_1)$ *divides* ℓ_2*, then* PPA-$k[\#\ell_1] \subseteq$ PPA-$k[\#\ell_2]$.

Proof. Since $\gcd(k, \ell_1)$ divides ℓ_2, there exists $m < k$ such that $m \times \ell_1 = \ell_2 \mod k$. Given an instance of Bipartite-mod-$k[\#\ell_1]$, take the union of m copies of the instance, i.e., $m2^n$ vertices on each side (and any additional isolated vertices needed to reach a power of 2). Then, merge the m different copies of the trivial solution into one (by redirecting edges to a single one). This vertex will have degree $m\ell_1 = \ell_2 \mod k$. Finally, apply the usual trick to ensure all degrees are in $\{0, 1, \ldots, k\}$ (Remark 9.1). □

In particular, we also get the nice result PPA-$k[\#\ell]$ = PPA-$k[\#\gcd(k, \ell)]$. Applying the result to the case $k = 6$, we get that PPA-6[#1] = PPA-6[#5], PPA-6[#2] = PPA-6[#4], as well as PPA-6[#1] ⊆ PPA-6[#2] and PPA-6[#1] ⊆ PPA-6[#2]. Thus, we have three "equivalence classes" $\{1, 5\}$, $\{2, 4\}$ and $\{3\}$ and the relationships $\{1, 5\} \leq \{2, 4\}$ and $\{1, 5\} \leq \{3\}$. In Section 9.3, we will show that $\{2, 4\}$ corresponds to PPA-3, $\{3\}$ to PPA-2 and $\{1, 5\}$ to PPA-2 ∩ PPA-3.

Now let us introduce some notation that will allow us to precisely describe the relationship between PPA-k and the PPA-$k[\#\ell]$.

Definition 9.5 (& operation [BJ12])**.** Let R_0 and R_1 be two TFNP problems. Then the problem $R_0 \,\&\, R_1$ is defined as: given an instance I_0 of R_0, an instance I_1 of R_1 and a bit $b \in \{0, 1\}$, find a solution to I_b.

This operation is commutative and associative (up to many-one equivalence). Indeed, $R_0 \,\&\, R_1$ is many-one equivalent to $R_1 \,\&\, R_0$, and $(R_0 \,\&\, R_1) \,\&\, R_2$ is many-one equivalent to $R_0 \,\&(R_1 \,\&\, R_2)$. Since the & operation is associative, the problem $\&_{\ell=1}^{k} R_\ell$ is well-defined up to many-one equivalence. It is also equivalent to the following problem: given instances $I_1, \ldots, I_k$ of $R_1, \ldots, R_k$ and an integer $j \in \{1, \ldots, k\}$, find a solution to I_j.

We extend the & operation to TFNP subclasses in the natural way. Let C_0 and C_1 be TFNP subclasses with complete problems R_0 and R_1 respectively. Then $C_0 \,\&\, C_1$ is the class of all TFNP problems that many-one reduce to $R_0 \,\&\, R_1$. Note that the choice of complete problems does not matter. Intuitively, this class contains all problems that can be solved in polynomial time by a Turing machine with a single oracle query to either C_0 or C_1.[1]

Using this definition we obtain:

Lemma 9.2. *For all* $k \geq 2$ *we have* PPA-$k = \overset{k-1}{\underset{\ell=1}{\&}}$ PPA-$k[\#\ell]$.

[1]Note that $C_0 \,\&\, C_1$ is not the same operation as $C_0 \cup C_1$. Indeed, it holds that $C_0 \cup C_1 \subseteq C_0 \,\&\, C_1$, but the two are not equal, unless $C_0 \subseteq C_1$ or $C_1 \subseteq C_0$.

Proof. One containment immediately follows from the fact that PPA-$k[\#\ell] \subseteq$ PPA-k for all $\ell \in \{1, \ldots, k-1\}$. For the other containment, note that for any instance I of Bipartite-mod-k we can easily compute $\ell \in \{1, \ldots, k-1\}$ such that I is also an instance of Bipartite-mod-$k[\#\ell]$. □

Together with Lemma 9.1, Lemma 9.2 yields, e.g., PPA-6 = PPA-6[#2] & PPA-6[#3].

9.2 Equivalent Definitions

In this section we show that PPA-k can be defined by using other problems instead of Bipartite-mod-k. The totality of these problems is again based on arguments modulo k. By showing that these problems are indeed PPA-k-complete, we provide additional support for the claim that PPA-k captures the complexity of "polynomial arguments modulo k." While these problems are not "natural" and thus not interesting in their own right, they provide equivalent ways of defining PPA-k, which can be very useful when working with these classes. In particular, we make extensive use of these equivalences in this work.

The TFNP problems we consider are the following:

- Imbalance-mod-k : given a directed graph and a vertex that is *unbalanced-mod-k*, i.e., out-degree $-$ in-degree $\neq 0 \mod k$, find another such vertex.
- Hypergraph-mod-k : given a hypergraph and a vertex that has degree $\neq 0 \mod k$, find another such vertex or a hyperedge that has size $\neq k$.
- Partition-mod-k : given a set of size $\neq 0 \mod k$ and a partition into subsets, find a subset that has size $\neq k$.

As usual, the size of the graph (respectively hypergraph, set) can be exponential in the input size, and the edges (resp. hyperedges, subsets) can be computed efficiently locally. We also define the corresponding problems Imbalance-mod-$k[\#\ell]$, Hypergraph-mod-$k[\#\ell]$ and Partition-mod-$k[\#\ell]$ analogously. The formal definitions of all these problems are provided in Section 9.2.1.

Theorem 9.1. *Let $k \geq 2$ and $1 \leq \ell \leq k-1$.*

- Imbalance-mod-$k[\#\ell]$, Hypergraph-mod-$k[\#\ell]$, Partition-mod-$k[\#\ell]$ *are* PPA-$k[\#\ell]$*-complete,*
- Imbalance-mod-k, Hypergraph-mod-k, Partition-mod-k *are* PPA-k*-complete.*

In his online post [Jeř17], Jeřábek proves that Bipartite-mod-3, Imbalance-mod-3 and Partition-mod-3 are equivalent and (correctly) claims that the proof generalises to any other prime. Thus, our contribution is the definition of the problems for any $k \geq 2$ (and the ℓ-parameter versions) and the generalisation of the result to any $k \geq 2$ (not only primes) and to the ℓ-parameter versions of the problems, as well as to the new problem Hypergraph-mod-k. The proof of Theorem 9.1 can be found in Section 9.2.2.

The problem Imbalance-mod-k is a generalisation of the PPAD-complete problem Imbalance (see Chapter 3): given a directed graph and a vertex that is unbalanced (i.e., out-degree − in-degree $\neq 0$), find another unbalanced vertex. It is easy to see that End-of-Line trivially reduces to Imbalance-mod-k, and thus Theorem 9.1 also yields:[2]

Corollary 9.1. *For all $k \geq 2$, we have* PPAD $\subseteq$ PPA-k.

Furthermore, if we use the convention that $a = b \mod 0$ if and only if $a = b$, then Imbalance-mod-0 actually corresponds to Imbalance. Thus, in a certain sense we could define PPA-0 = PPAD. On the other hand, Imbalance-mod-1 is a trivial problem.

9.2.1 Formal definitions

Imbalance. A directed graph on the vertex set $\{0,1\}^n$ is represented by Boolean circuits $S, P : \{0,1\}^n \to \mathsf{Set}_{\leq k}(\{0,1\}^n)$ that output the successor and predecessor set of a given vertex, respectively. As before, it is enough to consider the case where the in- and out-degree of any vertex is at most k, since the general case reduces to this (analogously to Remark 9.1). We syntactically enforce that $x \notin S(x) \cup P(x)$ and we interpret $S(x)$ (respectively, $P(x)$) as the set of potential successors (respectively, predecessors) of x. There is a directed edge from x to y if $y \in S(x)$ and $x \in P(y)$. The following problem was defined by Jeřábek [Jeř17], but only for prime k and without the ℓ-parameter version. Here we define it for any $k \geq 2$.

Definition 9.6. **Imbalance-mod-k:**

Input: Boolean circuits $S, P : \{0,1\}^n \to \mathsf{Set}_{\leq k}(\{0,1\}^n)$ representing a directed graph on the vertex set $\{0,1\}^n$ with $|S(0^n)| - |P(0^n)| \notin \{-k, 0, k\}$.
Output: Either

- $x \neq 0^n$ such that $|S(x)| - |P(x)| \notin \{-k, 0, k\}$, or
- x, y such that $y \in S(x)$ but $x \notin P(y)$, or $y \in P(x)$ but $x \notin S(y)$.

[2] This observation was also made by Jeřábek for the classes PPA-p (p prime).

For $1 \leq \ell \leq k-1$, Imbalance-mod-$k[\#\ell]$ is defined with the additional condition $|S(0^n)| - |P(0^n)| = \ell$.

We obtain a seemingly more general version of this problem by allowing the edges to have integer weights. In that case the imbalance of a vertex is measured as the difference of the weights of all incoming edges and the weights of all outgoing edges. It is easy to see that this problem is in fact equivalent to Imbalance-mod-k. First, all the weights can be assumed to be in $\{0, 1, \ldots, k-1\}$, since we can reduce them modulo k. Next, we can split an edge with weight ℓ into ℓ copies of the edge. Finally, to ensure that we don't have multi-edges, we add a new vertex in the middle of every edge. Note that the new vertex will be balanced by construction, and will thus not introduce any new solutions.

Hypergraph. A hypergraph on the vertex set $\{0,1\}^n$ is represented as follows. For every vertex $x \in \{0,1\}^n$, a circuit $C : \{0,1\}^n \to \mathsf{Set}_{\leq k}(\mathsf{Set}_{\leq k}(\{0,1\}^n))$ outputs the set $C(x)$ of all hyperedges containing x, where each hyperedge is a set of vertices in $\{0,1\}^n$. As usual, we only need to consider the case where every vertex is contained in at most k hyperedges and every hyperedge has size at most k. A hyperedge $\{x_1, \ldots, x_m\}$ exists in the hypergraph, if all the vertices involved indeed agree that it is present, i.e., if $\{x_1, \ldots, x_m\} \in C(x_i)$ for all $i \in \{1, \ldots, m\}$. For any $k \geq 2$, we define:

Definition 9.7. **Hypergraph-mod-k:**

Input: Boolean circuit $C : \{0,1\}^n \to \mathsf{Set}_{\leq k}(\mathsf{Set}_{\leq k}(\{0,1\}^n))$ representing a hypergraph on the vertex set $\{0,1\}^n$ with $|C(0^n)| \notin \{0, k\}$.
Output: Either

- $x \neq 0^n$ such that $|C(x)| \notin \{0, k\}$, or
- x such that $C(x)$ contains a hyperedge of size $\neq k$, or
- x, y such that $C(x)$ and $C(y)$ are not consistent with one another.

For $1 \leq \ell \leq k-1$, Hypergraph-mod-$k[\#\ell]$ is defined with the additional condition $|C(0^n)| = \ell$.

Note that for $k = 2$ this problem essentially corresponds to the PPA-complete problem Leaf and its (equivalent) generalisation Odd [Pap94; BCE+98]: given an undirected graph and a vertex with odd degree, find another one.

Partition. A partition of $\{0,1\}^n$ is represented by a Boolean circuit $C : \{0,1\}^n \to \{0,1\}^n$ as follows: $x \in \{0,1\}^n$ lies in the subset given by the orbit of x with respect to C, i.e., $\{C^i(x) : i \geq 0\}$, where $C^i(x) = C(C(\ldots C(x)) \ldots)$ (i times). The problem we define below is based on the simple observation that a base set of size $\neq 0 \mod k$

cannot be partitioned into sets of size k. The base set consists of all elements in $\{0,1\}^n$ except for m elements that have been removed, for some $m < 2^n$ such that $2^n - m \neq 0 \mod k$. Here it is convenient to identify $\{0,1\}^n$ with $\{0,1,\ldots,2^n-1\}$ in the natural way. Thus, we can think of the base set as simply being $\{m, m+1,\ldots,2^n-1\}$. For any $k \geq 2$, we define:

Definition 9.8. **PARTITION-MOD-k:**

Input: $m < 2^n$ with $2^n - m \neq 0 \mod k$ and a Boolean circuit $C : \{0,1\}^n \to \{0,1\}^n$, such that $C(x) = x$ for all $x < m$.
Output: Either

- $x \geq m$ and $d \in \mathbb{N}$ such that $C^d(x) = x$ and $d|k$, $d \neq k$, or
- $x \in \{0,1\}^n$ such that $C^k(x) \neq x$

where $d|k$ means that d divides k.

For $1 \leq \ell \leq k-1$, PARTITION-MOD-$k[\#\ell]$ is defined with the additional condition $2^n - m = \ell \mod k$.

The condition "$C(x) = x$ for all $x < m$" corresponds to excluding elements that do not lie in the base set and it can be enforced syntactically. The first solution type corresponds to finding a set in the partition such that its size divides k (but is $\neq k$), while the second solution type corresponds to finding a set such that its size does not divide k. Note that a solution is guaranteed to exist since $2^n - m \neq 0 \mod k$.

The definition of this problem is inspired by the MOD^k problems defined by Buss and Johnson [BJ12] (for prime $k > 2$) and by Johnson [Joh11] (for any $k \geq 2$). In Section 9.4 we argue that, unlike the problem defined above, the MOD^k problems only partially capture the complexity of arguments modulo k (when k is not a prime power). The problem was also defined by Jeřábek [Jeř17], but only for k prime and without the ℓ-parameter version. Finally, note that PARTITION-MOD-2 essentially corresponds to the PPA-complete problem LONELY (Section 2.3.2).

The definition of this problem can be modified in various ways without changing its complexity. For instance, the first solution type can be changed to simply ask for $x \geq m$ such that $C^d(x) = x$ for some $d < k$. We have defined the problem in a slightly more complicated way to make the connection with the MOD^k problems more immediate (see Section 9.4). Yet another equivalent way of defining the problem would be to consider a Boolean circuit $C : \{0,1\}^n \to \mathsf{Set}_{\leq k}(\{0,1\}^n)$ where $C(x) \subseteq \{0,1\}^n$ is interpreted as the set containing x in the partition. A solution would then be any $x \geq m$ with $|C(x)| < k$ or any x, y witnessing an inconsistency in the partition given by C.

9.2.2 Proof of Theorem 9.1

We omit some details that are easy to fill in. For example, when given an instance of Imbalance-mod-$k[\#\ell]$, we assume that all the edges are well-defined, i.e., solutions of the second type never occur. Indeed, given a generic instance of the problem, it can be reduced to an instance where this holds by modifying the circuits so that they check and correct the successor/predecessor list before outputting it. Note that in the new instance only solutions of the first type can occur, but they can yield a solution of the second type of the original problem. The same observation also holds for Bipartite-mod-$k[\#\ell]$ (edges well-defined), Hypergraph-mod-$k[\#\ell]$ (hyperedges well-defined) and Partition-mod-$k[\#\ell]$ (size of any subset divides k).

Bipartite-mod-$k[\#\ell] \leq$ Hypergraph-mod-$k[\#\ell]$: We construct a hypergraph on the vertex set $\{0,1\}^n$. We identify every vertex u of the hypergraph with the vertex $0u$ on the left-hand side of the bipartite graph. The hyperedges are given by the vertices on the right-hand side of the bipartite graph. More precisely, if for every right-hand side vertex $1v$ we let $N(1v)$ be the set of neighbours (on the left-hand side), then the set of hyperedges is exactly $\{N(1v) : v \in \{0,1\}^n\}$. Note that given $u \in \{0,1\}^n$, we can find all the hyperedges containing u in polynomial time. Furthermore, since the vertex 00^n has degree ℓ in the bipartite graph, the corresponding vertex 0^n in the hypergraph will also have degree ℓ. It is easy to check that any solution of the Hypergraph-mod-$k[\#\ell]$ instance (in particular also any hyperedge that does not have size k) yields a solution to the Bipartite-mod-$k[\#\ell]$ instance.

Hypergraph-mod-$k[\#\ell] \leq$ Imbalance-mod-$k[\#\ell]$: We construct a directed graph on the vertex set $\{0,1\}^{kn+2}$. For each vertex u of the hypergraph there is a vertex v_u in the directed graph (e.g., $v_u = 0u0^{(k-1)n+1}$) and for each hyperedge e of the hypergraph there is a vertex v_e in the directed graph (e.g., $v_e = 1u_1 \dots u_m 10^{(k-m)n}$ where $e = \{u_1, \dots, u_m\}$ is ordered lexicographically). We put a directed edge from v_u to v_e iff $u \in e$ (i.e., iff e appears in the hyperedge list of u). All other vertices are isolated. Note that $0^{kn+2} = v_{0^n}$ has imbalance ℓ and for any vertex in $\{0,1\}^{kn+2}$ we can compute the predecessors and successors in polynomial time. If there is an imbalance modulo k in a vertex v_e, then the corresponding hyperedge e does not have size k. If there is an imbalance modulo k in a vertex v_u, then u has degree $\neq 0 \mod k$ in the hypergraph.

IMBALANCE-MOD-$k[\#\ell] \leq$ PARTITION-MOD-$k[\#\ell]$: Consider an instance of the problem IMBALANCE-MOD-$k[\#\ell]$. Split every vertex v into two vertices v_{in} and v_{out}, such that v_{in} gets all the incoming edges and v_{out} gets all the outgoing edges. If v was balanced, i.e., in-deg(v) = out-deg$(v) = d$, then we add $k - d$ edges from v_{out} to v_{in}. We can assume that out-deg$(0^n) = \ell$ and in-deg$(0^n) = 0$ (just create a copy of 0^n that takes in-deg(0^n) incoming and outgoing edges), and thus out-deg$(0^n_{out}) = \ell$ and in-deg$(0^n_{in}) = 0$. Note that we are using multi-edges, which are not allowed in the definition of the problem. However, this is not an issue, since this is just an intermediate step of the reduction. This new instance has the property that no vertex has both incoming and outgoing edges. Furthermore, any solution (i.e., a vertex with in- or out-degree not in $\{0, k\}$, except 0^n_{out}) yields a solution of the original instance.

Thus, we can assume wlog that the IMBALANCE-MOD-$k[\#\ell]$ instance (with multi-edges) is such that no vertex has both incoming and outgoing edges. We construct an instance of PARTITION-MOD-$k[\#\ell]$ on the set $\{0,1\}^{k+n}$. Every vertex u of the directed graph has k corresponding elements in the set $\{0,1\}^{k+n}$, namely $u_1 = 10^{k-1}u$, $\ldots$, $u_k = 0^{k-1}1u$. If u does not have any outgoing edges, then $u_1, \ldots, u_k$ form a subset of the partition, i.e., $C(u_i) = u_{i+1 \mod k}$. If u has outgoing edges to $v^{(1)}, v^{(2)}, \ldots, v^{(j)}$ ($j \leq k$, ordered lexicographically), then for every $i = 1, \ldots, j$ we put u_i in a subset that we denote $S_{v^{(i)}}$. $u_{j+1}, \ldots, u_k$ are put into isolated subsets, i.e., $C(u_i) = u_i$ for all $i = j+1, \ldots, m$. Note that if v has k incoming edges, then S_v will contain k elements. Given any element u_i, we can compute all the elements in its subset in polynomial time (and thus efficiently construct C that cycles through them in lexicographic order). Furthermore, since out-deg$(0^n) = \ell$, the vertices $0^n_{\ell+1}, \ldots, 0^n_k$ will be in singleton sets. Consider the subset of $\{0,1\}^{k+n}$ $X = \{u_i : u \in \{0,1\}^n, i \in \{1, \ldots, k\}\} \setminus \{0^n_{\ell+1}, \ldots, 0^n_k\}$. Then $|X| = k2^n - (k - \ell) = \ell \mod k$. It is easy to check that any element in X that is not contained in a subset of size k (according to C), must yield a solution to the IMBALANCE-MOD-$k[\#\ell]$ instance. Finally, the last step is to construct an efficient bijection between X and the set of all integers $\{j : 2^{n+k} - |X| \leq j < 2^{n+k}\}$, which is easy to do. Thus, we have reduced the original instance to an instance of PARTITION-MOD-$k[\#\ell]$ with inputs $m = 2^{n+k} - |X|$ and C (modified according to the bijection).

PARTITION-MOD-$k[\#\ell] \leq$ BIPARTITE-MOD-$k[\#\ell]$: Let us consider any instance (C, m) of PARTITION-MOD-$k[\#\ell]$ with parameter n. In particular, it holds that $m < 2^n$ and $2^n - m = \ell \mod k$. We construct a bipartite graph as follows. The vertex sets on the left and right side are $A = \{0,1\}^n$ and $B = \{0,1\}^n$ respectively. We can define a canonical partition of the numbers $m, m+1, \ldots, 2^n - 1$ into sets of size k (and one set of size ℓ). For example, $\{m, m+1, \ldots, m+\ell-1\}$, $\{m+\ell, m+\ell+1, \ldots, m+\ell+k-1\}$, etc. Each set of the canonical partition corresponds to a vertex in A as follows: a set containing k numbers $x_1 < x_2 < \cdots < x_k$ is represented by the vertex $x_1 \in A$.

For the set of size ℓ in the canonical partition, we introduce a special case: it is represented by $0^n \in A$. Note that many vertices in A will not correspond to any set of the canonical partition. For a number $x \geq m$, we let $L(x) \in A$ denote the vertex in A representing the set containing x in the canonical partition.

Another partition of the numbers $m, m+1, \ldots, 2^n - 1$ into sets of size at most k is given by the instance (C, m) of Partition-mod-$k[\#\ell]$. Similarly to what we did above, we can associate each set in the partition given by C with a vertex in B. We let $R(x) \in B$ denote the vertex of B representing the set containing x in the partition given by C. Note that for any vertex in A or B we can efficiently determine whether it represents a set of one of the two partitions and if so, which set exactly it represents.

For every $x \geq m$ we add an edge between $L(x) \in A$ and $R(x) \in B$, i.e., between the sets that contain x in the two different partitions. This construction might introduce multi-edges (if some x and y lie in the same set in both partitions) but this can easily be resolved by using the *Mitosis gadgets* described below. It is easy to see that for any vertex of the bipartite graph we can efficiently compute the set of all its neighbours. Finally, note that the vertex $0^n \in A$ has degree ℓ, and any other vertex with degree $\neq 0 \mod k$ must necessarily lie in B and correspond to a set in the partition given by C that contains strictly less than k elements. Thus any such vertex immediately yields a solution to the original Partition-mod-$k[\#\ell]$ instance.

Mitosis gadgets. Let $k \geq 2$. We now show how to construct a small bipartite graph such that exactly one vertex on each side has degree 1 and all other vertices have degree k (or 0). This "gadget" can then be used to increase the degree of two vertices (one on each side of the bipartite graph) without adding any solutions, i.e., vertices with degree $\neq 0 \mod k$.

The gadget is a bipartite graph with $k+1$ vertices on each side: $a_1, \ldots, a_{k+1}$ and $b_1, \ldots, b_{k+1}$. It contains all the edges $\{a_i, b_j\}$ for $i, j \leq k$, except the edge $\{a_k, b_k\}$. It also contains the edges $\{a_k, b_{k+1}\}$ and $\{a_{k+1}, b_k\}$. Thus, all vertices have degree k, except for a_{k+1} and b_{k+1} which have degree 1.

We call this the "Mitosis" gadget, because it allows us to duplicate edges that already exist. Let u and v be two vertices in a bipartite graph, one on each side. Furthermore, consider the case where there is an edge $\{u, v\}$. We would like to increase the degree of u and v by 1, but without introducing any new solutions, in particular without introducing any vertex with degree $\neq 0 \mod k$. Using the Mitosis gadget, we can just add new vertices $a_1, \ldots, a_k$ and $b_1, \ldots, b_k$, and identify a_{k+1} with u and b_{k+1} with v. Adding the corresponding vertices of the gadget yields a bipartite graph where the degree of u and v has increased by 1, but no new solutions have been introduced. Note that this gadget can, in particular, be used to turn a bipartite graph with multi-edges into one without them, without changing the degree of existing vertices and without adding any new solutions.

9.3 Relationship Between the Classes

In this section, we present some results that provide deeper insights into how the classes relate to each other. For any $k \geq 2$, $\mathsf{PF}(k)$ denotes the set of all prime factors of k. The main conceptual result is that PPA-k is entirely determined by the set of prime factors of k:

Theorem 9.2. *For any $k \geq 2$ we have* $\mathsf{PPA}\text{-}k = \mathop{\&}_{p \in \mathsf{PF}(k)} \mathsf{PPA}\text{-}p$.

This equation can be understood as saying the following:

- Given a single query to an oracle for PPA-k, we can solve any problem in PPA-p for any $p \in \mathsf{PF}(k)$
- Given a single query to an oracle that solves any PPA-p problem for any $p \in \mathsf{PF}(k)$, we can solve any problem in PPA-k.

Corollary 9.2. *In particular, we have:*

- *For $k_1, k_2 \geq 2$, if $\mathsf{PF}(k_1) \subseteq \mathsf{PF}(k_2)$, then $\mathsf{PPA}\text{-}k_1 \subseteq \mathsf{PPA}\text{-}k_2$.*
- *For all $k_1, k_2 \geq 2$, $\mathsf{PPA}\text{-}k_1k_2 = \mathsf{PPA}\text{-}k_1 \mathbin{\&} \mathsf{PPA}\text{-}k_2$.*
- *For all $k \geq 2$ and all $r \geq 1$ we have $\mathsf{PPA}\text{-}k^r = \mathsf{PPA}\text{-}k$.*

Using the PPA-$k[\#\ell]$ classes, we can formulate an even stronger and more detailed result. For any $k \geq 2$, $1 \leq \ell \leq k-1$, we define $\mathsf{PF}(k, \ell) = \mathsf{PF}(k/\gcd(k, \ell))$. In this case the conceptual result says that PPA-$k[\#\ell]$ is entirely determined by the set of prime factors of $k/\gcd(k, \ell)$.

Theorem 9.3. *For any integer constants k and ℓ with $k \geq 2$ and $0 < \ell < k$, it holds that*

$$\mathsf{PPA}\text{-}k[\#\ell] = \mathsf{PPA}\text{-}\left(\prod_{p \in \mathsf{PF}(k,\ell)} p\right)[\#1] = \bigcap_{p \in \mathsf{PF}(k,\ell)} \mathsf{PPA}\text{-}p.$$

The proof of Theorem 9.3 can be found in the next section. Before we move on to that, let us briefly show that Theorem 9.2 follows from Theorem 9.3.

Proof of Theorem 9.2. Using Lemma 9.2 and Theorem 9.3 we can write

$$\mathsf{PPA}\text{-}k = \mathop{\&}_{\ell=1}^{k-1} \mathsf{PPA}\text{-}k[\#\ell] = \mathop{\&}_{\ell=1}^{k-1} \left(\bigcap_{p \in \mathsf{PF}(k,\ell)} \mathsf{PPA}\text{-}p\right) = \mathop{\&}_{p \in \mathsf{PF}(k)} \mathsf{PPA}\text{-}p$$

where the last equality follows by noting that $\mathsf{PF}(k, \ell) \subseteq \mathsf{PF}(k)$ for all ℓ, and $\mathsf{PF}(k, k/p) = \{p\}$ for all $p \in \mathsf{PF}(k)$. □

9.3.1 Proof overview

Proof of Theorem 9.3. All containment results follow from Theorem 9.4 below, except

$$\mathsf{PPA}\text{-}\left(\prod_{p\in\mathsf{PF}(k,\ell)} p\right)[\#1] \supseteq \bigcap_{p\in\mathsf{PF}(k,\ell)} \mathsf{PPA}\text{-}p.$$

Let $\mathsf{PF}(k,\ell) = \{p_1,\ldots,p_d\}$. We will show how to combine a set of instances $(C_1,m_1),\ldots,(C_d,m_d)$, where (C_i,m_i) is an instance of Partition-mod-$p_i[\#1]$, into a single instance of Partition-mod-$s[\#1]$, where $s = p_1p_2\cdots p_d$, such that any solution to this instance yields a solution to one of the (C_i,m_i) instances. Without loss of generality, we can assume that the parameter n is the same for all (C_i,m_i) instances. Without loss of generality, we can assume that $2^n - m_i = 1 \mod s$ for all i, because we can add at most $\prod_{j\neq i} p_i$ sets of size p_i to achieve this (see the proof of Lemma 9.5). Note that we then have $(2^n-m_1)(2^n-m_2)\cdots(2^n-m_d) = 1 \mod s$. Furthermore, for $(x_1,\ldots,x_d)$ and $(y_1,\ldots,y_d)$ with $x_i,y_i \geq m_i$ we can define $x \equiv y$ if and only if $x_i \equiv_i y_i$ for all i, where $x_i \equiv_i y_i$ means that x_i and y_i lie in the same set in instance (C_i,m_i). If for all i, x_i lies in a set of size p_i in (C_i,m_i), then x will lie in a set of size s. Thus any solution yields a solution to one of the original instances. The details to fully formalise this are very similar to the proof of Lemma 9.6. □

Beame et al. [BIK+96] investigated the relative proof complexity of so-called "counting principles." These counting principles are formulas that represent the fact that a set of size $\neq 0 \mod k$ cannot be partitioned into sets of size k. They investigated the relationship between these principles in terms of whether one can be proved from the other by using a constant-depth, polynomial-size Frege proof. Their main result is a full characterisation of when this is possible or not. As noted by Johnson [Joh11], these counting formulas do not yield NP search problems, but they can be related to corresponding NP search problems (TFNP, in fact). Indeed, Johnson uses this connection to obtain some separation results between his PMOD^k classes (see Section 9.4) from Beame et al.'s negative results. Our contribution is using Beame et al.'s positive results in order to prove inclusion results about the PPA-$k[\#\ell]$ classes. More precisely, we modify their proofs to obtain polynomial-time reductions between our Partition-mod-$k[\#\ell]$ problems. Thus, we obtain the following analogous result:

Theorem 9.4. *Let $k_1,k_2 \geq 2$ and $0 < \ell_i < k_i$ for $i = 1,2$. If $\mathsf{PF}(k_2,\ell_2) \subseteq \mathsf{PF}(k_1,\ell_1)$, then PPA-$k_1[\#\ell_1] \subseteq$ PPA-$k_2[\#\ell_2]$.*

Proof. From Lemma 9.1 we know that PPA-$k_i[\#\ell_i] =$ PPA-$k_i[\#\gcd(k_i,\ell_i)]$ for $i = 1,2$. The result then follows from a few technical lemmas proved in Section 9.3.2 below:

$$\mathsf{PPA}\text{-}k_1[\#\gcd(k_1,\ell_1)] \underset{\text{L } 9.5}{\subseteq} \mathsf{PPA}\text{-}\frac{k_1}{\gcd(k_1,\ell_1)}[\#1] \underset{\text{L } 9.6}{\subseteq} \mathsf{PPA}\text{-}\frac{k_2}{\gcd(k_2,\ell_2)}[\#1] \underset{\text{L } 9.4}{\subseteq} \mathsf{PPA}\text{-}k_2[\#\gcd(k_2,\ell_2)]$$ □

9.3.2 Technical Lemmas for Theorem 9.4

The proof ideas from [BIK+96] are used to construct some of these reductions.

Lemma 9.3. *Let $k \geq 2$ and $r \geq 1$. For all $0 < \ell < kr$ with $\ell \neq 0 \mod k$, it holds that* PPA-$kr[\#\ell] \subseteq$ PPA-$k[\#(\ell \mod k)]$.

Proof. Consider any instance of Bipartite-mod-$kr[\#\ell]$. Split every vertex v into r versions $v_1, \ldots, v_r$ and assign every incident edge to exactly one of these versions, e.g., by ordering the neighbours in increasing lexicographic order. Then exactly one version of the known starting vertex will have degree $\ell \mod k$ and all its other versions will have degree 0 or k. In the new graph, any vertex that has degree not in $\{0, k\}$ (apart from this one version of the starting vertex) will immediately yield a solution of the original instance. Thus, we have reduced to an instance of Bipartite-mod-$k[\#(\ell \mod k)]$. □

Lemma 9.4. *Let $k \geq 2$ and $r \geq 1$. For any $0 < \ell < k$ it holds that* PPA-$k[\#\ell] \subseteq$ PPA-$kr[\#\ell r]$.

Proof. Consider any instance of Bipartite-mod-$k[\#\ell]$. Assign weight r to every edge. Clearly, the starting vertex now has degree ℓr and any other vertex with degree not in $\{0, kr\}$ yields a solution to the original instance. Finally, note that we can remove the weights without changing the degrees and adding any new solutions by using the "mitosis" gadgets. □

Lemma 9.5. *Let $k \geq 2$ and $\ell \geq 1$. Then* PPA-$k\ell[\#\ell] \subseteq$ PPA-$k[\#1]$.

Proof. We reduce Partition-mod-$k\ell[\#\ell]$ to Partition-mod-$k[\#1]$ by adapting the proof in [BIK+96, Lemma 2.3] to obtain a reduction. Consider an instance of Partition-mod-$k\ell[\#\ell]$, i.e., a partition of the set of integers $[m, 2^n - 1]$ given by a circuit C for some m such that $2^n - m = \ell \mod k\ell$. This means that there exists some integer $\alpha \in [0, 2^{n-1}]$ such that $2^n - m = \ell + \alpha k\ell$. Clearly, there exists some integer $\beta \in [0, (\ell-1)! - 1]$ such that $\alpha + \beta = 0 \mod (\ell-1)!$, which implies $2^n - m + \beta k\ell = \ell \mod (k \cdot \ell!)$ (if $\ell \leq 2$, then this holds with $\beta = 0$). Thus, if we add β sets of size $k\ell$, the size of the ground set will be $= \ell \mod (k \cdot \ell!)$. Assuming that n is large enough such that $2^n \geq k \cdot \ell! \geq \beta k\ell$, we can achieve this by letting $n' = n + 1$, $m' = m + 2^n - \beta k\ell$ and extending C to also include the additional β sets of size $k\ell$. It is easy to see that this can be done efficiently and will yield a partition of $[m', 2^{n+1} - 1]$ such that any mistake immediately yields a mistake in the original partition. Thus, we can assume without loss of generality that $2^n - m = \ell \mod (k \cdot \ell!)$.

Let S denote the set of all subsets of $\{m, m+1, \ldots, 2^n - 1\}$ of size exactly ℓ. Note that $|S| = \binom{2^n - m}{\ell} = \prod_{i=0}^{\ell-1} \frac{2^n - m - i}{\ell - i} = 1 \mod k$, since $2^n - m - i = \ell - i \mod k(\ell - i)$.

We will now describe how to construct a partition of S into sets of size k such that any mistake yields a solution of the original instance.

Recall that we can assume that $C^{k\ell}(x) = x$ for all x. Thus, every x yields an orbit $O(x) = \{x, C(x), C^2(x), \ldots, C^{k\ell-1}(x)\}$ of size at most $k\ell$. In particular, we can pick the lexicographically smallest element of every orbit to be its representative. Denote by $R(x)$ the representative of the orbit containing x. We then have that $R(x) = R(y)$, if and only if x and y lie in the same orbit, i.e., in the same set in the original partition. For $a, b \in S$ we write $a \equiv b$ if a and b contain exactly the same number of elements from each set of the original partition. This can be checked efficiently by computing the representative of each element in a, ordering these lexicographically (with repetitions), doing the same for b and checking if the two lists are identical. This is an equivalence relation and we denote the equivalence class of $a \in S$ under $\equiv$ by $[a]$.

For any $a \in S$ there exist distinct representatives $x_1, \ldots, x_s$, $s \leq \ell$, and $\alpha_1, \ldots, \alpha_s \geq 1$ with $\sum_{i=1}^{s} \alpha_i = \ell$ such that a contains exactly α_i elements from the orbit represented by x_i, for all i. Thus, the size of $[a]$ is exactly $\prod_{i=1}^{s} \binom{k\ell}{\alpha_i}$, assuming that the orbits of $x_1, \ldots, x_s$ all have size $k\ell$, i.e., do not yield a solution to the original problem. It was shown in the proof of [BIK+96, Lemma 2.3] that this quantity is a multiple of k. Thus, the equivalence class of a can be perfectly partitioned into sets of size k. We now describe a way to do this explicitly and efficiently. Assume that the representatives $x_1, \ldots, x_s$ are in increasing lexicographic order. Find the smallest index i such that k divides $\binom{k\ell}{\alpha_i}$. Let F denote an arbitrary fixed efficient bijection between $\{0, \ldots, \binom{k\ell}{\alpha_i} - 1\}$ and $\binom{O(x_i)}{\alpha_i}$, where this denotes the set of all subsets of $O(x_i)$ of size exactly α_i.

The circuit C' determines the image of $a \in S$ by first computing $x_1, \ldots, x_s$ and $\alpha_1, \ldots, \alpha_s$ as described above, and determining the smallest index i as explained above. Let $a_i = a \cap O(x_i)$. The circuit outputs

$$(a \setminus a_i) \cup F(\lfloor F^{-1}(a_i)/k \rfloor \cdot k + (F^{-1}(a_i) + 1 \mod k)).$$

It is easy to check that as long as $|O(x_j)| = k\ell$ for all j, this procedure partitions $[a]$ into sets of size k. The last step is to set $m' = 2^{n\ell} - |S|$ and construct an efficient bijection between S and $\{2^{n\ell} - |S|, \ldots, 2^{n\ell} - 1\}$, which is easy to do. □

Lemma 9.6. *Let $k_1, k_2 \geq 2$. If all prime factors of k_2 also divide k_1, then* PPA-k_1[#1] $\subseteq$ PPA-k_2[#1].

Proof. Similarly to our proof of Lemma 9.5, we adapt the proof of the corresponding statement for the counting formulas from Beame et al. [BIK+96, Lemma 2.5] in order to obtain a polynomial-time reduction.

Since all prime factors of k_2 divide k_1, there exists some ℓ (bounded by a constant, e.g., k_2) such that k_2 divides k_1^ℓ. From Lemma 9.3 we know that PPA-k_1^ℓ[#1] $\subseteq$ PPA-k_2[#1]. Thus, it suffices to show that PPA-k_1[#1] $\subseteq$ PPA-k_1^ℓ[#1]. We write $k = k_1$ from now on.

Consider an instance (C, m) of Partition-mod-k[#1]. Since $2^n - m = 1 \mod k$, it follows that there exists some r such that $(2^n - m)^r = 1 \mod k^\ell$ by Euler's totient theorem. Pick such an $r \geq \ell$ – any large enough multiple of the totient $\phi(k^\ell)$ will do – and note that r is bounded by some constant, since both ℓ and $\phi(k^\ell)$ are.

Assume without loss of generality that $C^k(x) = x$ for all x. We construct a partition of $\{m, \ldots, 2^n - 1\}^r$ explicitly as follows. For any $(x_1, \ldots, x_r) \in \{m, \ldots, 2^n - 1\}^r$, let $i \in \{1, \ldots \ell\}$ denote the largest index such that x_j is the lexicographically smallest element in the orbit of x_j under C (i.e., x_j is the representative of $O(x_j)$), for all $j < i$. If there is no such x_j, set $i = 1$. If $i > \ell$, set $i = \ell$. The circuit C' computes $C'(x_1, \ldots, x_r) = (C(x_1), C(x_2), \ldots, C(x_i), x_{i+1}, \ldots x_r)$. It is easy to see that if the orbits $O(x_j)$ under C all have size k, then this yields an orbit of size k^ℓ. Thus, any orbit in the new instance that has size different from k^ℓ immediately yields some orbit of the original instance that does not have size k.

The final step is to set $m' = 2^{nr} - (2^n - m)^r$ and construct an efficient bijection between $\{m', \ldots, 2^{nr} - 1\}$ and $\{m, \ldots, 2^n - 1\}^r$, which is easy to do. □

9.4 Johnson's PMOD^k Classes and Oracle Separations

Inspired by the definition of the PPA-complete problem Lonely (Section 2.3.2), Buss and Johnson [BJ12] defined TFNP problems called MOD^p to represent arguments modulo some prime p. Their main motivation was to use these problems to show separations (in the type 2 setting) between Turing reductions with m oracle queries and Turing reductions with $m + 1$ oracle queries. In his thesis [Joh11], Johnson generalised the definition of MOD^k to any $k \geq 2$ and defined corresponding classes PMOD^k. He also proved some separations between these classes and other TFNP classes in the type 2 setting (which yield oracle separations in the standard setting). It seems that Johnson was not aware of Papadimitriou's [Pap94] PPA-p classes.

In this section, we study the classes PMOD^k and prove a characterisation in terms of the classes PPA-p. In particular, we show that PMOD^k does not capture the full strength of arguments modulo k, when k is not a prime power. This characterisation also allows us to use Johnson's separations to obtain some oracle separations involving PPA-k and other TFNP classes.

Informally, the problem MOD^k can be defined as follows. We are given a partition of $\{0, 1\}^n$ into subsets and the goal is to find one of these subsets that has size $\neq k$.

If k is not a power of 2, then such a subset must exist. If k is a power of 2, then we instead consider $\{0,1\}^n \setminus \{0^n\}$ and the problem remains total. Formally, for any $k \geq 2$, the problem is defined as follows [BJ12; Joh11].

Definition 9.9. **MOD^k**:

Input: Boolean circuit $C : \{0,1\}^n \to \{0,1\}^n$.
Output:

- If k is not a power of 2: Find
  - $x \in \{0,1\}^n$ and $d \in \mathbb{N}$ such that $C^d(x) = x$ and $d|k$, $d \neq k$, or
  - $x \in \{0,1\}^n$ such that $C^k(x) \neq x$
- If k is a power of 2: Let additionally $C(0^n) = 0^n$ and find
  - $x \in \{0,1\}^n \setminus \{0^n\}$ and $d \in \mathbb{N}$ such that $C^d(x) = x$ and $d|k$, $d \neq k$, or
  - $x \in \{0,1\}^n$ such that $C^k(x) \neq x$

where $C^\ell(x) = C(C(\dots C(x)) \dots)$ (ℓ times) and $d|k$ means that d divides k.

Definition 9.10 (PMOD^k [Joh11])**.** For any $k \geq 2$, the class PMOD^k is defined as the set of all TFNP problems that many-one reduce to MOD^k.

Note that the problem MOD^k is a special case of our problem Partition-mod-k (which was indeed inspired by this definition). As a result, we immediately get that $\mathsf{PMOD}^k \subseteq \mathsf{PPA}$-$k$. Unless k is a prime power, we don't expect this to hold with equality. The intuition is that restricting the size of the base set to always be a power 2 has the effect of only achieving a subset of the possible ℓ-parameter values of PPA-$k[\#\ell]$. Namely, only $\ell \in \{2^n \mod k : n \in \mathbb{N}\}$ are achieved (for k not a power of 2).

Johnson proves a lemma [Joh11, Lemma 7.4.5] that gives some idea of how the PMOD^k classes relate to each other. It can be stated as follows: if $k = p_1 p_2 \dots p_r$, where the p_i are distinct primes, then $\mathsf{PMOD}^k = \cap_i \mathsf{PMOD}^{p_i}$. He proves this if all $p_i \neq 2$ and claims that the proof also works if some $p_i = 2$. However, if some $p_i = 2$ then the proof does not work. This is easy to see, since our results below prove that $\mathsf{PMOD}^6 = \mathsf{PMOD}^3$ which is not equal to $\mathsf{PMOD}^2 \cap \mathsf{PMOD}^3$, unless $\mathsf{PMOD}^2 \subseteq \mathsf{PMOD}^3$. However, Johnson proves that $\mathsf{PMOD}^2 \not\subseteq \mathsf{PMOD}^3$ in the type 2 setting.

The following result provides a full characterisation of PMOD^k in terms of the classes PPA-p.

Theorem 9.5. *Let $k \geq 2$.*

- *If k is not a power of 2, then $\mathsf{PMOD}^k = \mathsf{PPA}$-$\widetilde{k}[\#1] = \cap_{p \in \mathsf{PF}(\widetilde{k})} \mathsf{PPA}$-$p$ where $\widetilde{k}$ is the largest odd divisor of k.*

- *If k is a power of 2, then* $\mathsf{PMOD}^k = \mathsf{PPA}\text{-}2$.

The proof of Theorem 9.5 is given below in Section 9.4.1.

Corollary 9.3. *In particular, we have:*

- *for all primes p and $r \geq 1$,* $\mathsf{PMOD}^{p^r} = \mathsf{PPA}\text{-}p^r = \mathsf{PPA}\text{-}p$
- *for all $k \geq 2$,* $\mathsf{PMOD}^{2k} = \mathsf{PMOD}^k$
- *for all odd $k \geq 3$,* $\mathsf{PMOD}^k = \mathsf{PPA}\text{-}k[\#1] = \cap_{p \in \mathsf{PF}(k)} \mathsf{PPA}\text{-}p$

If k is a prime power, then PMOD^k is the same as $\mathsf{PPA}\text{-}k$. However, for other values of k, we argue that PMOD^k fails to capture the full strength of arguments modulo k. For example, $\mathsf{PMOD}^{15} = \mathsf{PPA}\text{-}15[\#1] = \mathsf{PPA}\text{-}3 \cap \mathsf{PPA}\text{-}5$, whereas $\mathsf{PPA}\text{-}15 = \mathsf{PPA}\text{-}3 \,\&\, \mathsf{PPA}\text{-}5$. This means that PPA-15 can solve any problem that lies in PPA-3 or PPA-5, while PMOD^{15} can only solve problems that lie *both* in PPA-3 and PPA-5. In particular, if $\mathsf{PMOD}^{15} = \mathsf{PPA}\text{-}15$, then it would follow that $\mathsf{PPA}\text{-}3 = \mathsf{PPA}\text{-}5$, which is not believed to hold (see oracle separations below). Even worse perhaps, is the fact that $\mathsf{PMOD}^{2k} = \mathsf{PMOD}^k$ for any $k \geq 2$. In particular, this means that $\mathsf{PMOD}^6 = \mathsf{PMOD}^3$, which indicates that PMOD^6 does not really capture arguments modulo 6.

Nevertheless, Johnson's oracle separation results (obtained from the corresponding type 2 separations as in [BCE+98]) also yield corresponding results for the PPA-k classes (using Theorem 9.5). We briefly mention a few of the results obtained this way. See Johnson [Joh11, Chapter 8] for additional results. Relative to any generic oracle (see [BCE+98]):

- $\mathsf{PPA}\text{-}p \not\subseteq \mathsf{PPA}\text{-}q$ for any distinct primes p, q
- $\mathsf{PPA}\text{-}k \not\subseteq \mathsf{PPP}$, $\mathsf{PPA}\text{-}k \not\subseteq \mathsf{PLS}$, $\mathsf{PPA}\text{-}k \not\subseteq \mathsf{PPADS}$ for any $k \geq 2$
- $\mathsf{PPP} \not\subseteq \mathsf{PPA}\text{-}p$, $\mathsf{PLS} \not\subseteq \mathsf{PPA}\text{-}p$ for any prime p

9.4.1 Proof of Theorem 9.5

For $k = 2$, MOD^2 corresponds to the PPA-complete problem Lonely [BCE+98], and thus $\mathsf{PMOD}^2 = \mathsf{PPA} = \mathsf{PPA}\text{-}2$. Let $r \geq 2$. Consider an instance (C, m) of Partition-mod-$2^r[\#(2^r - 1)]$ on the set $\{0,1\}^n$. Without loss of generality, assume $n \geq r$. Then $2^n = 0 \mod 2^r$ and thus $m = 2^n - (2^n - m) = -(2^r - 1) \mod 2^r = 1 \mod 2^r$. This means that we can (efficiently) partition $\{0, 1, \ldots, m-1\}$ into subsets of size 2^r, leaving only $0 = 0^n$ out. Thus, we have reduced Partition-mod-$2^r[\#(2^r - 1)]$ to MOD$^{2^r}$. Since $\mathsf{PPA}\text{-}2^r[\#(2^r - 1)] = \mathsf{PPA}\text{-}2$ (Theorem 9.3), we obtain $\mathsf{PPA}\text{-}2 \subseteq \mathsf{PMOD}^{2^r}$. On the other hand we also have $\mathsf{PMOD}^{2^r} \subseteq \mathsf{PPA}\text{-}2^r = \mathsf{PPA}\text{-}2$ by Corollary 9.2.

Consider some $k \geq 3$ that is not a power of 2. First, let us show that $\mathsf{PMOD}^{2k} = \mathsf{PMOD}^k$. MOD^{2k} reduces to MOD^k by splitting every subset into two subsets of size k (or less, if the subset has size $< 2k$). Conversely, consider an instance of MOD^k on the set $\{0,1\}^n$. Make a copy of the instance, thus obtaining an instance on the set $\{0,1\}^{n+1}$. For every subset of the original instance, take the union with its copy. If the subset had size k, the new subset has size $2k$. Thus, we have reduced to MOD^{2k}.

Let $k \geq 3$ be coprime with 2. We will show $\mathsf{PMOD}^k = \mathsf{PPA}\text{-}k[\#1]$. Consider an instance of MOD^k on the set $\{0,1\}^n$. Since k and 2 are coprime, there exists $i \in \{0,\ldots,k-1\}$ such that $2^{n+i} = 1 \mod k$ (e.g., by using Euler's theorem). Thus, we take 2^i copies of the instance and obtain an instance on the set $\{0,1\}^{n+i}$, which is an instance of $\textsc{Partition-mod-}k[\#1]$ (with $m = 0$), since $2^{n+i} = 1 \mod k$. Conversely, consider an instance (C, m) of $\textsc{Partition-mod-}k[\#1]$ on the set $\{0,1\}^n$. As before, there exists $i \in \{0,\ldots,k-1\}$ such that $2^{n+i} = 1 \mod k$. We construct an instance C' of MOD^k on $\{0,1\}^{n+i}$ as follows. The element $x \in \{0,1\}^n$ of the original instance corresponds to the element $1^i x \in \{0,1\}^n$ of the new instance. If $x \geq m$, set $C'(1^i x) = 1^i C(x)$. The number of elements that have not yet been assigned to a subset is $m + (2^i - 1)2^n = (m - 2^n) + 2^{n+i} = 0 \mod k$. Thus, we can efficiently partition them into subsets of size k without introducing any solution. We have obtained an instance of MOD^k.

9.5 Many-one vs Turing Reductions

Theorem 9.6. *For any prime $p \geq 2$, $\mathsf{PPA}\text{-}p$ is closed under Turing reductions.*

In particular, $\mathsf{PPA}\text{-}p^r = \mathsf{PPA}\text{-}p$ is also closed under Turing reductions. The proof of Theorem 9.6 can be found in Section 9.5.1. Furthermore, we also obtain:

Corollary 9.4. *For all $k \geq 2$ and $0 < \ell < k$, $\mathsf{PPA}\text{-}k[\#\ell]$ is closed under Turing reductions.*

Proof of Corollary 9.4. Using Theorem 9.3, we have $\mathsf{PPA}\text{-}k[\#\ell] = \bigcap_{p=1}^{d} \mathsf{PPA}\text{-}p_i$, where we let $\{p_1,\ldots,p_d\} = \mathsf{PF}(k,\ell)$. Consider a Turing reduction from some problem to $\mathsf{PPA}\text{-}k[\#\ell]$. Since $\mathsf{PPA}\text{-}k[\#\ell] \subseteq \mathsf{PPA}\text{-}p_i$, this yields a Turing reduction to $\mathsf{PPA}\text{-}p_i$, in particular. By Theorem 9.6, it follows that there exists a many-one reduction to $\mathsf{PPA}\text{-}p_i$, i.e., the problem lies in $\mathsf{PPA}\text{-}p_i$. Since this holds for all p_i, the result follows. □

If k is not a prime power, then it is not known whether $\mathsf{PPA}\text{-}k$ is closed under Turing reductions. Using our results from Section 9.4, we can actually provide an oracle separation between $\mathsf{PPA}\text{-}k$ and the Turing-closure of $\mathsf{PPA}\text{-}k$, i.e., an oracle under which $\mathsf{PPA}\text{-}k$ is not closed under Turing reductions. Let $R_1,\ldots,R_k$ be TFNP

problems. Following Johnson [Joh11] we define $\bigotimes_{j=1}^{k} R_j$ as the problem: given instances $(I_1, \ldots, I_k)$, where I_j is an instance of R_j, solve I_j for all j. As we did with the & operation, with a slight abuse of notation, we can also use the operation $\otimes$ with the PPA-k classes. In [Joh11, Theorem 7.6.1], Johnson proved that for $m \geq 2$ and distinct primes $p_1, \ldots, p_m$, $\bigotimes_{i=1}^{m} \text{MOD}^{p_i}$ does not many-one reduce to $\&_{i=1}^{m} \text{MOD}^{p_i}$ in the type 2 setting. Together with our Theorems 9.2 and 9.5 this yields:

Theorem 9.7. *Let $k \geq 2$ not a power of a prime. Relative to any generic oracle, it holds that $\bigotimes_{p \in \mathsf{PF}(k)} \text{PPA-}p \not\subseteq \text{PPA-}k$. In particular, relative to any generic oracle, PPA-k is not closed under Turing reductions.*

$S = \bigotimes_{p \in \mathsf{PF}(k)} \text{PPA-}p$ corresponds to solving PPA-p for all prime factors p of k *simultaneously*. In particular, this can be done by using $|\mathsf{PF}(k)|$ queries to PPA-k, i.e., a Turing reduction to PPA-k. Thus, S lies in the Turing closure of PPA-k, but not in PPA-k (relative to any generic oracle).

9.5.1 Proof of Theorem 9.6

We essentially apply the same technique that was used by Buss and Johnson [BJ12] to show that PPA, PPAD, PPADS and PLS are closed under Turing reductions.

Let Π be a problem that Turing-reduces to some problem in PPA-p. This means that there exists a Turing machine M with access to a PPA-p-oracle that solves Π in polynomial time. Since Imbalance-mod-p is PPA-p-complete (Theorem 9.1), we assume that the oracle provides solutions to Imbalance-mod-p instances. Our goal is to show that all the oracle queries can be combined into a single one. Indeed, a Turing reduction that always uses a single oracle query immediately yields a many-one reduction. Thus, by the definition of PPA-p, this would yield $\Pi \in$ PPA-p.

We begin by showing that any Imbalance-mod-p-instance can be efficiently transformed into an instance that has a particular form, namely: the starting node has imbalance $+1$ (in-degree 0 and out-degree 1), and any solution has imbalance -1 (in-degree 1 and out-degree 0). This can be achieved by the following steps:

1. Ensure that all vertices have in- and out-degree at most p (by splitting vertices into multiple copies).

2. Ensure that any unbalanced vertex has in- or out-degree 0 (by creating a copy that will take all the edges that yield the imbalance).

3. Since p is prime, we can ensure that the starting vertex has imbalance $+1$.

4. Ensure that all vertices that have imbalance $\neq 0 \mod p$, actually have imbalance $+1$ or -1 (by splitting every such vertex into p vertices, each getting at most one edge).

5. Transform every solution that has imbalance $+1$ into $p-1$ solutions with imbalance -1 instead (by pointing to $p-1$ new vertices).

From now on we assume that all IMBALANCE-MOD-p-instances have this form. Given an instance I of problem Π, let (G_1^I, s_1^I) denote the first oracle query made by M on input I, where G_1^I is the IMBALANCE-MOD-p graph (represented implicitly by circuits) and s_1^I is the starting vertex. From now on we omit the superscript I for better readability. For any solution t_1 to (G_1, s_1), let $(G_2(t_1), s_2(t_1))$ be the second oracle query made by M, if the first query returned t_1. We construct a big graph G that contains a copy of G_1 and a copy of $G_2(t_1)$ for each solution t_1 of (G_1, s_1). A vertex u in $G_2(t_1)$ is represented as (t_1, u) in G. For each such t_1, we add an edge from t_1 to $(t_1, s_2(t_1))$. Note that these two vertices are now balanced. Thus, the instance (G, s_1) has the following property: all solutions are of the form (t_1, t_2), where t_1 is a solution to (G, s_1), and t_2 is a solution to $(G_2(t_1), s_2(t_1))$. The straightforward generalisation of this construction for a polynomial number of queries (instead of 2), yields a graph G such that any solution yields consistent query answers for a complete run of M on input I. Thus, we obtain a Turing reduction that only needs to make one oracle query and then simulates M with these query answers.

It remains to show that this graph G can be constructed in polynomial time from I, i.e., we can efficiently construct circuits that compute the edges incident on any given node. This is easy to see, because any node contains enough information to simulate a run of M up to the point that is needed to determine the neighbours in G. We omit the full details, since the formal arguments are analogous to the ones in the corresponding proofs in [BJ12; Joh11].

9.6 Conclusion and Future Directions

In this chapter we have provided various tools that will hopefully enable further work on the classes PPA-k. One very interesting direction for future work is to try to show that FACTORING lies in the classes PPA-p for all primes p. Jeřábek [Jeř16] has shown that this is the case for $p = 2$, namely that FACTORING lies in PPA (under randomised reductions). Extending this to all other values of p would take us one step closer to proving that FACTORING lies in PPAD. It is unclear whether the techniques used by Jeřábek [Jeř16] could be used for this, or if radically new ideas are needed.

Chapter 10

Topological Characterisations of PPA-p

Recall that PPAD can be defined as the class of all TFNP problems that reduce to a computational problem associated to Brouwer's fixed point theorem, or its discrete version, Sperner's lemma. PPA can similarly be characterised using the Borsuk-Ulam theorem or its discrete counterpart: Tucker's lemma. These topological characterisations are very useful for proving membership in these classes, since the corresponding topological theorems are often used in preexisting proofs of existence. This usually means that membership can be obtained without much effort. However, these topological characterisations, and in particular their discrete versions, have also been pivotal in obtaining many of the celebrated hardness results for these classes. Examples include the PPAD-completeness of finding a Nash equilibrium [DGP09; CDT09], or the PPA-completeness of Consensus-Halving [FG19], the first natural PPA-complete problem.

In this chapter, we present the first such topological characterisations for the classes PPA-p for primes $p \geq 3$. Namely, we provide generalisations of the computational versions of the Borsuk-Ulam theorem and Tucker's lemma, parameterized by p, which are complete for PPA-p. A highlight of our generalisations is the PPA-p completeness of a computational version of the Bárány-Shlosman-Szücs (BSS) theorem [BSS81]. The BSS Theorem is perhaps the most well-known generalisation of the Borsuk-Ulam theorem. Besides [Alo87], it has been used to prove existence of other interesting problems, including a generalisation of the Kneser-Lovász Theorem [Kne55; Lov78] due to Alon, Frankl and Lovász [AFL86], a generalised van Kampen-Flores Theorem [Sar91] and a generalisation to Tverberg's Theorem, proven in [BSS81].
The strength of our results lies in that

- they solidify the status of the classes PPA-p as classes containing interesting well-known problems (adding to the recent results of Göös et al. [GKSZ20]), and

- they set up an essential toolkit for obtaining more completeness results for the classes, e.g., possibly the PPA-p-completeness of p-thief Necklace Splitting.

All of our PPA-p-membership results are obtained via a new combinatorial proof for $\mathbb{Z}_p$-versions of Tucker's lemma. This new proof can be seen as a natural generalisation of the standard combinatorial proof of Tucker's lemma by Freund and Todd [FT81]. Thus, as a byproduct of our techniques we also obtain the following results:

1. A combinatorial proof of the BSS theorem. The original proof by Bárány, Shlosman and Szücs [BSS81] is not combinatorial, as it uses various tools from algebraic topology. Using our new technique, we are able to provide the first combinatorial proof for this theorem.

2. A combinatorial proof of the Necklace Splitting theorem. The existence of such a combinatorial proof had been an open problem since [Alo87]. This open problem was solved by Meunier [Meu14] using a rather complicated argument. In contrast, our new combinatorial proof uses more elementary tools and is conceptually simpler than Meunier's proof. For more details, see Chapter 11.

3. A stronger statement of the continuous Necklace Splitting theorem which is called Consensus-1/p-Division [Alo87; SS03]. The main advantage of our new theorem is that it actually works for valuation functions that are not necessarily additive and non-negative, for details see Theorem 11.4 in Chapter 11.

As a result, we believe that this new technique is of independent interest and will be useful for providing combinatorial proofs of other topological existence theorems such as Dold's Theorem [Dol83].
Throughout this chapter, unless otherwise specified, k denotes an integer larger or equal to 2, and p denotes a prime number.

10.1 Topological Preliminaries

In this section, we include all the necessary notation and topological definitions that are used throughout this chapter. For a more detailed exposition on simplicial complexes, we refer the interested reader to [Mat08].

Notation: Recall that $B^n = \{x \in \mathbb{R}^n : \|x\|_2 \leq 1\}$ denotes the n-dimensional unit ball and $S^{n-1} = \partial B^n$ the corresponding unit sphere.

Definition 10.1 (Homeomorphism)**.** A *homeomorphism* of topological spaces $(X_1, \mathcal{O}_1)$ and $(X_2, \mathcal{O}_2)$ is a bijection $\phi : X_1 \to X_2$ such that for every $U \subseteq X_1$, $\phi(U) \in \mathcal{O}_2$ if and only if $U \in \mathcal{O}_1$. In other words, a bijection $\phi : X_1 \to X_2$ is a homeomorphism if and only if both ϕ and ϕ^{-1} are continuous. If there is a homeomorphism $\phi : X_1 \to X_2$, we write $X \cong Y$.

We say that a function f has *order* p if $f^p = f$, where the notation f^i denotes the composition of f by itself i times.

Definition 10.2 (Free Action)**.** Let $f : X \to X$ be a function of order p. We say that f acts freely on X if for all $x \in X$ and all $i \in \{1, \ldots, p-1\}$, $f^i(x) \neq x$.

Definition 10.3 (Equivariance)**.** Let $f : X \to X$ be a function of order p. A function $g : X \to X$ is f-equivariant on $P \subseteq X$, if for all $x \in P$, it holds that $g(f(x)) = f(g(x))$.

Definition 10.4 (Affine Independence)**.** We call the points $v_1, v_2, \ldots, v_k$ *affine dependent* if there exist numbers $a_1, a_2, \ldots, a_k \in \mathbb{R}$ not all 0 such that $\sum_{i=1}^k a_i v_i = \mathbf{0}$ and $\sum_{i=1}^k a_i = 0$. Otherwise, $v_1, \ldots, v_k$ are called *affine independent*.

Geometrically, some examples of simplices are points, lines and triangles. Formally, the definition requires the notion of affine independence.

Definition 10.5 (Simplex)**.** A *simplex* σ is the convex hull of a finite set A of affine independent vectors in $\mathbb{R}^n$. The points in A are called the *vertices* of σ and denoted by $V(\sigma)$. The dimension of σ is equal to $|A| - 1$. Namely, a k dimensional simplex, called k-simplex for short, has $k + 1$ vertices. The convex hull of an arbitrary subset of the vertices of σ is called a *face* of σ. A proper face of σ is called *facet*.

From the above definitions, it holds that every face is itself a simplex. For simplicity, we denote a simplex as the set of its vertices.

10.1.1 Simplicial Complexes and Triangulations

Very central to our results is the notion of *geometric simplicial complexes*, which are used to describe subspaces of $\mathbb{R}^d$. These subspaces consist of simple building blocks, such as points, line segments, triangles, tetrahedra, that are pasted together.

Definition 10.6 (Simplicial Complex)**.** A *simplicial complex* K is a non-empty set of simplices that satisfies the following properties:

- Each face of a simplex $\sigma \in K$ is also a simplex in K.

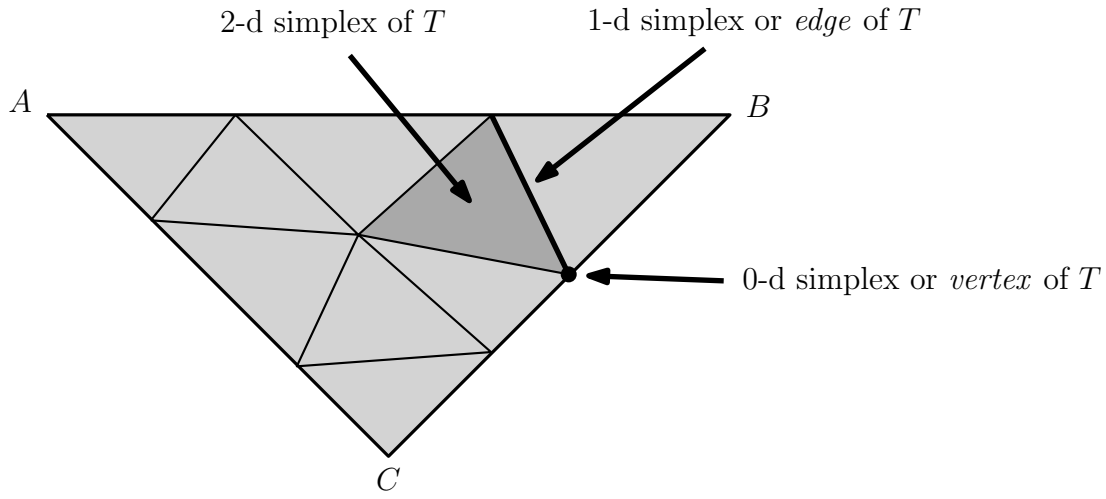


Figure 10.1: A simplicial complex T that defines a triangulation of the triangle $A-B-C$.

- The intersection $\sigma_1 \cap \sigma_2$ of any two simplices $\sigma_1, \sigma_2 \in K$ is a face of both σ_1 and σ_2.

The union of the simplices in K is called the polyhedron of K and is denoted by $\|K\|$. The dimension of K is $\dim(K) := \max_{\sigma \in K}\{\dim(\sigma)\}$ and the vertex set of K, denoted by $V(K)$, is the union of the vertex sets of all its simplices.

According to the above definition, zero-dimensional simplicial complexes correspond to points and one-dimensional simplicial complexes to sets of non-intersecting line segments as shown in Figure 10.1.

The notion of triangulation relates simplicial complexes with topological spaces.

Definition 10.7 (Triangulation)**.** A simplicial complex K is a triangulation of a topological space X if $\|K\| \cong X$.

For instance, the boundary of the n-simplex σ_n, namely a simplicial complex containing all proper faces of σ_n, is a triangulation of the sphere S^{n-1}.

Triangulation is a very powerful tool in studying the computational complexity of topological problems, because they allow us to partition a simplicial complex into smaller simplices that are connected in useful ways. We will mainly use the *Kuhn triangulation*, which is described in more detail below. We refer the interested reader to [Mat08] and [Mun84] for further information.

Definition 10.8 (Kuhn's Triangulation [Kuh60])**.** Kuhn's triangulation is a standard way to triangulate a domain that is a cube. For any $n \in \mathbb{N}$, the cube $[0,1]^n$ is triangulated with granularity $m \in \mathbb{N}$ as follows:

1. the set of vertices of the triangulation is U_m^n, where $U_m = \{0, 1/m, 2/m, \ldots, m/m\}$

2. every $x \in (U_m \setminus \{1\})^n$ is the base of the cube containing all vertices $\{y : y_i \in \{x_i, x_i + 1/m\}\}$

3. 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 $i \in [n]$ (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 $z^{\pi(1)} \leq \cdots \leq z^{\pi(k)}$ (componentwise).

- The restriction of Kuhn's triangulation of $[0,1]^n$ on some subspace $A_1 \times A_2 \times \cdots \times A_n$ of $[0,1]^n$, where for each $i \in [n]$, $A_i \in \{\{0\}, [0,1]\}$, coincides with Kuhn's triangulation of that subspace.

- Every n-dimensional simplex can be indexed by its smallest vertex (componentwise), which is also the base of the cube containing the simplex, and by the permutation that yields this simplex within this cube. Given some index, it is easy to check whether this is a valid index, and if so, output the vertices of the simplex.

- Given a point $x \in [0,1]^n$, we can efficiently determine the index of a simplex that contains it as follows. First find the base y of a cube of U_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 index of a simplex containing x.

- Given an n-simplex $\{z^0, \ldots, z^n\}$ and $i \in \{0, 1, \ldots, n\}$, we can efficiently compute the index of the other n-simplex that also has $\{z^0, \ldots, z^n\} \setminus \{z_i\}$ as a facet. This is called a *pivot* operation.

10.1.2 $\mathbb{Z}_p$-equivariant tie-breaking

For some of our constructions, we will require a tie-breaking that is $\mathbb{Z}_p$-equivariant and efficiently computable. Observe that non-efficient tie breaking rules exist by carefully choosing one representative for any equivalence class but this is not sufficient for our proofs. We define an efficiently computable rule below for any set $S \in 2^{\mathbb{Z}_p} \setminus \{\emptyset, \mathbb{Z}_p\}$, let $S + i := \{x + i : x \in S\}$.

Definition 10.9 ($\mathbb{Z}_p$-equivariant tie-breaking)**.** For any prime p, the $\mathbb{Z}_p$-equivariant tie-breaking function $T_p : 2^{\mathbb{Z}_p} \setminus \{\emptyset, \mathbb{Z}_p\} \to \mathbb{Z}_p$ is computed as follows on input $S \in 2^{\mathbb{Z}_p} \setminus \{\emptyset, \mathbb{Z}_p\}$:

1. For every $i \in \mathbb{Z}_p$, write $S + i$ as a bit-string of length p. Namely, construct the bit-string $b(i)$ where the jth bit from the left indicates whether $j \in S + i$ (for $j = 0, \ldots, p-1$).

2. Output $-i^* \in \mathbb{Z}_p$, where $i^* = \text{argmax}_i\, b(i)$ ($b(i)$ interpreted as a number in binary).

Lemma 10.1. *For any prime p, the $\mathbb{Z}_p$-equivariant tie-breaking function $T_p : 2^{\mathbb{Z}_p} \setminus \{\emptyset, \mathbb{Z}_p\} \to \mathbb{Z}_p$ is well-defined and satisfies for any $S \in 2^{\mathbb{Z}_p} \setminus \{\emptyset, \mathbb{Z}_p\}$:*

- $T_p(S) \in S$
- $T_p(S + i) = T_p(S) + i$ *for all* $i \in \mathbb{Z}_p$.

Proof. If p is prime and $S + i = S$ for some $i \in \mathbb{Z}_p \setminus \{0\}$, then $S \in \{\emptyset, \mathbb{Z}_p\}$. This follows from observing that the corresponding bit strings $b(0)$ and $b(i)$ must be equal and that this implies that the bits of $b(0)$ with index $\{k \cdot i\}_{k \in \mathbb{Z}_p}$ are all equal. Since p is prime, $\{k \cdot i\}_{k \in \mathbb{Z}_p} = \mathbb{Z}_p$.

Hence, $|\{S + i : i \in \mathbb{Z}_p\}| = p$ for any $S \in 2^{\mathbb{Z}_p} \setminus \{\emptyset, \mathbb{Z}_p\}$. Thus, the bit-strings $b(i)$ are all distinct, $T_p(S)$ is unique and T_p is well-defined. Next, by construction, it is easy to see that $T_p(S + i) = -(i^* - i) = T_p(S) + i$. Finally, since $S \neq \emptyset$, i^* will be such that $b(i^*)$ has a 1 in the left-most position. Thus, $0 \in S + i^*$, which implies that $-i^* \in S$. □

Example. Let $p = 3$ and $S = \{2\}$. Then, $S+0 = \{2\}$, $S+1 = \{0\}$, $S+2 = \{1\}$, and $b(0) = 010$, $b(1) = 001$ and $b(2) = 100$ (in binary). From Definition 10.9, $T_3(S) = 2$.

10.2 k-POLYGON-TUCKER: a PPA-k[#1]-complete Problem in 2D-space

In this section we present a generalisation of the Borsuk-Ulam Theorem in two dimensional space, the k-Polygon Borsuk-Ulam theorem, and its discrete counterpart, which we call *k-Polygon Tucker's Lemma*. We provide a combinatorial proof for the discrete Lemma, which allows us to put the corresponding computational problem k-POLYGON-TUCKER in PPA-k[#1]. Finally, by generalising existing techniques for showing PPA-hardness of Tucker's Lemma [ABB20], we prove that k-POLYGON-TUCKER is indeed PPA-k[#1]-complete. Recall from Section 9.3 that when $k = p^r$ is a prime power, PPA-k[#1] = PPA-k = PPA-p. Since the k-POLYGON-TUCKER problem is easy to state and two-dimensional, it is likely to be a useful tool for proving PPA-p-hardness.

10.2.1 The k-Polygon Borsuk-Ulam Theorem and k-Polygon Tucker's Lemma

We start this section with a unified description of Brouwer's Fixed Point Theorem, the Borsuk-Ulam Theorem and our generalisation: the k-Polygon Borsuk-Ulam Theorem. For this we will need the following definition.

Definition 10.10 (Rotation Operator)**.** We define the k-th *rotation operator* $\theta_k : \mathbb{R}^2 \to \mathbb{R}^2$ as follows: $\theta_k(x) = R_k x$, where R_k is the following two dimensional rotation matrix

$$R_k = \begin{bmatrix} \cos\left(-\frac{2\pi}{k}\right) & -\sin\left(-\frac{2\pi}{k}\right) \\ \sin\left(-\frac{2\pi}{k}\right) & \cos\left(-\frac{2\pi}{k}\right) \end{bmatrix}.$$

In other words, θ_k corresponds to a clockwise rotation by an angle of $2\pi/k$.

We now give a statement of Brouwer's Fixed Point Theorem that is different from the standard statement, but is well known to be equivalent to that (e.g., see [Mat08]).

Theorem 10.1 (Brouwer's Fixed Point Theorem, alternative statement)**.** *Let $f : B^2 \to \mathbb{R}^2$ be a continuous function such that $f(x) = x$ for all $x \in \partial B^2$. Then there exists $x^\star \in B^2$ such that $f(x^\star) = 0$.*

Next, using the same language, we give a statement of the Borsuk-Ulam Theorem.

Theorem 10.2 (Borsuk-Ulam Theorem)**.** *Let $f : B^2 \to \mathbb{R}^2$ be a continuous function such that $f(\theta_2(x)) = \theta_2(f(x))$ for all $x \in \partial B^2$. Then there exists $x^\star \in B^2$ such that $f(x^\star) = 0$.*

It is easy to see from the above expression that the Borsuk-Ulam Theorem is a generalisation of Brouwer's Fixed Point Theorem. This observation is in line with the fact that PPAD $\subseteq$ PPA, since Brouwer is complete for PPAD and Borsuk-Ulam is complete for PPA. In this section, we prove the following theorem for any $k \geq 3$.

Theorem 10.3 (k-Polygon Borsuk-Ulam Theorem)**.** *Let $f : B^2 \to \mathbb{R}^2$ be a continuous function such that $f(\theta_k(x)) = \theta_k(f(x))$ for all $x \in \partial B^2$. Then there exists $x^\star \in B^2$ such that $f(x^\star) = 0$.*

Clearly, the k-Polygon Borsuk-Ulam Theorem is also a generalisation of Brouwer's Fixed Point Theorem. We will show that the corresponding class is PPA-k[#1], and so this is in line with PPAD $\subseteq$ PPA-k[#1].

In order to state the discrete version of this theorem, we first provide some notation and definitions.

Definition 10.11 (k-polygon & Nice Triangulation)**.** For $k \geq 3$, let W_k be the regular k-polygon, i.e., regular k-gon, centered at $0 \in \mathbb{R}^2$ with radius 1. Let $u_1, \ldots, u_k$ denote the vertices of W_k, ordered such that $\theta_k(u_i) = u_{i+1 \pmod k}$ for all $i \in [k]$. We define $T^\star$ to be the triangulation of W_k that includes the simplices $\sigma_i = \mathsf{conv}(\{0, u_i, u_{i+1 \pmod k}\})$ for $i \in [k]$. We call a triangulation T *nice* if it satisfies the following two properties:

- it is a refinement of $T^\star$, and
- it is symmetric with respect to θ_k on the boundary. This means that for every edge $\psi \in T$ such that $\psi \subseteq \partial W_k$ it holds that $\theta_k(\psi) \in T$.

We can now state the corresponding discrete existence theorem.

Theorem 10.4 (k-Polygon Tucker's Lemma)**.** *Let $k \geq 3$ and fix some nice triangulation T of W_k. Suppose that every vertex $x \in T$ has a label $\lambda(x) \in \mathbb{Z}_k$ such that for any $y \in \partial T$ we have $\lambda(\theta_k(y)) = \lambda(y) + 1 \pmod k$. Then at least one of the following exists: (1) a triangle $\sigma \in T$ with vertices v_1, v_2, v_3 such that all the labels $\lambda(v_1)$, $\lambda(v_2)$, $\lambda(v_3)$ are different, or (2) an edge $\psi \in T$ with vertices v_1, v_2 such that $\lambda(v_1) - \lambda(v_2) \pmod k \notin \{0, 1, -1\}$.*

Remark. The topological theorems presented in this section can be proved by invoking Dold's Theorem from algebraic topology [Dol83]. However, the proof of Dold's theorem is not constructive and thus it is not useful for our purposes here. This is why we present a new combinatorial constructive proof of Theorem 10.4.

Before presenting the proof of Theorem 10.4 in the next section, we show that the k-Polygon Borsuk-Ulam Theorem and k-Polygon Tucker's Lemma are indeed equivalent.

Lemma 10.2. *The k-Polygon Borsuk-Ulam Theorem (Theorem 10.3) implies k-Polygon Tucker's Lemma (Theorem 10.4).*

Proof. We interpret each label $i \in \mathbb{Z}_k$ as the vector u_i, which is the i-th vertex of the polygon W_k. Let $h : W_k \to W_k$ be the piecewise linear extension of the function that has value $u_{\lambda(x)}$ on any vertex $x \in T$. Finally, we define $g : B^2 \to \mathbb{R}^2$ as the composition of h and a homeomorphism between W_k and B^2 that maps 0 to 0 and is θ_k-equivariant. Notice that by the definition of h and g and the equivariance assumption on λ, it holds that $g(\theta_k(x)) = \theta_k(g(x))$ for $x \in \partial B^2$. Then, it follows from Theorem 10.3 that there exists a $y^\star \in B^2$ such that $g(y^\star) = 0$ and hence there exists an $x^\star \in W_k$ such that $h(x^\star) = 0$.

Now, let $\sigma^\star$ be a triangle of T that contains $x^\star$. If the vertices of $\sigma^\star$ have only two consecutive labels, say 1 and 2 (corresponding to vectors u_1 and u_2), then it is impossible to have $h(x^\star) = 0$ since it is a non-zero linear interpolation of u_1 and u_2 and these vectors are linearly independent. Hence, it has to be that the vertices of $\sigma^\star$ have either two non-consecutive labels or three different labels and k-Polygon Tucker's Lemma follows. □

Lemma 10.3. *k-Polygon Tucker's Lemma (Theorem 10.4) implies the k-Polygon Borsuk-Ulam Theorem (Theorem 10.3).*

Proof. Let $h : W_k \to \mathbb{R}^2$ be the function obtained from f by using a homeomorphism between W_k and B^2 that fixes 0 and is θ_k-equivariant. Using standard compactness arguments that are used in the proof of both Brouwer's Fixed Point Theorem via Sperner's Lemma and the proof of Borsuk-Ulam via Tucker's Lemma, it is enough if for every $\varepsilon > 0$ we find a point x such that $\|h(x)\| \leq \varepsilon$, for details we refer to [Mat08]. Since h is a continuous function on a compact set, it is also uniformly continuous. Thus for every $\varepsilon > 0$ there exists a $\delta > 0$ such that for any $x, x' \in W_k$, if $\|x - x'\|_2 < \delta$, then $\|h(x) - h(x')\| < \varepsilon/k$. Assume that T is a nice triangulation of W_k such that any simplex $\sigma \in T$ has diameter at most δ.

We define the labelling $\lambda(x) = i$ to be equal to the index of the vertex u_i of W_k that is closest to $h(x)$. We break ties between i and $i + 1$ by picking i. Note that the only other kind of tie that can occur is if $h(x) = 0$. In that case all labels are tied. If $x = 0$, we pick one arbitrarily. Otherwise, i.e., if $x \neq 0$, we apply the same rule as for $h(x)$ above, but for x instead. It is easy to check that this tie-breaking is $\mathbb{Z}_k$-equivariant. For any $x \in \partial W_k$, because of the equivariance assumption on g, we have that $\lambda(\theta_k(x)) = \lambda(x) + 1 \pmod k$ and hence the assumptions of k-Polygon Tucker's Lemma are satisfied. Now we distinguish two cases.

$k = 3$. In this case, 3-Polygon Tucker's Lemma implies that there exists a triangle $\sigma \in T$ whose vertices v_1, v_2, v_3 have labels 1, 2, and 3 respectively. This implies the following set of inequalities by the definition of the labelling λ

$$\|h(v_1) - u_1\| \leq \min(\|h(v_1) - u_2\|, \|h(v_1) - u_3\|)$$
$$\|h(v_2) - u_2\| \leq \min(\|h(v_2) - u_1\|, \|h(v_2) - u_3\|)$$
$$\|h(v_3) - u_3\| \leq \min(\|h(v_3) - u_1\|, \|h(v_3) - u_2\|).$$

Hence, it is easy to see that the maximum angle between any of $h(v_1)$, $h(v_2)$ and $h(v_3)$ is at least $2\pi/3$. Now, assume for the sake of contradiction that for all v_1, v_2, v_3 it holds that

$$\|h(v_1)\| \geq \varepsilon, \ \|h(v_2)\| \geq \varepsilon, \ \|h(v_3)\| \geq \varepsilon.$$

This together with the fact that the maximum angle is at least $2\pi/3$ implies that the maximum distance is at least $2\sin(2\pi/6)\varepsilon$. But this implies that

$$\max(\|h(v_1) - h(v_2)\|, \|h(v_2) - h(v_3)\|, \|h(v_1) - h(v_3)\|) \geq \sqrt{3} \cdot \varepsilon > \frac{\varepsilon}{3}$$

which contradicts the definition of the triangulation T, where the vertices of the same simplex are at most δ far from each other and hence their images are at most $\varepsilon/k = \varepsilon/3$ far from each other. This implies that our assumption was wrong and hence for at least one $i \in [3]$ it holds that $\|h(v_i)\| \leq \varepsilon$ and the result for this case follows.

$k > 3$. In this case, k-Polygon Tucker's Lemma implies that there exists an edge $\psi \in T$ whose vertices v_1 and v_2 have labels that differ by more that one, without loss of generality assume that these labels are 1 and 3 respectively. By the definition of the labelling λ this implies that

$$\|h(v_1) - u_1\| \leq \min_i(\|h(v_1) - u_i\|), \quad \|h(v_2) - u_3\| \leq \min_i(\|h(v_2) - u_i\|)$$

Hence, it is easy to see that the angle between $h(v_1)$ and $h(v_2)$ is at least $2\pi/k$. Now, for sake of contradiction we assume that

$$\|h(v_1)\| \geq \varepsilon, \ \|h(v_2)\| \geq \varepsilon.$$

This together with the fact that the angle is at least $2\pi/k$ implies that the distance $\|h(v_1) - h(v_2)\|$ is at least $2\sin(2\pi/k)\varepsilon > \varepsilon/k$ which contradicts the definition of T. Therefore, in both cases for every $\varepsilon > 0$ we can find a $v \in W_k$ such that $\|h(v)\| \leq \varepsilon$. This, by standard arguments and the compactness of W_k, implies that there exists a point $x^\star \in W_k$ such that $h(x^\star) = 0$ and hence the result follows. □

10.2.2 Proof of k-Polygon Tucker's Lemma

In this section, we prove k-Polygon Tucker's Lemma (Theorem 10.4). The proof technique that we introduce here is a generalisation of the combinatorial proof of Tucker's lemma given by Freund and Todd [FT81].

We use a modulo-k argument to prove this theorem. We define a directed graph[1] where the nodes correspond to the simplices of T, and we also identify the symmetric edges of T on the boundary of W_k as the same node. Then, in this graph we describe a rule for defining edges such that there exist three types of nodes:

1. the node that corresponds to the 0-dimensional simplex $\{0\}$ which will have degree 1,
2. nodes that are balanced modulo k, i.e., outdegree $-$ indegree $= 0 \pmod k$,
3. nodes that are different from $\{0\}$ and are not balanced modulo k.

Due to a simple modulo-k argument and because $\{0\}$ is not balanced we can conclude that the constructed graph contains a node of type 3. Finally, we prove that all nodes of type 3 correspond to either a trichromatic triangle or a bichromatic edge with distinct non-consecutive labels and hence k-Polygon Tucker's Lemma follows. Note that the proof is combinatorial and is essentially a reduction to Imbalance-mod-k[#1]. This will allow us to obtain membership in PPA-k[#1] for the corresponding computational problem in the next section.

[1]To avoid any confusion, in this proof we refer to *nodes* of the directed graph, and to *vertices* of the triangulation.

Construction. For any simplex $\sigma \in T$ we define $S(\sigma)$ and $\lambda(\sigma)$:

- $S(\sigma) \subseteq [k]$ is the minimal subset of $[k]$ such that σ lies in the cone defined by $\{u_i : i \in S(\sigma)\}$,
- $\lambda(\sigma) = \{\lambda(x) : x \text{ is a vertex of } \sigma\}$,

and we let $S(\{0\}) = \emptyset$. A simplex $\sigma \in T$ is called *happy* if and only if $S(\sigma) \subseteq \lambda(\sigma)$.

Remark. Observe that because T is a refinement of T^*, we have that every simplex $\sigma \in T$ is contained in a cone defined by two consecutive vectors u_i, u_{i+1}. Hence, for $\sigma \neq \{0\}$, $S(\sigma)$ contains either a single number $i \in [k]$ or two consecutive numbers $i, i+1$.

We will define a graph G with node set $V(G) = T$. In G, we will only add directed edges to the following nodes, which we call *relevant*:

(a) nodes that correspond to a simplex $\sigma \in T$ such that $\sigma \notin \partial T$ and σ is happy,

(b) nodes that correspond to a simplex $\psi \in \partial T$ such that ψ is happy and:

- $\psi = \{u_1\}$, if ψ is 0-dimensional
- $\psi \subseteq \mathsf{conv}(\{u_1, u_2\})$, if ψ is 1-dimensional.

The reason for this distinction between simplices on the boundary and simplices not on the boundary is that we want to identify the symmetric simplices on the boundary as a *super node* in order to correctly use a modulo-k argument, as we described in the sketch of the proof.

We add an edge (v, v') to the graph G only if the simplices σ and σ' that correspond to v and v' are both relevant and one of the following rules applies:

1. **case $\boldsymbol{\sigma \notin \partial T, \sigma' \notin \partial T}$:** We add the edge if $\sigma' \subseteq \sigma$ and the labels of σ' suffice to make σ happy, i.e., $S(\sigma) \subseteq \lambda(\sigma')$,

2. **case $\boldsymbol{\sigma \notin \partial T, \sigma' \in \partial T}$:** Observe that since v' is relevant and $\sigma' \in \partial T$, v' is of type (b). So, instead of checking whether $\sigma' \subseteq \sigma$, we check whether there exists $t \in [k]$ such that $\tau := \theta_k^{(t)}(\sigma') \subseteq \sigma$ and τ suffices to make σ happy, i.e., $S(\sigma) \subseteq \lambda(\tau)$.

Directing the edges. The edge between σ and σ' is directed in the following natural way:

- if σ is 1-dimensional, then σ and σ' lie in $\mathsf{conv}(\{0, u_i\})$ for some i, and the edge is directed "away from 0." Formally, if $\sigma = \{z_0, z_1\}$ and $\sigma' = \{z_1\}$ are connected by an edge, then write $z_0 = \alpha_i u_i$ and $z_1 = \beta_i u_i$. If $\alpha_i - \beta_i > 0$, then the edge is incoming into σ. If $\alpha_i - \beta_i < 0$, then the edge is outgoing out of σ.

- if σ is 2-dimensional, then σ and σ' lie in $\mathsf{conv}(\{0, u_i, u_j\})$ for some i, j with $i - j = \pm 1 \pmod k$. If $j = i + 1$, then the edge is directed such that "we keep label i to our right, and label $j = i+1$ to our left, when we move in the direction of the edge." If $j = i - 1$, then the edge is directed such that "we keep label $j = i-1$ to our right, and label i to our left, when we move in the direction of the edge." Formally, if $\sigma = \{z_0, z_1, z_2\}$ and $\sigma' = \{z_1, z_2\}$ are connected by an edge, where $\lambda(z_1) = i$ and $\lambda(z_2) = j$, then write $z_0 = \alpha_i u_i + \alpha_j u_j$, $z_1 = \beta_i u_i + \beta_j u_j$ and $z_2 = \gamma_i u_i + \gamma_j u_j$. Construct the matrix

$$M = \begin{bmatrix} \alpha_i - \beta_i & \alpha_i - \gamma_i \\ \alpha_j - \beta_j & \alpha_j - \gamma_j \end{bmatrix}$$

  If $\det M > 0$, then the edge is incoming into σ. If $\det M < 0$, then the edge is outgoing out of σ. Notice that $\det M \neq 0$, because σ is a simplex. Furthermore, note that the determinant of the matrix does not change if we switch both i and j, and z_1 and z_2 (i.e., β and γ). Thus, the direction is well-defined.

If σ' corresponds to a node of type (b), then we apply the rule above to σ and τ instead.

Based on the description of the edges in the graph G we can prove the following Claims which complete the proof of k-Polygon Tucker's Lemma.

Claim 10.1. *The node that corresponds to the simplex $\{0\} \in T$ has outdegree 1 and indegree 0.*

Proof. Since $\{0\}$ is a 0-dimensional simplex, it does not have any sub-simplices and hence it can only be connected to a 1-dimensional simplex, i.e., an edge. An edge $\psi \in T$ that is not contained in a linear segment of the form $\mathsf{conv}(\{0, u_i\})$ cannot be a neighbour of $\{0\}$, because it requires two labels to become happy and $\{0\}$ has only one single label. Hence, $\{0\}$ can only be connected to a 1-dimensional simplex $\{0, z_i\}$ contained in $\mathsf{conv}(\{0, u_i\})$ for some i. Observe also that $S(\{0, z_i\}) = \{i\}$, which implies that $S(\{0, z_i\}) = \{\lambda(0)\}$. Therefore, there is exactly one 1-dimensional simplex that is connected to $\{0\}$, namely $\{0, z_i\}$, where $i = \lambda(0)$.

Furthermore, since $\{0\}$ and its neighbour $\{0, z_i\}$ lie in $\mathsf{conv}(\{0, u_i\})$, the edge is directed away from $\{0\}$. Formally, if we write $z_i = \alpha_i u_i$ and $0 = \beta_i u_i$, it will hold that $\alpha_i - \beta_i = \alpha_i > 0$. This means that the edge is incoming into $\{0, z_i\}$ and the claim follows. □

Claim 10.2. *Any node v in G that is unbalanced modulo-k, i.e.,* $\text{outdegree}(v) \neq \text{indegree}(v) \pmod{k}$ *and does not correspond to the simplex $\{0\}$, corresponds to a simplex σ that is either trichromatic or $\lambda(\sigma) \not\subseteq \{i, i+1\}$ for all $i \in [k]$.*

Proof. For this proof, we consider all three cases for the dimension of the simplex $\sigma^\star$ that corresponds to an unbalanced node of G separately.

Dimension of $\sigma^\star$ is 0. In this case, $\sigma^\star = \{z^\star\}$. It is easy to see that $\sigma^\star$ cannot make any 2-dimensional simplex happy since a 2-dimensional simplex σ has $|S(\sigma)| = 2$. So, $\sigma^\star$ can only be a neighbour of a 1-dimensional simplex ψ. Additionally, $\sigma^\star$ cannot make a 1-dimensional ψ happy if $|S(\psi)| = 2$. Thus, a neighbour of $\sigma^\star$ should be contained in the segment $\mathsf{conv}(\{0, u_i\})$ where i is such that $\lambda(\sigma^\star) = i$. If $z^\star \neq u_i$, then $\sigma^\star$ is connected to both of its two neighbouring 1-dimensional simplices in the segment $\mathsf{conv}(\{0, u_i\})$. Then, since the edges are directed away from 0, $\sigma^\star$ has one incoming and one outgoing edge. Formally, let $\{z_0, z^*\}$ and $\{z^*, z_0'\}$ be the two neighbouring 1-dimensional simplices, and write $z_0 = \alpha_i u_i$, $z_0' = \alpha_i' u_i$ and $z^* = \beta_i u_i$. It is easy to see that $\alpha_i - \beta_i$ and $\alpha_i' - \beta_i$ always have opposite signs, since z_0 and z_0' lie on opposite sides of $z^\star$ on $\mathsf{conv}(\{0, u_i\})$.

If $z^\star = u_i$, then from the definition of relevant nodes of G we have that $z^\star = u_1$ and $\{u_1\}$ is happy, i.e., $\lambda(u_1) = 1$. Thus, by the boundary conditions, for every $t \in [k]$, it holds that $\lambda(\theta_k^{(t)}(u_1)) = t + 1$. In other words, $\lambda(u_i) = i$ for all $i \in [k]$. As a result, it follows that for each $i \in [k]$, $\{u_1\}$ has an edge with the simplex $\{u_i, z_i\}$ which lies in $\mathsf{conv}(\{0, u_i\})$. This holds because $\theta_k^{(i)}(\{u_1\})$ suffices to make $\{u_i, z_i\}$ happy. Clearly, $\theta_k^{(i)}(\{u_1\})$ cannot make any other simplex happy, by the same arguments as above. Finally, note that all the edges are incoming into $\{u_1\}$, because edges are directed away from 0 on any $\mathsf{conv}(\{0, u_i\})$. Formally, if we write $z_i = \alpha_i u_i$ and $u_i = \beta_i u_i$, then we always have $\alpha_i - \beta_i = \alpha_i - 1 < 0$, so the edge is outgoing out of $\{u_i, z_i\}$. Thus, in this case, $z^\star$ has k neighbours and all of them with the same direction. Therefore, $z^\star$ is always balanced modulo-k. In conclusion, an unbalanced node of G different from $\{0\}$ cannot have dimension 0.

Dimension of $\sigma^\star$ is 1. First, assume that $\sigma^\star = \{z_1^\star, z_2^\star\}$ belongs to one of the line segments $\mathsf{conv}(\{0, u_i\})$. If additionally $\lambda(z_1^\star) = \lambda(z_2^\star)$, then $\sigma^\star$ is happy if $\lambda(z_1^\star) = \lambda(z_2^\star) = i$. Therefore, $|\lambda(\sigma^\star)| = 1$ and hence $\sigma^\star$ cannot be a neighbour of a 2-dimensional σ since $|S(\sigma)| = 2$ and $\sigma^\star$ cannot make σ happy. So, $\sigma^\star$ has exactly the two neighbours $\{z_1^\star\}$ and $\{z_2^\star\}$ since both of them make $\sigma^\star$ happy. Furthermore, the node is balanced, because one of the edges is incoming and the other one is outgoing, since edges are directed away from 0 on $\mathsf{conv}(\{0, u_i\})$. Formally, if we write $z_1^\star = \alpha_i u_i$ and $z_2^\star = \alpha_i' u_i$, then $\alpha_i - \alpha_i'$ and $\alpha_i' - \alpha_i$ always have opposite signs.

The next case we consider is when $\sigma^\star \subseteq \mathsf{conv}(\{0, u_i\})$ with $i = \lambda(z_1^\star) \neq \lambda(z_2^\star) = j$. If $i - j \neq \pm 1$, then $\sigma^\star$ yields a solution to k-Polygon Tucker's Lemma, and so is

allowed to be unbalanced. On the other hand, if $j = i \pm 1 \pmod{k}$, $\sigma^\star$ has exactly two neighbours, the simplex $\{z_1^\star\}$, which makes $\sigma^\star$ happy, and the unique 2-dimensional simplex $\sigma = \{z_1^\star, z_2^\star, z_3\}$ that contains $\sigma^\star$ as a face and is contained in the cone $\mathsf{conv}(\{u_i, 0, u_j\})$. Also, one of the edges is incoming and other one outgoing and hence $\sigma^\star$ cannot be unbalanced. Formally, write $z_2^\star = \beta_i u_i$, $z_1^\star = \beta_i' u_i$ and $z_3 = \alpha_i u_i + \alpha_j u_j$. Then, it holds that the edge goes from $\{z_1^\star\}$ to $\{z_1^\star, z_2^\star\}$ if $\beta_i' - \beta_i > 0$, and otherwise in the other direction. Furthermore, the edge goes from $\sigma = \{z_1^\star, z_2^\star\}$ to $\sigma = \{z_1^\star, z_2^\star, z_3\}$, if $\det M > 0$, where we can compute that $\det M = (\alpha_i - \beta_i)\alpha_j - (\alpha_i - \beta_i')\alpha_j = \alpha_j(\beta_i' - \beta_i)$. Since $\alpha_j > 0$, it follows that both expressions have the same sign, and thus $\sigma^\star$ is balanced.

The final case for 1-dimensional $\sigma^\star$ is when $\sigma^\star$ is not contained in any line segment of the form $\mathsf{conv}(\{0, u_i\})$. Let $\sigma^\star$ be contained in the cone $\mathsf{conv}(\{u_i, 0, u_{i+1}\})$. If $\sigma^\star$ is not relevant, then it cannot be unbalanced. To be relevant, and hence happy, it must hold that $\lambda(z_1^\star) = i$ and $\lambda(z_2^\star) = i + 1$. If $\sigma^\star$ is not in the boundary ∂T, then $\sigma^\star$ is the face of exactly two 2-dimensional simplices σ, σ' that are both contained in $\mathsf{conv}(\{u_i, 0, u_{i+1}\})$. Therefore, $\sigma^\star$ makes both of them happy and no other simplex, and consequently it has an edge with both of them and no other edge. Furthermore, one of the edges is incoming and other one is outgoing (from the perspective of $\sigma^\star$), so it cannot be unbalanced. Intuitively, if we let $\sigma = \{z_1^\star, z_2^\star, z_3\}$ and $\sigma' = \{z_1^\star, z_2^\star, z_3'\}$, then z_3 and z_3' lie on opposite sides of the line defined by $\{z_1^\star, z_2^\star\}$. Thus, the basis of $\mathbb{R}^2$ defined by $\{z_3 - z_1^\star, z_3 - z_2^\star\}$ has opposite orientation compared to the basis $\{z_3' - z_1^\star, z_3' - z_2^\star\}$. As a result, $\det M$ will have a different sign for the two edges. For a formal proof of this, we refer to the proof of Proposition 10.5, where we prove a more general version of this fact.

If $\sigma^\star \in \partial T$, then $\sigma^\star$ is relevant only if $\sigma^\star \subseteq \mathsf{conv}(\{u_1, u_2\})$. In this case, $\sigma^\star$ has exactly k neighbours each of which is a 2-dimensional simplex that has as a face one of the k symmetric copies of $\sigma^\star$. Namely, the neighbours of $\sigma^\star$ are $\sigma_1, \ldots, \sigma_k$, where $\theta_k^{(i)}(\sigma^\star)$ makes σ_i happy for all $i \in [k]$. To see that all k edges are incoming or all are outgoing, notice that if we have 1 to our right and 2 to our left when we reach the boundary, then it will hold that we have i on our right and $i + 1$ on our left when we reach the boundary at the corresponding position in cone $\mathsf{conv}(\{0, u_i, u_{i+1}\})$. More formally, the sign of the determinant of the matrix $M(\sigma_i, \theta_k^{(i)}(\sigma^\star))$ constructed to determine the direction of the edge with σ_i will be the same for all $i \in [k]$. For a full formal proof, we again refer to the proof of Proposition 10.5.

Dimension of $\sigma^\star$ is 2. Let $\sigma^\star = \mathsf{conv}(\{z_1^\star, z_2^\star, z_3^\star\})$. Assume that $\sigma^\star$ is contained in the simplex $\mathsf{conv}(\{u_i, 0, u_j\})$, with $j = i \pm 1 \pmod{k}$ and without loss of generality $\lambda(z_1^\star) = i$, $\lambda(z_2^\star) = j$. If $\lambda(z_3^\star) \notin \{i, j\}$ then $\sigma^\star$ is a trichromatic triangle, and thus it can be unbalanced. Assume that $\lambda(z_3^\star) \in \{i, j\}$ and without loss of generality

$\lambda(z_3^\star) = i$. In this case, $\sigma^\star$ has exactly two neighbours: the 1-dimensional simplices $\psi_1 = \mathsf{conv}(\{z_1^\star, z_2^\star\})$ and $\psi_2 = \mathsf{conv}(\{z_3^\star, z_2^\star\})$. Once again, one edge is incoming and the other one is outgoing, and thus $\sigma^\star$ is balanced. Intuitively, this follows from the fact that if we move with i to our left and j to our right, then we can "enter" the simplex from one side, and "exit" from the other one. More formally, if we write $z_1^\star = \alpha_i u_i + \alpha_j u_j$, $z_2^\star = \beta_i u_i + \beta_j u_j$ and $z_3^\star = \alpha'_i u_i + \alpha'_j u_j$, then it holds that:

$$\det \begin{bmatrix} \alpha_i - \beta_i & \alpha_i - \alpha'_i \\ \alpha_j - \beta_j & \alpha_j - \alpha'_j \end{bmatrix} = -\det \begin{bmatrix} \alpha'_i - \beta_i & \alpha'_i - \alpha_i \\ \alpha'_j - \beta_j & \alpha'_j - \alpha_j \end{bmatrix}$$

by using standard rules about the determinant.

Hence, the only unbalanced nodes different from $\{0\}$ correspond to either trichromatic simplices or simplices such that $\lambda(\sigma) \not\subseteq \{i, i+1\}$ for all $i \in [k]$. □

Finally, from the definition of the graph G and Claim 10.1, we have that there has to be a node in G that is unbalanced (mod k) and different from $\{0\}$. By Claim 10.2, any such node proves the validity of k-Polygon Tucker's Lemma.

10.2.3 The k-POLYGON-TUCKER problem

In this section, we define the computational problem associated with k-Polygon Tucker's Lemma, which we call k-POLYGON-TUCKER. Using the proof of k-Polygon Tucker's Lemma given in the previous section essentially we show that k-POLYGON-TUCKER lies in PPA-k[#1]. The hardness result is then proved in the next section. Note that it is also possible to define a computational problem directly associated with the k-Polygon Borsuk-Ulam Theorem, but for technical reasons the definition is a bit tedious, so it is not included here. For more details, see [FHSZ21].

To define the computational problem associated with k-Polygon Tucker's Lemma we have to fix for any $m \in \mathbb{N}$ a triangulation $T(m)$ of W_k with diameter (i.e., max distance between two vertices of a simplex) at most $1/2^m$. We pick the following natural triangulation.

Definition 10.12 (Edge Parallel Triangulation)**.** For every $m \in \mathbb{N}$, we define $\widehat{T}(m)$ to be the following nice triangulation of W_k. Starting from $T^\star$ (see Definition 10.11) we define a simplicial complex $\widehat{T}_i(m)$ of every simplex $\sigma_i^\star = \mathsf{conv}(\{u_i, 0, u_{i+1 \pmod k}\})$ and then set $\widehat{T}(m) = \cup_{i\in[k]} \widehat{T}_i(m)$. To define $\widehat{T}_i(m)$, we divide the edges $\psi_1^i = \mathsf{conv}(\{u_i, 0\})$, $\psi_2^i = \mathsf{conv}(\{u_i, u_{i+1 \pmod k}\})$ and $\psi_3^i = \mathsf{conv}(\{0, u_{i+1 \pmod k}\})$ of $\sigma_i^\star$ equally into 2^{m+1} intervals. Then, from any endpoint of the subintervals of the edge ψ_2^i we consider the lines that are parallel to either ψ_1^i or ψ_3^i and from any endpoint of the subintervals in ψ_1^i and ψ_3^i we consider the line that is parallel to ψ_2^i. We define $\widehat{T}_i(m)$ to be the set of simplices that are created by these lines and lie inside $\sigma_i^\star$. Namely, the intersection

points and endpoints of subintervals are the 0-dimensional simplices in $\widehat{T}_i(m)$, the line segments and the triangles between 0-dimensional simplices are also simplices in $\widehat{T}_i(m)$. It is simple to see that $\widehat{T}(m)$ is nice; we call $\widehat{T}(m)$ an *edge parallel triangulation* of W_k.

We assume that the labelling λ of the vertices of $\widehat{T}(m)$ is given via a Boolean circuit $\mathcal{L} : V(\widehat{T}(m)) \to \mathbb{Z}_k$, where $V(\widehat{T}(m))$ denotes the set of vertices of $\widehat{T}(m)$. We omit details on how to represent a vertex of $V(\widehat{T}(m))$ as a bit-string. We are now ready to define the total search problem that is associated with k-Polygon Tucker's Lemma.

Definition 10.13. k-POLYGON-TUCKER:

Input: Boolean circuit $\mathcal{L} : V(\widehat{T}(m)) \to \mathbb{Z}_k$.
Output: One of the following:

1. Two vertices x_1, x_2 on $\partial\widehat{T}(m)$ with $x_2 = \theta_k(x_1)$, but $\mathcal{L}(x_2) \neq \mathcal{L}(x_1) + 1 \pmod k$.
2. Three vertices x_1, x_2, x_3 of $\widehat{T}(m)$ such that the simplex $\sigma = \mathsf{conv}(\{x_1, x_2, x_3\})$ belongs to $\widehat{T}(m)$ and all the labels $\mathcal{L}(x_1)$, $\mathcal{L}(x_2)$, $\mathcal{L}(x_3)$ are different from each other.
3. Two vertices x_1, x_2 of $\widehat{T}(m)$ such that the edge $\psi = \mathsf{conv}(\{x_1, x_2\})$ belongs to $\widehat{T}(m)$ and the labels $\mathcal{L}(x_1)$, $\mathcal{L}(x_2)$ are different and satisfy $\mathcal{L}(x_1) - \mathcal{L}(x_2) \neq \pm 1 \pmod k$.

We prove the following theorem.

Theorem 10.5. *For any $k \geq 3$, k-*POLYGON-TUCKER *is* PPA-k[#1]*-complete.*

Recall from Chapter 9 that PPA-k[#1] $= \cap_{p \in \mathsf{PF}(k)}$PPA-$p$, where PF$(k)$ denotes the set of prime factors of k. In particular, this means that if $k = p^r$ is a prime power, then k-POLYGON-TUCKER is PPA-p-complete.

Theorem 10.5 follows from Proposition 10.1 below, and Proposition 10.2 which is stated and proved in the next section.

Proposition 10.1. *For any $k \geq 3$, k-*POLYGON-TUCKER *lies in* PPA-k[#1].

Proof. This follows from the proof of k-Polygon Tucker's Lemma that we presented in Section 10.2.2, which is essentially a reduction to the problem IMBALANCE-MOD-k[#1]. Note, in particular, that the node corresponding to $\{0\}$ always has imbalance exactly $+1$. Furthermore, it is easy to check that the edges of a node in the directed graph can be determined in polynomial time in size$(\mathcal{L})$. Thus, a Boolean circuit outputting successors and predecessors can be constructed efficiently. □

10.2.4 k-POLYGON-TUCKER is PPA-k[#1]-hard

In this section, we prove the following proposition by generalising the proof of PPA-hardness of Tucker's Lemma by Aisenberg, Bonet and Buss [ABB20].

Proposition 10.2. *For any $k \geq 3$, k-*POLYGON-TUCKER *is* PPA-k[#1]*-hard.*

Proof. To prove this hardness result we reduce from BIPARTITE-MOD-k[#1] (defined in Chapter 9). Recall that in this problem we are given a bipartite graph on the set of nodes $A \cup B$, where $A = \{0\} \times \{0,1\}^n$ and $B = \{1\} \times \{0,1\}^n$, such that the node $00^n \in A$ has degree 1. The goal is to find any other node that has degree $\neq 0 \mod k$. All nodes have degree in $\{0, 1, \ldots, k\}$ and we can assume (as explained in Section 9.2.2) that the circuit C which implicitly represents the bipartite graph is consistent, i.e., for all x, y we have $y \in C(x)$ iff $x \in C(y)$.

Consider the regular k-polygon W_k in $\mathbb{R}^2$ with the edge parallel triangulation (Definition 10.12). For any $i \in \mathbb{Z}_k$ let $R(i, i+1)$ denote the triangle with endpoints $\{\mathbf{0}, u_i, u_{i+1}\}$. In this proof we refer to *nodes* of the BIPARTITE-MOD-k[#1] instance and to *vertices* of the triangulation of W_k.

To every node $x \in (A \setminus \{00^n\}) \cup B$ we associate a distinct interval on the outer boundary of $R(1,2)$, i.e., on the edge $\mathsf{conv}(\{u_1, u_2\})$. This interval, which we denote by $K_1(x)$, is picked such that it covers $4k$ vertices of the triangulation. It is easy to see that we can pick the triangulation to be fine enough so that there are indeed enough vertices on $\mathsf{conv}(\{u_1, u_2\})$ to associate a distinct interval of $4k$ vertices to each node $x \in (A \setminus \{00^n\}) \cup B$. Since the triangulation is symmetric on the boundary with respect to θ_k, we can immediately also extend this association to the outer boundary of $R(i, i+1)$ for all other i. In other words, for any $i \in \mathbb{Z}_k$ and any $x \in (A \setminus \{00^n\}) \cup B$, we let $K_i(x) = (\theta_k)^{i-1}(K_1(x))$, which is an interval of $4k$ vertices on $\mathsf{conv}(\{u_i, u_{i+1}\})$. Thus, for any x there are k distinct intervals on the boundary associated to it, one in each of $\mathsf{conv}(\{u_i, u_{i+1}\})$, $i \in \mathbb{Z}_k$.

In the rest of this proof we explain how to assign a label in $\mathbb{Z}_k$ to every vertex of the triangulation so that the k-POLYGON-TUCKER boundary conditions are satisfied and any solution (i.e., a trichromatic triangle or an edge with non-consecutive labels) must contain a vertex that lies in some interval $K_i(x)$ where x is a solution-node of BIPARTITE-MOD-k[#1] (i.e., a node with degree $\neq 0 \mod k$). This will ensure that from any solution of the k-POLYGON-TUCKER instance we can easily obtain a solution-node of the BIPARTITE-MOD-k[#1] instance.

We begin by defining the "environment" label for every vertex of the triangulation. This corresponds to the standard label that the vertex will have, unless we specify it otherwise in the construction. Any vertex lying in $R(i, i+1) \setminus \mathsf{conv}(\{\mathbf{0}, u_{i+1}\})$ has the environment label i. Next, we define "cables," which will be used to embed the edges of the BIPARTITE-MOD-k[#1] instance in our construction.

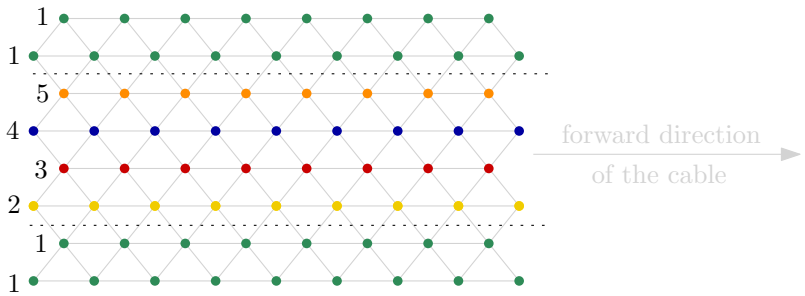


Figure 10.2: A cable for the case $k = 5$. The cable uses the labels $\mathbb{Z}_5 \setminus \{1\}$ and the environment has label 1. The labels are colour-coded as indicated on the left-hand side, i.e., green is label 1, yellow is label 2 etc. The forward direction of the cable is indicated by an arrow on the right-hand side. Note that the portion of the cable shown in the figure does not introduce any solution of 5-POLYGON-TUCKER.

Wires and Cables. A "wire" has an associated label $i \in \mathbb{Z}_k$ and it simply consists of a path of vertices in the triangulation such that all vertices on the path have the label i. A "cable" is made out of $k-1$ wires, where each wire has a distinct associated label. The wires are arranged in parallel inside the cable so that only wires with consecutive labels are allowed to touch. More precisely, if the cable uses labels $\mathbb{Z}_k \setminus \{i\}$, then the wires are arranged according to their labels in the order $i+1, i+2, \ldots, i+(k-1)$ from right to left, in the forward direction of the cable. Note that while wires are not directed, we can define a direction for every cable, based on the order of the wires inside the cable. A cable using the labels $\mathbb{Z}_k \setminus \{i\}$ is only allowed to exist inside a region with environment label i. This ensures that any vertex that is adjacent to either side of the cable, and thus to the wire labelled $i+1$ or $i+(k-1)$, is labelled i and does not introduce a solution. The construction of the cables ensures that the wires are "isolated" from each other and from the environment, in the sense that no solution is introduced along the cable. However, if the start or end of a cable using the labels $\mathbb{Z}_k \setminus \{i\}$ is allowed to "touch" the environment label i, then this will necessarily yield a trichromatic triangle at that point. See Figure 10.2 for an illustration of how a cable is constructed.

It is easy to see that a cable can turn without introducing solutions. Next, let us see how a cable can transition from one environment to another. Consider a cable in environment $i \in \mathbb{Z}_k$, i.e., it uses the labels $\mathbb{Z}_k \setminus \{i\}$ and lies in $R(i, i+1)$. If the cable arrives on the boundary $\mathsf{conv}(\{\mathbf{0}, u_{i+1}\})$ of $R(i, i+1)$, we can transform it into a cable that uses the labels $\mathbb{Z}_k \setminus \{i+1\}$ and continues into $R(i+1, i+2)$, i.e., in the environment $i+1$, on the other side of $\mathsf{conv}(\{\mathbf{0}, u_{i+1}\})$. Importantly, the direction of the cable does not change and we do not introduce any new solutions. This transformation of the cable is shown in Figure 10.3. The idea is simple. Consider a cable in environment i that arrives on $\mathsf{conv}(\{\mathbf{0}, u_{i+1}\})$ moving forward. The wires are arranged according to their labels in the order $i+1, i+2, \ldots, i+(k-1)$ from right

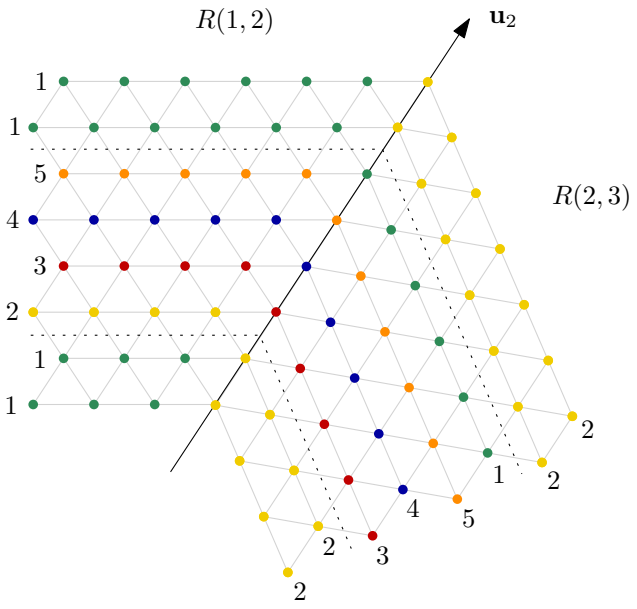


Figure 10.3: Transformation of a cable for the case $k = 5$. In the region $R(1,2)$ the cable uses labels $\mathbb{Z}_5 \setminus \{1\}$ and has environment 1. When the cable reaches region $R(2,3)$, the construction shown in the figure ensures that from now on, the cable uses labels $\mathbb{Z}_5 \setminus \{2\}$ and has environment 2. The cable uses the labels $\mathbb{Z}_5 \setminus \{2\}$ and the environment has label 2. Note that the transformation of the cable as shown in the figure does not introduce any solution of 5-POLYGON-TUCKER.

to left, in the forward direction of the cable. When the cable reaches $\mathsf{conv}(\{\mathbf{0}, u_{i+1}\})$, the right-most wire (which is labelled $i+1$) is dropped from the cable, i.e., it merges into the environment $i+1$ of $R(i+1, i+2)$. On the other side of the cable, a new wire with label $i = i+1+(k-1)$ is created by using the environment i of $R(i, i+1)$. Thus, we obtain a cable with wires $i+2, \ldots, i+1+(k-1)$ (from right to left) in the environment $i+1$, as desired. It is easy to see that this construction does not introduce any new solutions, because we have ensured that non-consecutive labels do not "touch."

When a cable starts or ends on the outer boundary of $R(i, i+1)$, i.e., on $\mathsf{conv}(\{u_i, u_{i+1}\})$, the k-POLYGON-TUCKER boundary conditions force a cable to start or end at the corresponding position in each of the regions $R(j, j+1)$, $j \in \mathbb{Z}_k \setminus \{i\}$. These $k-1$ cables will have the same direction as the original cable. In other words, if the cable in region $R(i, i+1)$ ends on the outer boundary, then the $k-1$ corresponding cables will also end on their corresponding boundary. Similarly, if the original cable starts on the boundary, then the $k-1$ corresponding cables will start on their corresponding boundary.

Construction of the instance. Before we begin, let us introduce the following useful terminology. Every node x of the BIPARTITE-MOD-k[#1] instance has some number $\ell \in \{0, 1, \ldots, k\}$ of neighbours, as given by $C(x)$. For any $i \in [\ell]$, we define

the "ith neighbour of x" to be the ith node in the lexicographically ordered list of neighbours of x.

We are now ready to begin describing the instance we construct. Recall that we have defined an environment label for every vertex of the triangulation except $\mathbf{0}$. Now consider the labelling where every vertex is simply labelled by its environment label. Clearly, this labelling satisfies the k-POLYGON-TUCKER boundary conditions. Furthermore, no matter how we pick the label of $\mathbf{0}$, there will be a solution there, since $\mathbf{0}$ is adjacent to all environments. Now it is easy to see that we can "move" this solution by locally modifying some of the labels. Namely, instead of having all labels meet at $\mathbf{0}$, we can instead construct a cable that uses the labels $\mathbb{Z}_k \setminus \{1\}$ and moves into the region with environment label 1. As a result, there is no longer a solution at $\mathbf{0}$, but instead there is now a solution at the end of the cable. In more detail, every environment label except 1 yields a wire with the corresponding label and the wires are arranged into a cable that uses the labels $\mathbb{Z}_k \setminus \{1\}$. Figure 10.4 illustrates this construction in the case $k = 5$.

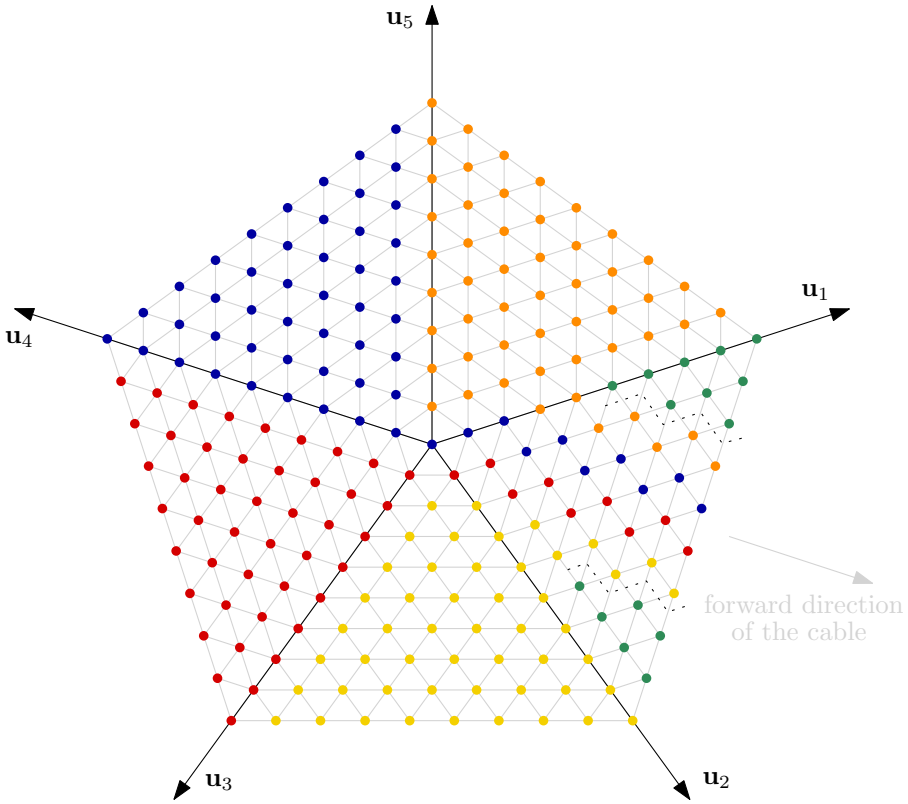


Figure 10.4: Construction around the origin for the case $k = 5$. Even though all the environments "meet" at the origin, the construction shown in the figure ensures that there is no solution around the origin, but instead a cable is created. The labels are colour-coded as in Figure 10.2 and Figure 10.3.

Recall that the node $00^n \in A$ has exactly one neighbour $y \in B$. Let $j \in [k]$ be the number such that 00^n is the jth neighbour of y. The cable that we just constructed

at the center of the instance will be routed so that it ends at $K_j(y)$ (a segment on the outer boundary of $R(j, j+1)$, as defined above). Furthermore, for any $x \in A \setminus \{00^n\}$ and $y \in B$ such that x and y are neighbours, there will be a cable starting at $K_i(x)$ and ending at $K_j(y)$, where $i, j \in [k]$ are such that x is the jth neighbour of y, and y is the ith neighbour of x. This ensures that the following properties hold.

Consider a node $x \in A \setminus \{00^n\}$ that has ℓ neighbours, where $\ell \in \{0, 1, \ldots, k\}$. If $\ell = 0$, i.e., x is an isolated node, then for all $i \in [k]$ there is no cable at $K_i(x)$. In particular, there is no k-POLYGON-TUCKER solution in any $K_i(x)$. If $\ell \in [k]$, then for all $i \in [\ell]$ there is a cable starting at $K_i(x)$ (and ending at some $K_j(y)$). For all $i \in [k] \setminus [\ell]$, there is a start of a cable at $K_i(x)$, but the cable just stops immediately and does not go anywhere. This ensures that the k-POLYGON-TUCKER boundary conditions are satisfied, but also that there is a k-POLYGON-TUCKER solution at $K_i(x)$ for all $i \in [k] \setminus [\ell]$ (because of an exposed end of cable). Thus, we obtain that

- if $\ell \in [k-1]$, there is a solution at $K_i(x)$ for some $i \in [k]$,
- if $\ell \in \{0, k\}$, then there is no solution in any $K_i(x)$.

Similarly, consider a node $y \in B$ that has ℓ neighbours, where $\ell \in \{0, 1, \ldots, k\}$. If $\ell = 0$, i.e., y is an isolated node, then for all $j \in [k]$ there is no cable at $K_j(y)$. In particular, there is no k-POLYGON-TUCKER solution in any $K_j(y)$. If $\ell \in [k]$, then for all $j \in [\ell]$ there is a cable ending at $K_j(y)$ (that started at some $K_i(x)$). For all $j \in [k] \setminus [\ell]$, there is an end of a cable at $K_j(y)$, but the cable just started there and does not come from anywhere else. This ensures that the k-POLYGON-TUCKER boundary conditions are satisfied, but also that there is a k-POLYGON-TUCKER solution at $K_j(y)$ for all $j \in [k] \setminus [\ell]$ (because of an exposed start of cable). Thus, we obtain that

- if $\ell \in [k-1]$, there is a solution at $K_j(y)$ for some $j \in [k]$,
- if $\ell \in \{0, k\}$, then there is no solution in any $K_j(y)$.

As a result, if the cables can indeed be constructed to connect the various $K_i(x)$ and $K_j(y)$ as desired, then any k-POLYGON-TUCKER solution of the instance will have to be next to some $K_i(x)$ such that x is a solution-node or next to some $K_j(y)$ such that y is a solution node. One immediate obstacle to routing the cables as desired is that we are working in two dimensions and it is very likely that cables will have to cross each other. Fortunately, there is a simple "trick" that has been used in prior work to resolve this issue [CD09]. Consider the following idea: cut the two cables that want to cross each other at the point of crossing. This creates two ends of cables and two starts of cables. It is easy to see that we can connect an end of cable with a start of cable, and the other end with the other start, so that no crossing occurs anymore. This modification of the cables is completely local and does not have any impact on the rest of the instance.

Constructing the labelling function. Since we want to be able to construct a circuit for the labelling function, we need to be a bit more precise about the path followed by every cable. One way to achieve this is to reserve a separate "circular lane" for each pair (y, j), where $y \in B$ and $j \in [k]$. A circular lane is a path of sufficient width that simply stays parallel to the outer boundary in each region $R(i, i+1)$ and thus makes a full "circle" around the center of the domain. By picking a fine enough triangulation, we can ensure that there is a separate, disjoint circular lane $L_{y,j}$ for each pair (y, j), where $y \in B$ and $j \in [k]$. Then the cable going from some $K_i(x)$ to some $K_j(y)$ will be routed as follows. Starting at $K_i(x)$, move perpendicularly to the outer boundary towards the inside of the domain, until the circular lane $L_{y,j}$ is reached. Next, follow the lane in clockwise direction. When following the lane, the cable might have to transition from some environment to the next and this is implemented as described earlier. The cable stops following the lane when it reaches the part of the lane lying just "above" $K_j(y)$. In other words, the cable stops following the lane when it is at the point where it can just turn left and move straight towards the boundary to end up at $K_j(y)$, as desired. Clearly, we can pick the triangulation fine enough so that this routing is indeed well defined.

This construction has two advantages. First of all, it ensures that any crossing involves at most two cables, not more. We can then use the trick described above to locally resolve these crossings. Furthermore, it ensures that we can construct a Boolean circuit computing the labelling function. Indeed, given any vertex of the triangulation we can easily determine on which circular lane and on which path perpendicular to the outer boundary it lies. This gives us enough information to then use the BIPARTITE-MOD-k[#1] circuit C as a sub-routine to determine whether the vertex lies on a cable and if so, how exactly the cable behaves locally (including possible crossing-avoiding trick). The circuit C only needs to be queried a constant number of times and thus the resulting circuit for the labelling will have polynomial size with respect to the size of C and n. We omit the full details, since this part of the proof is essentially the same as in prior work (see e.g., [CD09]). □

10.3 The BSS Theorem is PPA-p-complete

In this section, we prove that a computational problem associated with a topological theorem due to Bárány, Shlosman and Szücs [BSS81], that we call the BSS Theorem, is PPA-p-complete. The hardness result is obtained from the hardness of p-POLYGON-TUCKER proved in the previous section, while the membership in PPA-p follows from that of the more general problem $\mathbb{Z}_p$-STAR-TUCKER, presented in Section 10.4.

10.3.1 The BSS Theorem and Equivalent Formulations

Our notation follows that of Bárány, Shlosman and Szücs [BSS81]. Let $p \geq 2$ prime, $n \in \mathbb{N}$ and let $P = \{(v_1, v_2, \ldots, v_p) \in (\mathbb{R}^n)^p : \sum_{i=1}^{p} v_i = \mathbf{0}\}$ and $P_2 = \{u \in P : \|u\|_2 \leq 1\}$. It holds that

$$P \cong \mathbb{R}^{n(p-1)} \text{ and } P_2 \cong B^{n(p-1)}.$$

Note that P is a hyperplane of $\mathbb{R}^{np}$ with dimension $n(p-1)$. Let $\mathcal{B}$ be an orthogonal basis of P in $\mathbb{R}^{np}$ and let $\phi : P_2 \to B^{n(p-1)}$ be the function that maps $x \in P_2$ to its coefficients in base $\mathcal{B}$. Then, ϕ is a homeomorphism of P_2 and $B^{n(p-1)}$. Notice that $\phi(\partial P_2) = S^{n(p-1)-1}$ and that ϕ and ϕ^{-1} are efficiently computable. The same mapping shows that P and $\mathbb{R}^{n(p-1)}$ are homeomorphic.

Let

$$\theta(v_1, \ldots, v_p) = (v_p, v_1, \ldots, v_{p-2}, v_{p-1}).$$

The function θ has order p; namely the composition by itself p times is equal to the identity. Also, for all $i \in \{1, \ldots, p-1\}$ and all $x \in P \setminus \{\mathbf{0}\}$, $\theta^i(x) \neq x$. In other words, for any $i < p$, θ^i restricted to $P \setminus \{\mathbf{0}\}$ has no fixed points. Hence, θ acts freely on $P \setminus \{\mathbf{0}\}$. It follows with a similar argument that the function θ acts freely also on $P_2 \setminus \{\mathbf{0}\}$ and on ∂P_2.

The original statement of the BSS Theorem requires the notion of CW-complexes, but since in our work we do not use CW-complexes, we will not define them formally. Intuitively, a CW-complex consists of building blocks that can be topologically glued together.

Let p be a prime, $n \geq 1$ and X be a CW-complex consisting of p copies of the $n(p-1)$-dimensional ball glued on their boundaries.

Let $\alpha : \mathbb{R}^{n(p-1)} \to \mathbb{R}^{n(p-1)}$ such that $\alpha = \phi \circ \theta \circ \phi^{-1}$. Note that α acts freely on $\mathbb{R}^{n(p-1)} \setminus \{\mathbf{0}\}$, $B^{n(p-1)} \setminus \{\mathbf{0}\}$ and $S^{n(p-1)-1}$. Let ω be the extension of α on X defined as follows:

$$\omega(y, r, q) = (\alpha y, r, q + 1 \pmod p),$$

where (y, r, q) denotes the point of the q-th ball with radius r and direction $y \in S^{n(p-1)-1}$.

The map ω is a free action on X and the following theorem holds:

Theorem 10.6 (BSS Theorem, [BSS81])**.** *For the mapping ω and any continuous map $h : X \to \mathbb{R}^n$, there exists an $x \in X$ such that $h(x) = h(\omega x) = \cdots = h(\omega^{p-1} x)$.*

The following equivalent formulations of Theorem 10.6 are useful for defining the computational problem related to the BSS Theorem.

Theorem 10.7 (BSS Theorem, equivalent formulations)**.** *The following statements are equivalent to the BSS Theorem:*

1. *Let $g : P_2 \to P$ be continuous and such that $g(\theta x) = \theta g(x)$ for all $x \in \partial P_2$. Then, there exists $x \in P_2$ such that $g(x) = 0$.*
2. *Let $g : B^{n(p-1)} \to \mathbb{R}^{n(p-1)}$ be continuous and such that $g(\alpha x) = \alpha g(x)$ for all $x \in S^{n(p-1)-1}$. Then, there exists $x \in B^{n(p-1)}$ such that $g(x) = 0$.*

Proof. We first show that the two statements are equivalent and then that statement (2) is equivalent to Theorem 10.6.

(1) $\iff$ (2) The equivalence follows from $P_2 \cong B^{n(p-1)}$ and $P \cong \mathbb{R}^{n(p-1)}$, as well as from the equivariance of ϕ.

(2) $\iff$ (BSS) We first show that the BSS Theorem implies (2). Recall that the CW-complex X consists of p copies of $B^{n(p-1)}$ with their boundaries "glued" together. Define $h : X \to \mathbb{R}^n$ as follows: for $x = (y, r, i) \in X$, let $h(x) = [\phi^{-1} \circ g \circ \alpha^{1-i}(ry)]_{2-i} \in \mathbb{R}^n$, where for $j \in \mathbb{Z}_p \equiv [p]$, $[\cdot]_j$ denotes the j-th component of an element in $(\mathbb{R}^n)^p$ (i.e., $[(v_1, \ldots, v_p)]_j = v_j$). The mapping h is well-defined and continuous on the glued boundary, because for all i and $y \in S^{n(p-1)-1}$, we have $h(y, 1, i) = [\phi^{-1} \circ g \circ \alpha^{1-i}(y)]_{2-i} = [\theta^{1-i}\phi^{-1} \circ g(y)]_{2-i} = [\phi^{-1} \circ g(y)]_1$ (which does not depend on i). Finally, note that any $x = (y, r, 1) \in X$ with $h(x) = h(\omega x) = \cdots = h(\omega^{p-1}x)$ yields $z = ry \in B^{n(p-1)}$ with $[\phi^{-1} \circ g(z)]_1 = \cdots = [\phi^{-1} \circ g(z)]_p$, which implies $\phi^{-1} \circ g(z) = 0$ (by definition of P), and thus $g(z) = 0$.

Conversely, (2) implies BSS. We identify $B^{n(p-1)}$ with the first ball in the CW-complex X. Given a continuous function $h : X \to \mathbb{R}^n$, define $g : B^{n(p-1)} \to \mathbb{R}^{n(p-1)}$ by $g(x) = \phi(h(\omega^p x) - h(\omega^{p-1}x), h(\omega^{p-1}x) - h(\omega^{p-2}x), \ldots, h(\omega x) - h(x))$. It is easy to check that $g(\alpha x) = \alpha g(x)$ for all $x \in S^{n(p-1)-1}$ by noting that $\omega x = \alpha x$ for such x. □

10.3.2 The BSS-Tucker Lemma

In this section, we define a generalisation of Tucker's Lemma, that we call the *BSS-Tucker Lemma* and show that it is equivalent to the BSS Theorem. The BSS-Tucker Lemma applies to triangulations of P_2 that have some special properties.

For $j \in [n]$, let $e_j \in \mathbb{R}^n$ be the j-th unit vector. For $(i, j) \in \mathbb{Z}_p \times [n]$, let

$$e^{i,j} = \frac{1}{2(p-1)}(-e_j, \ldots, -e_j, (p-1)e_j, -e_j, \ldots, -e_j) \in P$$

where the term $(p-1)e_j$ is in the i-th position. For each $s = (s_1, \ldots, s_n) \in [p]^n$, we define the simplex $\sigma_s = \{\mathbf{0}\} \cup \{e^{i,j} \text{ s.t. for each } j \in [n], i \in \mathbb{Z}_p \setminus \{s_j\}\}$. Note that $|V(\sigma_s)| = (p-1)n + 1$.

Lemma 10.4. *$T^* = \{\tau : \tau \subseteq \sigma_s$ with $s \in [p]^n\}$ is a triangulation of P_2.*

Proof. Let $C = \mathsf{conv}\left((e^{i,j})_{(i,j)\in\mathbb{Z}_p\times[n]}\right)$. Then, by definition of T^*, $C \cong \|T^*\|$. So, the lemma follows from the fact that $C \cong P_2$. □

Definition 10.14. We say that a triangulation T of P_2 is *nice* if it satisfies the following two conditions:

- If $\sigma \in T \cap \partial P_2$ then $\theta\sigma \in T$.
- T refines the triangulation T^* (that is, for each $\sigma \in T$ there is $\tau \in T^*$ with $\sigma \subseteq \tau$).

Theorem 10.8 (BSS-Tucker Lemma)**.** *Let T be a nice triangulation of P_2. Let $\lambda = (\lambda_1, \lambda_2) : V(T) \to \mathbb{Z}_p \times [n]$ be a labelling such that for all $x \in \partial T$, $\lambda(\theta x) = (\lambda_1(x)+1, \lambda_2(x))$. Then, there exists a $(p-1)$-simplex σ of T such that $\lambda(\sigma) = \mathbb{Z}_p \times \{j\}$ for some $j \in [n]$, where $\lambda(\sigma) = \{\lambda(x) : x \in V(\sigma)\}$.*

Lemma 10.5. *The BSS Theorem (Theorem 10.7) implies the BSS-Tucker Lemma (Theorem 10.8).*

Proof. We interpret each label $(i, j) \in \mathbb{Z}_p \times [n]$ as the vector $e^{i,j}$ and we set g to be the extension of λ to a piecewise linear function on P_2. Notice that $g(\theta x) = \theta g(x)$ for $x \in \partial P_2$. Then, it follows from Theorem 10.7 that there exists an $x \in P_2$ such that $g(x) = 0$. The lemma follows by noting that any convex combination of different vectors $e^{i,j}$ that equals $\mathbf{0}$ must contain p vectors $e^{i_1,j_1}, \ldots, e^{i_p,j_p}$ such that $\{i_k, j_k\}_{k\in[p]} = \mathbb{Z}_p \times \{j\}$. Hence, the point x lies in a simplex with a $(p-1)$-dimensional face σ such that $\lambda(\sigma) = \mathbb{Z}_p \times \{j\}$ for some $j \in [n]$. □

Lemma 10.6. *The BSS-Tucker Lemma (Theorem 10.8) implies the BSS Theorem (Theorem 10.7).*

Proof. We show that the BSS-Tucker Lemma implies Statement (1) of Theorem 10.7. Using standard arguments, it suffices to show that for every $\varepsilon > 0$ we can find a point x such that $\|g(x)\|_\infty \le \varepsilon$. Since $g : P_2 \to P$ is a continuous function in a compact set, it is also uniformly continuous. Thus, for every $\varepsilon > 0$, there exists δ such that if $\|x - x'\|_2 < \delta$, then $\|g(x) - g(x')\|_\infty < \varepsilon/n$. Assume that T is a nice triangulation of P_2 with diameter at most δ.

For any point $x \in V(T)$, the label $\lambda(x) = (i^*, j^*)$ is defined as follows. For $(i, j) \in \mathbb{Z}_p \times [n]$, let $[g(x)]_{i,j}$ denote the (i, j)-coordinate of $g(x) \in P$. First, consider the case where $g(x) \neq \mathbf{0}$. Then, pick $j^* = \mathrm{argmax}_{j\in[n]} \max_{i\in[p]}[g(x)]_{i,j}$. Break ties by picking the smallest such j. Let $S = \{i : [g(x)]_{i,j^*} = \max_\ell [g(x)]_{\ell,j^*}\}$ and pick $i^* = T_p(S)$, where T_p is the $\mathbb{Z}_p$-equivariant tie-breaking function of Definition 10.9.

Note that $S \notin \{\emptyset, [p]\}$, because $g(x) \neq \mathbf{0}$ and by the choice of j^*. If $g(x) = \mathbf{0}$, then if $x = \mathbf{0}$, we assign an arbitrary label, and if $x \neq \mathbf{0}$, we use the same procedure as above but with x instead of $g(x)$.

With this definition of the labelling, it is easy to check that for any $x \in \partial T$, we always have $\lambda(\theta x) = (\lambda_1(x) + 1, \lambda_2(x))$, since $g(\theta x) = \theta g(x)$. Hence, by Theorem 10.8 there exists a $\sigma = \{x_1, \ldots, x_p\} \in T$ and a $j^* \in [n]$ such that $\lambda(x_i) = (i, j^*)$ for all $i \in [p]$.

The lemma follows by showing that there exists $i \in [p]$ such that $\|g(x_i)\|_\infty \leq \varepsilon$. First of all, note that for any $x \in P$, by definition we have that $\|g(x)\|_\infty \leq n \cdot \max_{i,j}[g(x)]_{i,j}$. Now, assume for the sake of contradiction that $\|g(x_i)\|_\infty > \varepsilon$ for all $i \in [p]$. Then, it follows that $[g(x_i)]_{i,j^*} > \varepsilon/n$ for all $i \in [p]$. But by the choice of diameter for the triangulation and uniform continuity of g, this implies that $[g(x_1)]_{i,j^*} > 0$ for all $i \in [p]$, which contradicts $x_1 \in P$. □

10.3.3 The p-BSS-Tucker problem

In this section, we define the computational problem p-BSS-Tucker corresponding to the BSS-Tucker Lemma and thus embodying the BSS Theorem. We prove that p-BSS-Tucker is PPA-p-complete. A computational problem that more directly captures the BSS Theorem can also be defined, but the definition is quite tedious, see [FHSZ21].

For computational purposes, it is convenient to use Kuhn's triangulation instead of some other ad-hoc triangulation. In order to use Kuhn's triangulation for the BSS-Tucker Lemma, we work on the domain

$$C_\infty = \{(c^1, \ldots, c^p) \in ([0,1]^n)^p | \forall j \in [n], \exists i \in [p] : c^i_j = 0\}$$

instead of P_2. These are coordinates with respect to the vectors $e^{i,j}$. Note that $C_\infty \cong P_\infty$, where $P_\infty = \{\sum_{i,j} c^i_j e^{i,j} | (c^1, \ldots, c^p) \in C_\infty\}$, and clearly $P_\infty \cong P_2$.

We can triangulate C_∞ by using Kuhn's triangulation (Definition 10.8) to triangulate (to the desired precision) each cube $\{(c^1, \ldots, c^p) \in C_\infty | \forall j \in [n] : c^{i_j}_j = 0\}$ for each $(i_1, \ldots, i_n) \in [p]^n$. By the properties of Kuhn's triangulation, it immediately follows that this yields a triangulation of C_∞. In particular, the triangulations of two cubes "match" on their common subspace. Since we constructed the triangulation separately on each cube, it follows that it refines the triangulation T^*. Furthermore, for any simplex σ lying on the boundary of C_∞, it follows that $\theta\sigma$ is also a simplex of the triangulation. This is easy to see, because θ just changes the order of the coordinates, and the Kuhn triangulation is invariant with respect to such transformations by definition. Thus, Kuhn's triangulation is indeed a nice triangulation. We let T^m_∞ denote Kuhn's triangulation of C_∞ of size m, as described above. We are now ready to define the computational problem.

Definition 10.15. p-**BSS-Tucker**:

Input: Boolean circuit $\lambda : V(T^m_\infty) \to \mathbb{Z}_p \times [n]$.
Output: One of the following:

1. A vertex $x \in \partial T^m_\infty$ such that $\lambda(\theta x) \neq (\lambda_1(x) + 1, \lambda_2(x))$.
2. A simplex $\sigma^* \in T^m_\infty$ such that $\lambda(\sigma^*) = \mathbb{Z}_p \times \{j\}$ for some $j \in [n]$.

We prove the following theorem.

Theorem 10.9. *For every prime p, the problem p-*BSS-Tucker *is* PPA-p*-complete.*

Theorem 10.9 follows firstly from Proposition 10.3 below which reduces from p-polygon-Tucker, and secondly from Lemma 10.7 and Proposition 10.5, presented in Section 10.4.

Proposition 10.3. *For any prime $p \geq 3$, p-*BSS-Tucker *is* PPA-p*-hard, even for fixed dimension $n = 1$.*

Note that for $p = 2$, p-BSS-Tucker corresponds to the standard version of Tucker's lemma, which is known to be PPA-hard for $n = 2$ [ABB20], but is easy for $n = 1$. For any prime p, it is easy to see that p-BSS-Tucker with dimension $n + 1$ is at least as hard as p-BSS-Tucker with dimension n [FHSZ21].

Proof. We prove that p-BSS-Tucker is PPA-p-hard for $n = 1$ by presenting a reduction from p-polygon-Tucker to p-BSS-Tucker with $n = 1$. Instead of the triangulation we used in the presentation of p-polygon-Tucker, we will use Kuhn's triangulation (Definition 10.8). It can be shown that the hardness of p-polygon-Tucker proved in Proposition 10.2, also holds if we use Kuhn's triangulation, since there is a simple homeomorphism between the two domains. We omit the details for this.

Let λ be an instance of p-polygon-Tucker with Kuhn's triangulation of size m. Thus, the domain for this problem can be written as

$$A = \{(c_1, \ldots, c_p) \in (U_m)^p | \exists i \in [p], \forall j \notin \{i, i+1\} : c_j = 0\}.$$

We construct an instance of p-BSS-Tucker with $n = 1$ on the domain C_∞ with Kuhn's triangulation of size m, denoted T^m_∞. Recall that the set of vertices can be written as $V(T^m_\infty) = \{(c_1, \ldots, c_p) \in U^p_m | \exists i \in \mathbb{Z}_p : c_i = 0\}$. Let $D := \cup_{i \in \mathbb{Z}_p} D_i$, where $D_i := \{(c_1, \ldots, c_p) \in V(T^m_\infty) | \forall j \notin \{i, i+1\} : c_j = 0\}$.

We are going to embed the p-polygon-Tucker domain A into D in the most natural way. Since we are using Kuhn's triangulation, the restriction of the triangulation T_∞^m to D corresponds to Kuhn's triangulation on that domain, and thus to Kuhn's triangulation of A. We define $\lambda' : V(T_\infty^m) \to \mathbb{Z}_p$:

$$\lambda'(c_1, \ldots, c_p) = \begin{cases} \lambda(c_1, \ldots, c_p) & \text{if } (c_1, \ldots, c_p) \in D \\ T_p(\{j : c_j = 0\}) & \text{otherwise} \end{cases}$$

where T_p is the $\mathbb{Z}_p$-equivariant tie-breaking function defined in Definition 10.9. Note that λ' is well-defined, because if $(c_1, \ldots, c_p) \in D$, then it corresponds to a point in the p-polygon-Tucker domain. Furthermore, if $(c_1, \ldots, c_p) \notin D$, then $|\{j : c_j = 0\}| \in [1, p-1]$, so is a valid input to T_p.

Since $\theta D = D$, $\lambda(\theta c) = \lambda(c) + 1$ and $T_p(\{j : (\theta c)_j = 0\}) = T_p(\{j : c_j = 0\}) + 1$, it follows that λ' satisfies the boundary conditions.

Let $c^1, c^2, \ldots, c^p$ be a $(p-1)$-simplex of T_∞^m that carries all the labels in $\mathbb{Z}_p$ (with respect to λ'). We now show how this yields a solution to the p-polygon-Tucker instance. Without loss of generality, we can assume that $c^1, c^2, \ldots, c^p$ are ordered in the order in which the simplex is defined by Kuhn's triangulation. Namely, for every $i \in \{1, \ldots, p-1\}$, there exists j_i such that $c_{j_i}^{i+1} = c_{j_i}^i + 1/m$ and $c_j^{i+1} = c_j^i$ for all $j \neq j_i$. Furthermore, the j_i are all distinct. Note that if for some i^*, $c^{i^*} \notin D$, then $c^i \notin D$ for all $i \geq i^*$. The following cases can occur:

- the simplex does not intersect D: it follows that $c^1 \neq 0 \in D$, and thus there exists j such that $c_j^1 > 0$. But then $c_j^i > 0$ for all i and the label j cannot be obtained by any vertex of this simplex.

- the intersection of the simplex with D is a face of dimension 2: then the three vertices of this face have pairwise distinct labels, and thus yield a solution to p-polygon-Tucker.

- the intersection of the simplex with D is a face of dimension 1: then it must hold that $c^1, c^2 \in D$ and $c^3, \ldots, c^p \notin D$. We distinguish between the two sub-cases:

  - $c_{j_1}^2 > 0$ and $c_j^2 = 0$ for all $j \neq j_1$. Then, since $c^3 \notin D$, it follows that $j_2 \notin \{j_1 - 1, j_1, j_1 + 1\}$. By definition of the j_i, we have that $c_{j_1}^3 > 0$ and $c_{j_2}^3 > 0$, which implies that $c^3, \ldots, c^p$ can only have labels in $\mathbb{Z}_p \setminus \{j_1, j_2\}$. As a result, c^1 and c^2 must have labels j_1 and j_2 (in any order). This yields a solution to p-polygon-Tucker, because the labels are distinct and non-consecutive.

  - there exists j^* such that $c_{j^*}^2 > 0$, $c_{j^*+1}^2 > 0$ and $c_j^2 = 0$ for all $j \notin \{j^*, j^*+1\}$. Since $c^3 \notin D$, it must hold that $j_2 \notin \{j^*, j^* + 1\}$. Thus we have $c_{j_2}^3 > 0$, and the vertices $c_3, \ldots, c_p$ can only obtain the labels $\mathbb{Z}_p \setminus \{j^*, j^* + 1, j_2\}$. It follows that the simplex cannot possibly be fully labelled.

- the intersection of the simplex with D is a face of dimension 0: then $c^2 \notin D$, and thus there exist distinct j, j' (and non-consecutive, but we don't need this here) such that $c^2_j > 0$ and $c^2_{j'} > 0$. But then, the vertices $c_2, \ldots, c_p$ can only obtain the labels $\mathbb{Z}_p \setminus \{j, j'\}$. It follows that the simplex cannot possibly be fully labelled.

This completes the proof. □

10.4 The $\mathbb{Z}_p$-STAR-TUCKER Lemma: Statement and PPA-p-completeness

In this section, we introduce a $\mathbb{Z}_p$-generalisation of Tucker's Lemma. We further define the associated computational problem $\mathbb{Z}_p$-STAR-TUCKER, and show that it is PPA-p-complete.

In $\mathbb{Z}_p$-STAR-TUCKER, the coordinates of the vertices lie on a star-like domain. This domain was used by Meunier [Meu14] for the first fully combinatorial proof of necklace splitting with p thieves. For more details on this, see Chapter 11.

For any prime p and any $m \geq 1$, we define $R_{p,m} = \{0\} \cup \{*^i j : i \in \mathbb{Z}_p, j \in [m]\}$, where $[m] = \{1, 2, \ldots, m\}$. For ease of notation, we also let $*^i 0 = 0$ for all $i \in \mathbb{Z}_p$. The symbols $*^1, \ldots, *^p$ should be interpreted as p different "signs" that will generalise the use of "+" and "−" in Tucker's Lemma. The way to picture $R_{p,m}$ is as follows: the point 0 lies at the center and there are p segments of length m leaving from 0 in p different directions. In that sense, we also call $R_{p,m}$ a p-star. The boundary of the p-star is the set of points $*^1 m, \ldots, *^p m$.

The $\mathbb{Z}_p$-action θ is defined on $R_{p,m}$ in the natural way, i.e., $\theta(*^i j) = *^{i+1} j$ (recall that $i \in \mathbb{Z}_p$). In particular, $\theta(0) = 0$. For any $d \geq 1$, θ can be extended to $R^d_{p,m} := (R_{p,m})^d$ by simply applying θ separately to each coordinate. Note that θ is a free action when restricted to the boundary of $R^d_{p,m}$ (i.e., the points that have at least one coordinate of the form $*^{\cdot} m$). See Figure 10.5.

There is a very natural metric on $R_{p,m}$. $\mathrm{dist}(*^{i_1} j_1, *^{i_2} j_2)$ is defined to be $|j_1 - j_2|$ if $i_1 = i_2$, and $j_1 + j_2$ otherwise. We let $\mathrm{dist}_\infty(\cdot, \cdot)$ denote the generalisation to $R^d_{p,m}$ (where we take the maximum). Finally, we triangulate the domain $R^d_{p,m}$ by using Kuhn's triangulation on every subcube (see Remark 10.1 below for details). We can now state a Tucker's lemma for this domain.

Theorem 10.10 ($\mathbb{Z}_p$-star Tucker's Lemma)**.** *Let p be prime and $m, t \geq 1$, $d = t(p-1)$, and T be Kuhn's triangulation of $R^d_{p,m}$. Let $\lambda : R^d_{p,m} \to R_{p,t} \setminus \{0\}$ be any labelling*[2] *that satisfies $\lambda(\theta x) = \theta\lambda(x)$ for all $x \in \partial R^d_{p,m}$. Then there exists a $(p-1)$-simplex $x_1, \ldots, x_p$ of T and $j \in [t]$ such that $\lambda(x_i) = *^i j$ for all $i \in [p]$.*

[2]Notice that the set $R_{p,t} \setminus \{0\}$ is isomorphic to the set $[p] \times [t]$.

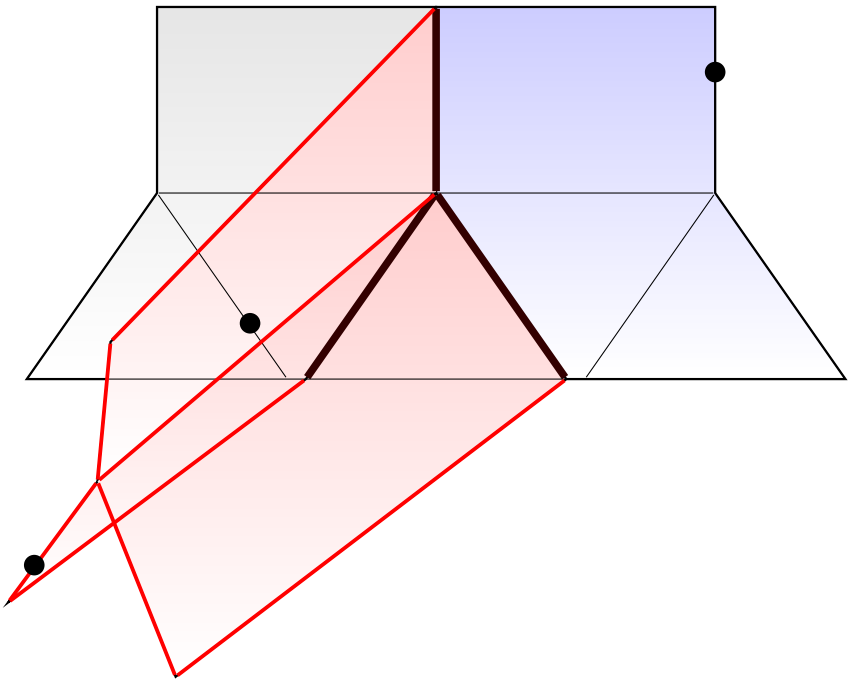


Figure 10.5: A view of the domain $R^d_{p,m}$ for $p = 3$ and $d = 2$. Note that this corresponds to $R_{3,m} \times R_{3,m}$. The three black points are in correspondence under θ. The three thick lines at the center of the picture correspond to the place where the three pieces are "glued" together.

In particular, by the properties of Kuhn's triangulation, it holds that for every solution $x_1, \ldots, x_p$, we have $\text{dist}_\infty(x_i, x_k) \leq 1$ for all $i, k \in [p]$.

Note that $\mathbb{Z}_2$-star Tucker's Lemma corresponds to the standard version of Tucker's Lemma. Since we are interested in the computational aspect, we also define the naturally corresponding computational problem.

Definition 10.16. $\mathbb{Z}_p$-STAR-TUCKER:

Input: $m, t \geq 1$, $d = t(p-1)$ and a Boolean circuit computing a labelling $\lambda : R^d_{p,m} \to R_{p,t} \setminus \{0\}$ that satisfies $\lambda(\theta x) = \theta\lambda(x)$ for all $x \in \partial R^d_{p,m}$.
Output: A $(p-1)$-simplex $x_1, \ldots, x_p \in R^d_{p,m}$ and $j \in [t]$ such that $\lambda(x_i) = *^i j$ for all $i \in [p]$.

Note that the property "$\lambda(\theta x) = \theta\lambda(x)$ for all $x \in \partial R^d_{p,m}$" can be enforced syntactically. Thus, $\mathbb{Z}_p$-STAR-TUCKER is not a promise problem.

The main result of the section is the following theorem.

Theorem 10.11. *For all primes p, $\mathbb{Z}_p$-STAR-TUCKER is PPA-p-complete.*

Theorem 10.11 follows from Proposition 10.4 below and Proposition 10.5 which is proved in the next section and is the main technical contribution of this chapter. The reduction in the proof of Proposition 10.5 also (implicitly) yields a proof of Theorem 10.10.

Remark 10.1. The domain $R^d_{p,m}$ can be triangulated in a standard way as follows. We start by subdividing the domain into hypercubes $\{*^{i_1}(a_1-1), *^{i_1}a_1\} \times \cdots \times \{*^{i_d}(a_d-1), *^{i_d}a_d\}$ for $a_1, \ldots, a_d \in [m]$ and $i_1, \ldots, i_d \in \mathbb{Z}_p$. Then, we can use Kuhn's triangulation on each hypercube.

Similarly to the triangulation used for p-BSS-Tucker, the triangulation T of $R^d_{p,m}$ has the following nice properties:

1. The restriction of T on any sub-orthant of $R^d_{p,m}$ (i.e., a subspace of the form $A_1 \times A_2 \times \cdots \times A_d$, where $A_\ell = \{*^{i_\ell} j : 0 \le j \le m\}$ or $A_\ell = \{0\}$) yields a triangulation of that sub-orthant.

2. On the boundary of $R^d_{p,m}$, the triangulation T is symmetric with respect to θ : for any simplex σ of T that lies on the boundary of $R^d_{p,m}$, the simplices $\theta\sigma, \theta^2\sigma, \ldots, \theta^{p-1}\sigma$ are also simplices of T (that also lie on the boundary).

3. T is computationally efficient, in the sense that we can perform pivoting and indexing operations in polynomial time.

Proposition 10.4. *For any prime p, $\mathbb{Z}_p$-star-Tucker is* PPA-p*-hard, even for fixed dimension $t = 1$.*

This follows from the following lemma, together with Proposition 10.3, which states that p-BSS-Tucker with dimension $n = 1$ is PPA-p-hard.

Lemma 10.7. *For any prime p, p-BSS-Tucker with dimension n reduces to $\mathbb{Z}_p$-star-Tucker with dimension $t = n$.*

Proof. Let λ be an instance of p-BSS-Tucker for some prime p and some $n \ge 1$, where we use Kuhn's triangulation of size m of the domain C_∞, denoted by T^m_∞.

In this case the domain of p-BSS-Tucker corresponds to

$$V(T^m_\infty) = \{(c^1, \ldots, c^p) \in U^{np}_m | \forall j \in [n], \exists i \in [p] : c^i_j = 0\}.$$

On the other hand, the domain of $\mathbb{Z}_p$-star-Tucker can be described as

$$A = \{(a^1, \ldots, a^p) \in U^{np(p-1)}_m | \forall j,k \in [n]\times[p-1], \exists i^* \in [p] : a^i_{j,k} = 0 \text{ for all } i \in [p]\backslash\{i^*\}\}.$$

Note that $\theta(c^1, \ldots, c^p) = (c^p, c^1, \ldots, c^{p-1})$ and $\theta(a^1, \ldots, a^p) = (a^p, a^1, \ldots, a^{p-1})$.

Define $\Psi : A \to V(T^m_\infty)$ as $\Psi(a^1, \ldots, a^p) = (\psi(a^1), \ldots, \psi(a^p))$, where $\psi_j(a^i) = \max\{a^i_{j,k} : k \in [p-1]\}$ for all $i \in [p]$, $j \in [n]$. Note that Ψ is well-defined, namely if $a \in A$, then $\Psi(a) \in V(T^m_\infty)$. Indeed, for any $j \in [n]$, it holds that $|\{i \in [p] | \psi_j(a^i) > 0\}| = |\{i \in [p] | \exists k \in [p-1] : a^i_{j,k} > 0\}| \le |\{(i,k) \in [p] \times [p-1] | a^i_{j,k} > 0\}| \le p-1$,

since for every $k \in [p-1]$ there exists at most one $i \in [p]$ such that $a^i_{j,k} > 0$ (for any fixed j). Furthermore, it is easy to see that $\Psi(\theta a) = \theta\Psi(a)$ by construction.

Now define the labelling $\lambda' : A \to \mathbb{Z}_p \times [n]$ by $\lambda'(a) := \lambda(\Psi(a))$. Since $\Psi(A) \subseteq V(T^m_\infty)$, λ' is well-defined. Furthermore, for any $a \in \partial A$, it holds that $\Psi(a) \in \partial V(T^m_\infty)$. Thus, we get that for any $a \in \partial A$, $\lambda'(\theta a) = \lambda(\Psi(\theta a)) = \lambda(\theta\Psi(a)) = \theta\lambda(\Psi(a)) = \theta\lambda'(a)$, i.e., λ' satisfies the boundary conditions.

If $\sigma = \{z_1, \ldots, z_p\}$ is a $(p-1)$-simplex of A such that $\lambda'(\sigma) = \mathbb{Z}_p \times \{j\}$ for some $j \in [n]$, then the set of vertices $\Psi(\sigma) = \{\Psi(z_1), \ldots, \Psi(z_p)\}$ satisfies $\lambda(\Psi(\sigma)) = \mathbb{Z}_p \times \{j\}$. Thus, the proof is completed by the following claim:

Claim. *If $\sigma = \{z_1, \ldots, z_p\}$ is a $(p-1)$-simplex in Kuhn's triangulation of A, then $\Psi(\sigma) = \{\Psi(z_1), \ldots, \Psi(z_p)\}$ is a simplex in Kuhn's triangulation T^m_∞.*

Proof. Since we use Kuhn's triangulation, without loss of generality, we can assume that $z_1, \ldots, z_p$ are ordered such that $z_1 \leq z_2 \leq \cdots \leq z_p$ (component-wise) and $\|z_1 - z_p\|_\infty \leq 1/m$. By construction of Ψ it holds that $\Psi(a) \leq \Psi(a')$ whenever $a \leq a'$. Thus, $\Psi(z_1) \leq \cdots \leq \Psi(z_p)$. In order to show that $\Psi(\sigma)$ is a simplex in Kuhn's triangulation T^m_∞, it remains to prove that $\|\Psi(z_1) - \Psi(z_p)\|_\infty \leq 1/m$. To see this, note that if $\|a - a'\|_\infty \leq 1/m$, then for all $i \in [p]$ and $j \in [n]$, we get that $|\psi_j(a^i) - \psi_j(a'^i)| = |\max\{a^i_{j,k} : k \in [p-1]\} - \max\{a'^i_{j,k} : k \in [p-1]\}| \leq 1/m$. □

This concludes the proof of Lemma 10.7. □

10.4.1 $\mathbb{Z}_p$-star-Tucker lies in PPA-p

In this section, we prove the following result.

Proposition 10.5. *For all primes p, $\mathbb{Z}_p$-star-Tucker lies in PPA-p.*

The result is proved by reducing the problem to Imbalance-mod-p. In particular, this also provides a combinatorial proof of $\mathbb{Z}_p$-star Tucker's Lemma (Theorem 10.10). The proof can be seen as a generalisation of the combinatorial proof of Tucker's Lemma given by Freund and Todd [FT81]. We begin with an overview of the proof.

Proof Overview. As noted earlier, $\mathbb{Z}_2$-star-Tucker corresponds to the standard version of Tucker's Lemma and the domain is equivalent to $\{-m, -(m-1), \ldots, 0, \ldots, m\}^d$. The computational problem is known to lie in PPA (recall that PPA = PPA-2) by using an argument given by Freund and Todd [FT81]. More precisely, Freund and Todd gave a constructive proof of Tucker's Lemma and as noted by [Pap94; ABB20], this yields a reduction to PPA. The constructive proof relies on a path-following argument on a graph where the nodes are simplices of the triangulation. We start by giving some details about their argument, since our proof is a generalisation of their construction.

The nodes of the graph G consist of all simplices of the triangulation that satisfy some properties that depend on the labels of the simplex (and the coordinate subspace orthant in which the simplex lies). Following the presentation of the proof given by Matoušek [Mat08], we call these "happy simplices." Undirected edges are added between happy simplices based on some simple rules (e.g., if they share a facet and that facet has some desired labels, etc). Given the definition of the edges, it is easy to show that the happy simplex 0^d has degree 1 and any other node of degree 1 is either a solution, or is a happy simplex lying on the boundary of the domain. In the latter case, because of the boundary conditions, this means that there must be another such simplex that lies on the antipodally opposite side of the domain and also has degree 1. Thus, merging these two nodes into a single one yields a vertex of degree 2, eliminating these "fake" solutions.

Our first contribution is to note that the edges of the graph can be directed in a *consistent* way in Freund and Todd's construction. Namely, any non-merged degree-2 vertex has one incoming and one outgoing edge, and any merged vertex has either two incoming edges, or two outgoing edges. This yields a reduction to Imbalance-mod-2. Note that if all degree 2 vertices were always perfectly balanced, then we would obtain a reduction to End-of-Line, which is impossible, unless $\mathsf{PPAD} = \mathsf{PPA}$.

When we move to the case $p > 2$, the notions used to define the graph can be generalised in a natural way, despite the unusual domain $R^d_{p,m}$. While the ability to direct edges was not actually needed for $p = 2$, it now becomes absolutely necessary. Indeed, for any degree-1 happy simplex on the boundary, there are now $p - 1$ other such simplices (by using θ). Merging these into a single node yields degree p. We show that directing the edges yields a merged node that is balanced modulo p (namely, all p edges are incoming, or they are all outgoing).

However, another difficulty arises for $p > 2$. Recall that a path can visit simplices of various dimensions. The vertices where the dimension changes are special vertices, that we call super-happy simplices. These super-happy simplices have one edge with a same or lower-dimensional happy simplex, and k edges with k different higher-dimensional happy-simplices, where $k \in [p-1]$. Directing the edges as before, yields that the k edges are directed the same way, and in the opposite direction to the single edge. By changing the way the direction of edges is defined, it is possible to salvage the situation for $p = 3$. However, this fails for any $p \geq 5$.

The solution is to carefully assign *weights* to all edges. The weight of an edge only depends on the nature of the coordinate subspace orthants in which it lies, in particular the dimension. With these weights, we show that any vertex that is not a solution is now balanced modulo p (except the trivial solution 0^d). Namely, the non-solution vertices of the graph are:

- the trivial solution 0^d: all its edges are outgoing and it has degree $(-1)^t \mod p$
- the merged simplices on the boundary: p edges, all incoming/outgoing, all the same weight
- happy, but not super-happy simplices: once incoming edge, one outgoing, both same weight
- super-happy simplices: one incoming edge with weight w and $k \in [p-1]$ outgoing edges, each with weight w/k (or opposite direction for all edges)

Thus, apart from the trivial solution and any actual solutions, all vertices are balanced modulo p.

Remark (Path-following arguments). Even though this proof is a natural generalisation of the argument by Freund and Todd [FT81], it is not a *path-following argument* for $p \geq 3$. Indeed, in the case where $p \geq 3$, it is not clear how we could explore this graph by following a path that is guaranteed to end at a solution. In fact, we provide strong evidence that it is not possible to prove $\mathbb{Z}_p$-STAR-TUCKER by a path-following argument. Since $\mathbb{Z}_p$-STAR-TUCKER is PPA-p-hard (Proposition 10.4), and since a path-following proof of $\mathbb{Z}_p$-STAR-TUCKER would presumably show that the problem lies in PPA, this would imply that PPA-$p \subseteq$ PPA, for prime $p \geq 3$. However, this is not expected to hold, as we saw in Chapter 9.

We are now ready to prove Proposition 10.5 by providing a reduction from $\mathbb{Z}_p$-STAR-TUCKER to IMBALANCE-MOD-p.

Proof of Proposition 10.5. The proof is a generalisation of the construction given by Freund and Todd [FT81] for Tucker's lemma. The main difficulties in generalising their approach are:

- For $p = 2$, when a path hits the boundary, there is a corresponding path that also hits the boundary on the antipodal side, and we can join the two endpoints. For $p > 2$, when a path hits the boundary, there are now $p - 1$ other corresponding paths that also hit the boundary. We ensure that no solution occurs there by directing all the edges. We show how to direct the edges consistently and efficiently.
- For $p = 2$, the original construction associates a label with each axis of the domain. For $p > 2$, there are more axes than labels, and so a single label must be associated to multiple axes. This creates imbalanced nodes in the graph that are not solutions. We solve this problem by carefully assigning weights to the edges of the graph.

Sub-orthants. Recall that $d = t(p-1)$. Consider the domain $R^d_{p,m}$ with a labelling function λ given by a Boolean circuit. Let T be Kuhn's triangulation of $R^d_{p,m}$ as described earlier. The domain $R^d_{p,m}$ can be subdivided into what we call sub-orthants, which are orthants of coordinate subspaces. Formally, a sub-orthant is a space of the form $A_1 \times A_2 \times \cdots \times A_d$, where $A_\ell = \{*^{i_\ell} j : 0 \le j \le m\}$ or $A_\ell = \{0\}$ for $\ell = 1, \ldots, d$.

We associate a label to every axis of $R^d_{p,m}$ as follows. The label $*^i j$ is associated to the $*^i$-axis of the $[(j-1)(p-1)+\ell]$th copy of $R_{p,m}$, for $\ell = 1, 2, \ldots, p-1$. Thus, every label is associated to exactly $p-1$ axes. For any sub-orthant X, let $S(X)$ denote the set of labels associated with the axes that are used by X. For any simplex σ of T, we let $O(\sigma)$ denote the smallest sub-orthant that contains σ.

For any sub-orthant O and $j \in [t]$, let $r_j(O)$ be the number of coordinates in the range $(j-1)(p-1)+1, \ldots, j(p-1)$ that are equal to 0 in O. In particular, we have $\sum_{j=1}^{t} r_j(O) = t(p-1) - \dim(O)$. Note that if $|S(O)| = \dim(O)$, then $r_j(O) = p - 1 - |\{i \in \mathbb{Z}_p : *^i j \in S(O)\}|$. We abuse notation and denote $r_j(\sigma) := r_j(O(\sigma))$.

Happy simplices. Let $k \in \{0, 1, \ldots, d\}$. A k-dimensional simplex σ of T is *happy*, if

1. $\dim(O(\sigma)) = k$ *(σ is full-dimensional in its sub-orthant)*
2. $|S(O(\sigma))| = \dim(O(\sigma))$ *(the sub-orthant uses ≤ 1 axis associated with each label)*
3. $S(O(\sigma)) \subseteq \lambda(\sigma)$ *(σ carries all the labels associated with its sub-orthant)*

A happy simplex is called *super-happy* if we actually have $S(O(\sigma)) \subsetneq \lambda(\sigma)$. In particular, the 0-dimensional simplex 0^d is super-happy.

Consider a happy simplex σ that has a facet $\tau \subset \partial R^d_{p,m}$ such that $\lambda(\tau) = S(O(\sigma))$. Such a simplex is called a boundary-happy simplex. If σ is such a simplex, then the simplex that has $\theta\tau$ as a facet is also boundary-happy. Thus, we group $\tau, \theta\tau, \ldots, \theta^{p-1}\tau$ together into an equivalence class $[\tau]$. Every such equivalence class has size exactly p. Formally, let B be the set of all simplices $\tau \subset \partial R^d_{p,m}$ such that $\lambda(\tau) = S(O(\tau))$ and $|S(O(\tau))| = \dim(O(\tau))$. We define an equivalence relation on B by $\tau_1 \equiv \tau_2$ if and only if $\exists i \in \mathbb{Z}_p$ such that $\tau_1 = \theta^i \tau_2$.

We construct a graph G. The vertices of G are the happy simplices of T and the equivalence classes $[\tau]$ (for all $\tau \in B$).

Orientation. We now define an orientation for our simplices. Fix an ordering of the labels, e.g., $*^1 1, *^2 1, \ldots, *^p 1, *^1 2, \ldots$. Let σ be any happy k-simplex, $k \geq 1$, and let $x_0 x_1 \ldots x_k$ be any ordering of its vertices. Let $\widehat{x}_0, \widehat{x}_1, \ldots, \widehat{x}_k \in \mathbb{N}^k$ denote the coordinates of $x_0, x_1, \ldots, x_k$ in $O(\sigma)$, where the coordinates are ordered according to the fixed ordering of the labels. Note that there is at most one coordinate associated to each label (because $|S(O(\sigma))| = k$) and thus the coordinate vectors are uniquely determined. Furthermore, note that the coordinates are non-negative. We define the $k \times k$ matrix $M = [\widehat{x}_0 - \widehat{x}_1, \widehat{x}_0 - \widehat{x}_2, \ldots, \widehat{x}_0 - \widehat{x}_k]$, i.e., the ith column is $\widehat{x}_0 - \widehat{x}_i$. Then, we define the orientation of happy simplex σ with ordering $x_0 x_1 \ldots x_k$ as $\mathrm{or}(\sigma | x_0 \ldots x_k) = \det M$. Note that by construction of the triangulation T, we always have $\det M \in \{-1, +1\}$. Indeed, it is easy to check that M can be transformed into the identity matrix by elementary operations that can only change the sign of the determinant.

Edges. Let σ be a happy k-simplex, $k \geq 1$. If σ is super-happy, then it has a single facet τ such that $\lambda(\tau) = S(O(\sigma))$. If it is not super-happy, then it has exactly two facets τ_1 and τ_2 such that $\lambda(\tau_i) = S(O(\sigma))$, $i = 1, 2$. In any case, any such facet τ of σ yields an edge as follows:

- If τ does not lie in the boundary of the sub-orthant $O(\sigma)$, then there is exactly one other k-simplex σ' in $O(\sigma)$ that also has τ as its facet. σ' is also happy, and we put an edge between σ and σ'.
- If τ lies in the boundary of the sub-orthant $O(\sigma)$, there are two cases:
  - τ lies in $\partial R^d_{p,m}$. In that case, σ is a boundary-happy simplex and we put an edge between σ and $[\tau]$.
  - τ does not lie in $\partial R^d_{p,m}$. Then, τ is a super-happy $(k-1)$-simplex and we put an edge between σ and τ.

In all of these cases, the direction of the edge is determined as follows. Let $x_0 x_1 \ldots x_k$ be the ordering of the vertices of σ such that $\tau = \{x_1, \ldots, x_k\}$ and $x_1 \ldots x_k$ are ordered according to their labels. If $\mathrm{or}(\sigma | x_0 \ldots x_k) = 1$, then the edge is incoming into σ. Otherwise, it is outgoing out of σ. Finally, the weight of the edge is always $\prod_{j=1}^{t} r_j(\sigma)!$.

By the definition, it follows that there are three types of edges:

- (Type 1) An edge between two happy k-simplices σ_1, σ_2 that lie in the same sub-orthant and share a facet τ with $\lambda(\tau) = S(O(\sigma_i))$, $i = 1, 2$.
- (Type 2) An edge between a happy simplex σ and its super-happy facet τ such that $\lambda(\tau) = S(O(\sigma))$.

- (Type 3) An edge between a boundary-happy simplex σ and its facet equivalence class $[\tau]$.

An edge of Type 2 or 3 is always "created" by exactly one of its endpoints. Thus, its direction and weight are well-defined. An edge of Type 1 however, is "created" by both of its endpoints and we will prove that it is well-defined, i.e., both endpoints agree on its direction and weight. For the weight this is easy to see, since $O(\sigma_1) = O(\sigma_2)$ implies that $r_j(\sigma_1) = r_j(\sigma_2)$ for all j. For the direction, we postpone this consistency check to the end of the proof.

We now prove that all vertices of G that do not yield a solution are balanced modulo p, except 0^d.

The trivial solution 0^d. We have $\lambda(0^d) = *^i j$. There are exactly $p-1$ sub-orthants O such that $S(O) = \{*^i j\}$ and each of them contains a happy 1-simplex σ that has 0^d as a facet. It follows that 0^d has $p-1$ edges, each with weight $((p-1)!)^t/(p-1)$. Furthermore, all the edges are outgoing, because $\mathrm{or}(\sigma|x_0 0^d) = 1$ (i.e., incoming into σ). It follows that the total imbalance of 0^d is $(p-1)((p-1)!)^t/(p-1) = ((p-1)!)^t = (-1)^t \mod p$, where we used $(p-1)! = -1 \mod p$ since p is prime (Wilson's theorem). It follows that 0^d is always a valid trivial solution for IMBALANCE-MOD-p, because $(-1)^t \neq 0 \mod p$ for all $t \geq 1$.

Happy, but not super-happy. Consider a happy k-simplex σ with two facets τ_1, τ_2 that satisfy $\lambda(\tau_i) = S(O(\sigma))$, $i = 1, 2$. Then, σ has two edges and they both have the same weight. Let $x_0 x_1 \dots x_k$ be the ordering of σ such that $\tau_1 = \{x_1, \dots, x_k\}$ and $x_1 \dots x_k$ are ordered according to their labels. Let $i \in [k]$ be the index such that x_i and x_0 have the same label. In particular, $\tau_2 = \{x_1, \dots, x_{i-1}, x_0, x_{i+1}, \dots, x_k\}$ and $x_1 \dots x_{i-1} x_0 x_{i+1} \dots x_k$ are ordered according to their labels. We have

$$\begin{aligned}
&\det[\widehat{x}_0 - \widehat{x}_1, \widehat{x}_0 - \widehat{x}_2, \dots, \widehat{x}_0 - \widehat{x}_k] \\
=&\det[\widehat{x}_i - \widehat{x}_1, \dots, \widehat{x}_i - \widehat{x}_{i-1}, \widehat{x}_0 - \widehat{x}_i, \widehat{x}_i - \widehat{x}_{i+1}, \dots, \widehat{x}_i - \widehat{x}_k] \\
=&-\det[\widehat{x}_i - \widehat{x}_1, \dots, \widehat{x}_i - \widehat{x}_{i-1}, \widehat{x}_i - \widehat{x}_0, \widehat{x}_i - \widehat{x}_{i+1}, \dots, \widehat{x}_i - \widehat{x}_k]
\end{aligned}$$

where we first subtracted the ith column from all other columns, and then multiplied the ith column by -1. It follows that $\mathrm{or}(\sigma|x_0 \dots x_k) = -\mathrm{or}(\sigma|x_i x_1 \dots x_{i-1} x_0 x_{i+1} \dots x_k)$. Thus, one edge is incoming and the other outgoing, i.e., σ is balanced.

Equivalence class. Consider an equivalence class $[\tau]$. Let σ be the happy k-simplex that has τ as a facet. In G, $[\tau]$ has exactly p edges: one with each of $\sigma, \theta\sigma, \theta^2\sigma, \ldots, \theta^{p-1}\sigma$. Since $S(O(\theta^i\sigma)) = \theta^i S(O(\sigma))$ for all i, it follows that $r_j(\theta^i\sigma) = r_j(\sigma)$ for all i, j. Thus, all p edges have the same weight. Let $x_0 \ldots x_k$ be the ordering of σ such that $\tau = \{x_1, \ldots, x_k\}$ and $x_1 \ldots x_k$ are ordered according to their labels. Let $y_i = \theta x_i$ for all i. Then, $y_1 \ldots y_k$ might not be ordered according to their labels. We let π denote the permutation that we would have to apply to order them correctly. As before, $\widehat{x}_i$ denotes the coordinates of x_i restricted to $O(\sigma)$, where the coordinates are ordered according to the associated label. $\widehat{y}_i$ denotes the coordinates of y_i restricted to $O(\theta\sigma)$, where the coordinates are ordered according to the associated label. Since the associated labels have changed according to θ, it follows that if we re-order the coordinates of $\widehat{x}_i$ according to π we obtain $\widehat{y}_i$ for all $i = 0, 1, \ldots, k$. Thus, we have

$$\begin{aligned}
\mathrm{or}(\theta\sigma | y_0\pi(y_1 \ldots y_k)) &= \mathrm{sgn}(\pi) \det[\widehat{y}_0 - \widehat{y}_1, \widehat{y}_0 - \widehat{y}_2, \ldots, \widehat{y}_0 - \widehat{y}_k] \\
&= \mathrm{sgn}(\pi)^2 [\widehat{x}_0 - \widehat{x}_1, \widehat{x}_0 - \widehat{x}_2, \ldots, \widehat{x}_0 - \widehat{x}_k] \\
&= \mathrm{or}(\sigma | x_0 \ldots x_k)
\end{aligned}$$

It follows that all edges of $[\tau]$ are directed the same way, i.e., they are all incoming or all outgoing. Since there are p edges and they also have the same weight, it follows that $[\tau]$ has imbalance 0 modulo p. In this argument we assumed that λ satisfies the boundary conditions. Thus, if $[\tau]$ is not balanced modulo p, we obtain a counter-example, which is a solution.

Super-happy. Consider a super-happy k-simplex σ, $k \geq 1$. Note that σ has a single facet τ that satisfies $\lambda(\tau) = S(O(\sigma))$. Thus, σ "creates" a single edge. Let $x_0 \ldots x_k$ be the ordering of σ such that $\tau = \{x_1, \ldots, x_k\}$ and $x_1 \ldots x_k$ are ordered according to their labels. The edge has weight $\prod_{j=1}^{t} r_j(\sigma)!$ and it is incoming if $\mathrm{or}(\sigma | x_0 \ldots x_k) = 1$, outgoing otherwise.

Since σ is super-happy, we have $\lambda(x_0) \notin \lambda(\tau) = S(O(\sigma))$. Let $*^i\ell = \lambda(x_0)$. If $\{*^j\ell | j \in [p] \setminus \{i\}\} \subseteq S(O(\sigma))$, or equivalently if $r_\ell(\sigma) = 0$, then σ yields a solution. Otherwise, there are exactly $r_\ell(\sigma)$ different sub-orthants O such that $O(\sigma) \subset O$ and $S(O) = S(O(\sigma)) \cup \{*^i\ell\}$. Thus, there are exactly $r_\ell(\sigma)$ happy $(k+1)$-simplices ρ such that σ is a facet of ρ and ρ is happy because of σ (i.e., $S(O(\rho)) = \lambda(\sigma)$). It follows that σ has $r_\ell(\sigma)$ additional edges (apart from the one it "created"). Each of these edges has weight $\prod_{j=1}^{t} r_j(\rho)! = (r_\ell(\sigma))^{-1} \prod_{j=1}^{t} r_j(\sigma)!$, since $r_\ell(\rho) = r_\ell(\sigma) - 1$. Thus, if these $r_\ell(\sigma)$ edges have opposite direction to the edge "created" by σ (from the perspective of σ), σ will be balanced.

Consider any such ρ. Let $x_0 \ldots x_{k+1}$ be the ordering of ρ such that $\sigma = \{x_1, \ldots, x_{k+1}\}$ and $x_1 \ldots x_{k+1}$ are ordered according to their labels. Let $t \in [k+1]$ be

the index of label $*^i\ell$ if we order the labels in $S(O(\rho))$. Then, we also have that $\tau = \sigma \setminus \{x_t\}$. Furthermore, $x_1 \dots x_{t-1}x_{t+1} \dots x_{k+1}$ are also ordered according to their labels. From the perspective of σ, the edge it "created" is directed according to $\mathrm{or}(\sigma|x_t x_1 \dots x_{t-1}x_{t+1} \dots x_{k+1})$ and the edge created by ρ is directed according to $-\mathrm{or}(\rho|x_0 x_1 \dots x_{k+1})$. As before, let $\widehat{x}_j$ denote the coordinates of x_j restricted to $O(\rho)$, where the coordinates are ordered according to the associated label. Let $\bar{x}_j$ denote the coordinates of x_j restricted to $O(\sigma)$, where the coordinates are ordered according to the associated label. Note that if we remove the tth coordinate from $\widehat{x}_j$, then we obtain $\bar{x}_j$. We now have

$$\begin{aligned}
\mathrm{or}(\rho|x_0 x_1 \dots x_{k+1}) &= \det[\widehat{x}_0 - \widehat{x}_1, \dots, \widehat{x}_0 - \widehat{x}_{k+1}] \\
&= \det[\widehat{x}_t - \widehat{x}_1, \dots, \widehat{x}_t - \widehat{x}_{t-1}, \widehat{x}_0 - \widehat{x}_t, \widehat{x}_t - \widehat{x}_{t+1}, \dots, \widehat{x}_t - \widehat{x}_{k+1}] \\
&= (-1)^{t+t} \det[\bar{x}_t - \bar{x}_1, \dots, \bar{x}_t - \bar{x}_{t-1}, \bar{x}_t - \bar{x}_{t+1}, \dots, \bar{x}_t - \bar{x}_{k+1}] \\
&= \mathrm{or}(\sigma|x_t x_1 \dots x_{t-1}x_{t+1} \dots x_{k+1})
\end{aligned}$$

where we first subtracted the tth column from all other columns, and then we used Laplace's determinant formula along the tth row. Note that the tth entry in $\widehat{x}_t - \widehat{x}_j$ is 0 for all $j \in [k+1]$ and it is 1 in $\widehat{x}_0 - \widehat{x}_t$.

Consistency. The only thing that remains to be checked is that edges of Type 1 are well-defined, in terms of the direction. Let σ_1, σ_2 be two happy k-simplices that lie in the same sub-orthant and share a facet τ with $\lambda(\tau) = S(O(\sigma_i))$, $i = 1, 2$. Let $\{x_1, \dots, x_k\} = \tau$ and $x_1 \dots x_k$ be the ordering according to their labels. Let $\{x_0\} = \sigma_1 \setminus \tau$ and $\{x'_0\} = \sigma_2 \setminus \tau$. We want to show that $\mathrm{or}(\sigma_1|x_0 x_1 \dots x_k)$ and $\mathrm{or}(\sigma_2|x'_0 x_1 \dots x_k)$ have opposite signs. This can be proved directly combinatorially by using the way the triangulation is constructed, but we provide a proof that is more general here. Let $\phi : \mathbb{R}^k \to \mathbb{R}^k$ be the unique linear function such that $\phi(\widehat{x}_0 - \widehat{x}_1) = \widehat{x}'_0 - \widehat{x}_1$ and $\phi(\widehat{x}_1 - \widehat{x}_i) = \widehat{x}_1 - \widehat{x}_i$ for all $i \in [k] \setminus \{1\}$. ϕ is unique, because $\widehat{x}_0 - \widehat{x}_1, \widehat{x}_1 - \widehat{x}_2, \dots, \widehat{x}_1 - \widehat{x}_k$ form a basis of $\mathbb{R}^k$. ϕ is the identity function on the hyperplane given by $\widehat{x}_1 - \widehat{x}_2, \dots, \widehat{x}_1 - \widehat{x}_k$ and maps $\widehat{x}_0 - \widehat{x}_1$ to $\widehat{x}'_0 - \widehat{x}_1$. Since x_0 and x'_0 lie on opposite sides of the hyperplane defined by τ, it follows that $\widehat{x}_0 - \widehat{x}_1$ and $\widehat{x}'_0 - \widehat{x}_1$ lie on opposite sides of the hyperplane on which ϕ is the identity. It follows that $\det \phi < 0$. Thus, we can write

$$\begin{aligned}
\det[\widehat{x}'_0 - \widehat{x}_1, \dots, \widehat{x}'_0 - \widehat{x}_k] &= \det[\widehat{x}'_0 - \widehat{x}_1, \widehat{x}_1 - \widehat{x}_2 \dots, \widehat{x}_1 - \widehat{x}_k] \\
&= \det \phi \det[\widehat{x}_0 - \widehat{x}_1, \widehat{x}_1 - \widehat{x}_2 \dots, \widehat{x}_1 - \widehat{x}_k] \\
&= \det \phi \det[\widehat{x}_0 - \widehat{x}_1, \dots, \widehat{x}_0 - \widehat{x}_k]
\end{aligned}$$

and the statement follows. □

10.5 Conclusion and Future Directions

Our topological characterisation of PPA-p can possibly enable us to obtain similar membership or hardness results for other interesting problems. For example, are the problems whose totality is established via the BSS Theorem, like the Chromatic Number of Kneser Hypergraphs[3] studied in [AFL86] in PPA-p? Are they PPA-p-complete? We believe that due to its simplicity, our p-POLYGON-TUCKER problem can be a very useful tool for obtaining hardness results for these problems. What about other problems that generalise problems that are known to be in PPA or are even PPA-complete? For example, Filos-Ratsikas and Goldberg [FG19] showed that the discrete Ham-Sandwich problem is also PPA-complete. Is there a generalisation of the problem that could be complete for PPA-p? A computational version of the Center Transversal Theorem [Dol92; ZV90] might be a good candidate. Another interesting open problem is to investigate the connection of the general statement of Dold's Theorem [Dol83] from algebraic topology with the subclasses of TFNP.

[3]The computational version of this problem would be of the form: given a colouring (as a circuit) that cannot possibly be correct everywhere, because it does not use enough colours, find any edge where it makes a mistake.

Chapter 11

Necklace-Splitting and Consensus-Division

In this chapter, we apply the topological tools developed in the previous chapter to study two (closely related) problems from combinatorics and fair division: the Necklace-Splitting and Consensus-Division problems. As our main result, we prove that these problems lie in PPA-k under Turing reductions.

11.1 Background and Definitions

Necklace Splitting. The Necklace Splitting problem is a classical problem in combinatorics, dating back to the mid-1980s and the works of Goldberg and West [GW85], Alon and West [AW86] and Alon [Alo87], among others. In this problem, k thieves are aiming to split an open necklace containing n beads of t different colours (with exactly $k \cdot a_i$ beads of colour i, for some $a_i \in \mathbb{N}$), such that each thief receives exactly a_i beads of colour i. Furthermore, the thieves are allowed to use only $(k-1)t$ cuts to obtain this division.

A solution to the Necklace Splitting problem is guaranteed to exist, as was proven by Alon [Alo87]. Earlier on, Goldberg and West [GW85] and Alon and West [AW86] had proven the existence of a solution for the case of 2 thieves using the Borsuk-Ulam Theorem (Section 2.3.2). Alon's proof for the general case proceeds in two steps. First, using a simple argument, he proves that if the theorem holds for any prime number p of thieves, it also holds for any other number of thieves. In the second step, which is significantly more involved, he proves the theorem for any prime number by using the BSS Theorem [BSS81], which we studied in Chapter 10.

Questions about the computational complexity of finding a solution were raised explicitly as early as when the first existence results were proven [GW85] and then later on by a series of papers [Alo88; Alo90; Meu08; MS09; MN12]. Formally, the computational problem with k thieves is defined as follows [Pap94; FG19]:

Definition 11.1. k-NECKLACE-SPLITTING:

Input: An open necklace with n beads, each of which has one of t colours, and where there are exactly $a_i k$ beads of colour $i = 1, \ldots, t$, where $a_i \in \mathbb{N}$.
Output: A partitioning of the necklace into k (not necessarily connected) pieces such that each piece contains exactly a_i beads of colour i, using at most $(k-1)t$ cuts.

The first definitive answer was provided by Filos-Ratsikas and Goldberg [FG19], via their PPA-completeness result, following an initial PPAD-hardness result by Filos-Ratsikas et al. [FFGZ18]. Crucially however, their result only applies to the case of two thieves,[1] i.e., to 2-NECKLACE-SPLITTING. In fact, Filos-Ratsikas and Goldberg observed (also referencing Longueville and Živaljević [LŽ06] and Meunier [Meu14] as previously having made similar observations) that the version of the problem with $p \geq 3$ thieves does not seem to boil down to the principle associated with the class PPA. To this end, they conjectured that p-NECKLACE-SPLITTING is complete for the computational class PPA-p.

Consensus-Division. Alon [Alo87] proved the Necklace Splitting theorem by using the BSS Theorem to show existence for a continuous version of Necklace Splitting and then "rounding" a solution of the continuous problem to obtain a splitting of the necklace. When investigating the complexity of k-NECKLACE-SPLITTING, it is very convenient to consider this more general continuous version, which is also an interesting fair division problem by itself. Even though the continuous theorem is termed as a "generalised Hobby-Rice theorem" by Alon [Alo87], we instead use the term *Consensus-1/k-Division* proposed by Simmons and Su [SS03]. The corresponding computational problem is defined as follows [FFGZ18; FG18]:

Definition 11.2. CONSENSUS-$1/k$-DIVISION:

Input: $\varepsilon > 0$ and continuous probability measures $\mu_1, \ldots, \mu_t$ on $[0, 1]$. The probability measures are given by their density functions on $[0, 1]$, which are step functions (explicitly given in the input).
Output: A partitioning of the unit interval into k (not necessarily connected) pieces $A_1, \ldots, A_k$ using at most $(k-1)t$ cuts, such that $|\mu_j(A_i) - \mu_j(A_\ell)| \leq \varepsilon$ for all i, j, ℓ.

Note that we can equivalently ask for $\mu_j(A_i) = 1/k \pm \varepsilon$ for all i, j. Indeed, the computational problems are equivalent.

Alon's rounding procedure yields a reduction from k-NECKLACE-SPLITTING to CONSENSUS-$1/k$-DIVISION with $\varepsilon = 0$. Filos-Ratsikas and Goldberg [FG18] extended this result by showing that k-NECKLACE-SPLITTING reduces to CONSENSUS-$1/k$-DIVISION (with $\varepsilon > 0$).

[1] Filos-Ratsikas and Goldberg [FG19] also extend the PPA-membership straightforwardly to numbers of thieves which are powers of 2.

Proposition 11.1 (Alon [Alo87] and Filos-Ratsikas and Goldberg [FG18])**.** *For any $k \geq 2$, k-*Necklace-Splitting *reduces to* Consensus-$1/k$-Division*.*

In fact, Filos-Ratsikas and Goldberg [FG18] show that k-Necklace-Splitting is *equivalent* to Consensus-$1/k$-Division with inverse-polynomial ε. Filos-Ratsikas and Goldberg [FG19] then prove that 2-Necklace-Splitting is PPA-hard by proving that Consensus-1/2-Division (usually called Consensus-Halving) with inverse-polynomial ε is PPA-hard.

The PPA-membership of Consensus-1/2-Division is proved by reducing to a computational version of Tucker's Lemma (Section 2.3.2), a discrete analogue of the Borsuk-Ulam theorem [FFGZ18]. This reduction is based on a proof of existence for 2-thief Necklace Splitting that uses Tucker's Lemma [SS03]. Since the proof of existence for Consensus-$1/p$-Division given by Alon [Alo87] relies on the BSS theorem [BSS81], one would expect that the membership of p-BSS-Tucker in PPA-p (which we proved in Chapter 10) would suffice to prove that Consensus-$1/p$-Division lies in PPA-p. However, it turns out that there are other steps in the proof of Alon [Alo87] for which it is unclear whether they can be carried out in polynomial time. As a result, we prove the membership in PPA-p by providing a reduction to our new problem $\mathbb{Z}_p$-star-Tucker instead. Our reduction also yields a new combinatorial proof of existence for Necklace Splitting and Consensus-Division, which is conceptually simpler than the only other such proof by Meunier [Meu14].

11.2 Reduction to $\mathbb{Z}_p$-star-Tucker

In this section, we present the main technical result of this chapter.

Theorem 11.1. *For any prime p,* Consensus-$1/p$-Division *reduces to* $\mathbb{Z}_p$-star-Tucker*.*

This reduction has various consequences that are explored in the next section, including membership of Consensus-$1/p$-Division and p-Necklace-Splitting in PPA-p.

We now proceed to the proof Theorem 11.1. In fact, we provide an even stronger result: we show that a much more general version of Consensus-$1/p$-Division reduces to $\mathbb{Z}_p$-star-Tucker. Namely, we consider any polynomially computable and polynomially continuous (Section 2.2.3) class $\mathcal{F}$ of cumulative distribution functions on $[0, 1]$. For any $f \in \mathcal{F}$ and any $a \in [0, 1]$, $f(a)$ is the probability of the interval $[0, a]$ according to f. In particular, note that the class of all cumulative distribution functions given by step function densities (represented explicitly) is polynomially computable and polynomially continuous.

Definition 11.3. Let $k \geq 2$ and let $\mathcal{F}$ be a polynomially computable and polynomially continuous class of cumulative distribution functions on $[0,1]$. The problem Consensus-$1/k$-Division$[\mathcal{F}]$ is defined exactly as Consensus-$1/k$-Division, except that the probability measures are given by cumulative distribution functions in $\mathcal{F}$.

Notice that Consensus-$1/k$-Division corresponds to the special case where $\mathcal{F}$ is the class of all cumulative distribution functions given by step function densities (represented explicitly). Thus, the following is a stronger version of Theorem 11.1.

Theorem 11.2. *Let p be prime and $\mathcal{F}$ be a polynomially computable and polynomially continuous class of cumulative distribution functions on $[0,1]$. Then* Consensus-$1/p$-Division$[\mathcal{F}]$ *reduces to* $\mathbb{Z}_p$-star-Tucker.

Proof. Let $\varepsilon > 0$ and $\mu_1, \ldots, \mu_t$ be probability measures on $[0,1]$ given by functions in $\mathcal{F}$. We consider the domain $D = R^d_{p,m}$, where $d = t(p-1)$ and $m \geq 1$ will be set later. A point in D represents a way to partition $[0,1]$ into p (not necessarily connected) pieces using at most $t(p-1)$ cuts. This is a slight modification of the domain that was used by Meunier [Meu14] to encode a splitting of a necklace. Intuitively it can be explained as follows. Let $x = (*^{i_1} j_1, \ldots, *^{i_d} j_d) \in D$. We interpret each element $i \in \mathbb{Z}_p$ as a different colour. Then:

1. Paint the whole interval $[0,1]$ with the colour 1.
2. For $\ell = 1, 2, \ldots, d$: paint $[0, j_\ell/m]$ with the colour i_ℓ

Note that applying a fresh coat of paint on a previously painted part of the interval covers up the old paint. The way the interval $[0,1]$ is coloured at the end of this procedure gives us the partition encoded by $x \in D$. An important advantage of this encoding is that it is sufficiently continuous in a certain sense. Indeed, small changes in the coordinates of x have a small effect on the corresponding partition. Other simpler encoding schemes do not have this property.

Formally, this encoding can be described as follows. Add a "fake" 0th coordinate $*^{i_0} j_0 = *^1 m$. Place cuts at all positions $j_0/m, j_1/m, \ldots, j_d/m$. This subdivides the interval $[0,1]$ into at most $d+1 = t(p-1)+1$ subintervals. Then, allocate the subinterval $[a,b]$ to $i_{\widehat{\ell}} \in \mathbb{Z}_p$, where $\widehat{\ell} = \max\{0 \leq \ell \leq d : j_\ell/m \geq b\}$.

This encoding also behaves nicely with respect to θ. For any $x \in \partial D$, θx encodes the same partition as x, except that i has been replaced by $i+1$, for all $i \in \mathbb{Z}_p$. This is easy to see since for any $x \in \partial D$, there exists $\ell \geq 1$ such that $j_\ell = m$ and thus the "fake" coordinate $*^{i_0} j_0$ does not play any role.

We are now ready to define the labelling $\lambda : D \to R_{p,t} \setminus \{0\}$. This labelling is a natural generalisation of the one used by Simmons and Su [SS03]. Given $x \in D$, construct the partition it encodes, namely $A_1(x), \ldots, A_p(x)$. Then, for all $i \in \mathbb{Z}_p$ and $j \in [t]$, let $\mu_{j,i}(x) = \mu_j(A_i(x))$, i.e., the total measure of type j that is allocated to i. Finally, set $\lambda(x) = *^i j$, where i, j are determined as follows:

1. Pick $j \in [t]$ that maximises $\max_{i_1,i_2} |\mu_{j,i_1}(x) - \mu_{j,i_2}(x)|$. Break ties by picking the minimum such j.

2. Then, pick $i \in \mathbb{Z}_p$ that maximises $\mu_{j,i}(x)$. If there are multiple i's that maximise this, break ties by picking the one such that $\min A_i(x)$ is minimal (i.e., such that $A_i(x)$ contains the point closest to the left end of the unit interval).

By using the observation above about the behaviour of θ on ∂D, it is easy to see that $\lambda(\theta x) = \theta\lambda(x)$ for all $x \in \partial D$. Thus, λ is a valid instance of $\mathbb{Z}_p$-star-Tucker and we obtain a solution $x_1, \ldots, x_p \in D$ and $\widehat{\jmath} \in [t]$, such that $\text{dist}_\infty(x_i, x_k) \leq 1$ and $\lambda(x_i) = *^i\widehat{\jmath}$ for all $i, k \in [p]$. It remains to show that by picking m large enough, we obtain a solution to Consensus-$1/p$-Division$[\mathcal{F}]$.

Let $\varepsilon' = \frac{\varepsilon}{2t(p-1)}$. Since $\mathcal{F}$ is polynomially continuous, we can pick m large enough so that the value $\mu_j([0, a])$ changes by at most ε', if a moves by $1/m$. Note that m has representation length polynomial in the size of the instance. If a single coordinate of x changes by 1, $\mu_{j,i}(x)$ changes by at most ε' for all i, j. Since there are $d = t(p-1)$ coordinates, it follows that $|\mu_{j,i}(x_k) - \mu_{j,i}(x_\ell)| \leq \varepsilon' t(p-1) = \varepsilon/2$ for all i, j and for all $k, \ell \in [p]$.

Let $x := x_1$. By construction of the labelling, we obtain that for all j, i, ℓ

$$|\mu_{j,i}(x) - \mu_{j,\ell}(x)| \leq \max_{i_1,i_2} |\mu_{\widehat{\jmath},i_1}(x) - \mu_{\widehat{\jmath},i_2}(x)| \leq \varepsilon$$

The first inequality holds because $\lambda(x) = *^1\widehat{\jmath}$. The second inequality holds because if we instead had $\mu_{\widehat{\jmath},i_1}(x) > \mu_{\widehat{\jmath},i_2}(x) + \varepsilon$ for some i_1, i_2, then it would follow that $\mu_{\widehat{\jmath},i_1}(x_{i_2}) > \mu_{\widehat{\jmath},i_2}(x_{i_2})$, contradicting $\lambda(x_{i_2}) = *^{i_2}\widehat{\jmath}$. Thus, x corresponds to an ε-approximate solution. □

11.3 Consequences and Future Directions

In this section, we present some computational and mathematical consequences of Theorem 11.1, i.e., the reduction from Consensus-$1/p$-Division to $\mathbb{Z}_p$-star-Tucker. We then finish by mentioning some interesting open questions.

11.3.1 Consequences: Computational Complexity

Theorem 11.3. *For any $k \geq 2$, k-Necklace-Splitting and Consensus-$1/k$-Division lie in PPA-k under Turing reductions. In particular, if $k = p^r$ where p is prime and $r \geq 1$, then the problems lie in PPA-p (under many-one reductions).*

Recall that when k is not a prime power, there is some evidence indicating that PPA-k is unlikely to be closed under Turing reductions (Section 9.5).

Proof. Since $\mathbb{Z}_p$-STAR-TUCKER lies in PPA-p (Theorem 10.11), Theorem 11.1 immediately implies that for any prime p, CONSENSUS-1/p-DIVISION lies in PPA-p. As noted by Alon [Alo87, Proposition 3.2], for any $k, \ell \geq 2$, a Consensus-1/$(k\ell)$-Division can be obtained by first finding a Consensus-1/k-Division – which divides the interval into k (not necessarily connected) pieces – and then finding a Consensus-1/ℓ-Division of each of the k pieces. Note that we obtain (at most) the desired number of cuts. It is easy to check that this approach also works if we consider approximate instead of exact solutions. Thus, we can solve an instance of CONSENSUS-1/$(k\ell)$-DIVISION by first solving an instance of CONSENSUS-1/k-DIVISION, and then k instances of CONSENSUS-1/ℓ-DIVISION.

In particular, for any prime p and any $r \geq 1$, CONSENSUS-1/p^r-DIVISION can be solved by solving $1 + p + p^2 + \cdots + p^{r-1}$ instances of CONSENSUS-1/p-DIVISION. Thus, we obtain a Turing reduction from CONSENSUS-1/p^r-DIVISION to CONSENSUS-1/p-DIVISION. Since CONSENSUS-1/p-DIVISION lies in PPA-p and PPA-p is closed under Turing reductions (Theorem 9.6), it follows that CONSENSUS-1/p^r-DIVISION lies in PPA-p.

Now consider any $k = \prod_{i=1}^{m} p_i^{r_i}$, where $m \geq 1$, $r_i \geq 1$ and the p_i are distinct primes. Then, CONSENSUS-1/k-DIVISION can be solved by a query to CONSENSUS-1/$p_1^{r_1}$-DIVISION, then $p_1^{r_1}$ queries to CONSENSUS-1/$p_2^{r_2}$-DIVISION (which can be turned into a single query to PPA-p_2), then $p_1^{r_1} p_2^{r_2}$ queries to CONSENSUS-1/$p_3^{r_3}$-DIVISION (which can also be turned into a single query to PPA-p_3), etc. Thus, CONSENSUS-1/k-DIVISION can be solved by a query to PPA-p_1, then a query to PPA-p_2, then one to PPA-p_3, ..., and finally a query to PPA-p_m. Since PPA-$p_i \subseteq$ PPA-k for $i = 1, \ldots, m$ (Corollary 9.2), it follows that there is a Turing reduction from CONSENSUS-1/k-DIVISION to a PPA-k-complete problem (e.g., IMBALANCE-MOD-k).

Since k-NECKLACE-SPLITTING reduces to CONSENSUS-1/k-DIVISION (Proposition 11.1), the results also hold for k-NECKLACE-SPLITTING. □

11.3.2 Consequences: Mathematical Existence

Together, the proofs of Proposition 10.5 and Theorem 11.1 yield a reduction from CONSENSUS-1/p-DIVISION to IMBALANCE-MOD-p. Since every instance of IMBALANCE-MOD-p has a solution (and the proof of this is trivial), we obtain a proof that CONSENSUS-1/p-DIVISION always has a solution. Thus, this proves that if the probability measures are step functions (described by rational numbers), there always exists a Consensus-1/p-Division. While we have given the proof in terms of a reduction (since this is required for our complexity results), it can also be written as a mathematical proof of existence (without any computational considerations).

Once existence of a Consensus-$1/p$-Division for step functions has been proved, a constructive argument by Alon [Alo87, Section 2] also gives existence for Necklace-Splitting with p thieves. Putting everything together, the proof of Necklace-Splitting thus obtained is a fully combinatorial proof that does not use any advanced machinery and is easier to follow than existing proofs. Indeed, the original proof by Alon [Alo87] uses the BSS theorem of Bárány, Shlosman and Szücs [BSS81] as a black box. The only other fully combinatorial proof by Meunier [Meu14], while quite elegant, is significantly more involved.

Going back to Consensus-$1/p$-Division, the proof we obtain (which uses $\mathbb{Z}_p$-star Tucker's lemma) actually works for any probability measures, not only step functions. Moreover, similarly to Simmons and Su [SS03], our proof does not make use of the fact that the measures are additive and non-negative. Thus, we obtain a stronger version of the Consensus-$1/p$-Division theorem given by Alon [Alo87, Theorem 1.2] (which he calls a generalisation of the Hobby-Rice theorem). Let $\mathcal{B}([0,1])$ denote the Borel σ-algebra on the unit interval and let λ denote the Lebesgue measure on the unit interval. Finally, let $\triangle$ denote the symmetric difference, i.e., $A \triangle B = (A \setminus B) \cup (B \setminus A)$.

Theorem 11.4. *Let p be any prime and $t \geq 1$. Let $v_1, \ldots, v_t : \mathcal{B}([0,1]) \to \mathbb{R}$ be such that, for all $1 \leq j \leq t$, v_j satisfies the following continuity condition: for all $\varepsilon > 0$ there exists $\delta > 0$ such that*

$$|v_j(A) - v_j(B)| \leq \varepsilon \quad \text{for all } A, B \in \mathcal{B}([0,1]) \text{ that satisfy } \lambda(A \triangle B) \leq \delta.$$

Then, there exists a Consensus-$1/p$-Division. Namely, it is possible to partition the unit interval into p (not necessarily connected) pieces $A_1, \ldots, A_p$ using at most $(p-1)t$ cuts, such that $v_j(A_i) = v_j(A_\ell)$ for all $1 \leq i, \ell \leq p$, $1 \leq j \leq t$.

Before we move on to the proof, let us briefly explain why we only obtain the result for prime p. In the usual setting where the valuations are probability measures, it is enough to prove the statement for primes. Indeed, using the standard argument by Alon [Alo87, Proposition 3.2], if a Consensus-$1/k$-Division and a Consensus-$1/\ell$-Division always exist, then a Consensus-$1/(k\ell)$-Division exists. But Alon's argument makes use of the additivity property of the measures. Indeed, consider the non-additive setting and say that we are trying to show that a Consensus-1/4-Division exists. We know that a Consensus-1/2-Division exists and this yields a partition of $[0,1]$ into A_1 and A_2. We have that $v_j(A_1) = v_j(A_2)$ for all j. Following Alon's argument, we then find a Consensus-1/2-Division of A_1 and of A_2. This yields $A_{11} \cup A_{12} = A_1$ and $A_{21} \cup A_{22} = A_2$ such that $v_j(A_{11}) = v_j(A_{12})$ and $v_j(A_{21}) = v_j(A_{22})$ for all j. However, we might not have $v_j(A_{11}) = v_j(A_{22})$, since we no longer have $v_j(A_{11}) + v_j(A_{12}) = v_j(A_{21}) + v_j(A_{22})$.

If we assume that the valuations are additive (even just finite additivity), but still allow them to take negative values, then Alon's argument works as before and thus the result again holds for any $k \geq 2$.

Proof. Using $\mathbb{Z}_p$-star Tucker's lemma (Theorem 10.10) it follows that for any $\varepsilon > 0$, there exists an ε-approximate Consensus-$1/p$-Division, i.e., we can partition the interval into p pieces $A_1, \ldots, A_p$ (using at most $(p-1)t$ cuts) such that $|v_j(A_i) - v_j(A_\ell)| \leq \varepsilon$ for all $1 \leq i, \ell \leq p$, $1 \leq j \leq t$. Indeed, it suffices to follow the same steps as in the proof of Theorem 11.2.

Let R_p denote the continuous version of $R_{p,m}$. R_p consists of p copies of the segment $[0,1]$ (namely $*^1[0,1], \ldots, *^p[0,1]$) that share the same origin (i.e., $*^1 0 = \cdots = *^p 0$). Using the interpretation given in the proof of Theorem 11.2, every point in R_p^d corresponds to a way to partition $[0,1]$ into p (not necessarily connected) pieces using at most $(p-1)t$ cuts. Since $(R_p^d, \mathrm{dist}_\infty)$ is a compact metric space, every sequence must have a subsequence that converges. Thus, a sequence $(x_n)_n$, where $x_n \in R_p^d$ is a $1/2^n$-approximate Consensus-$1/p$-Division, must have a subsequence that converges to some $x \in R_p^d$. For any i, j the function $f : R_p^d \to \mathbb{R}$, $y \mapsto v_j(A_i(y))$ is continuous. It follows that x must correspond to an exact Consensus-$1/p$-Division. □

11.3.3 Future Directions

The most interesting direction is of course to resolve the complexity of k-Necklace-Splitting and Consensus-$1/k$-Division by proving hardness results matching our membership results. When $k = p^r$ is a prime power, then PPA-k = PPA-p, and it seems reasonable to conjecture that k-Necklace-Splitting and Consensus-$1/k$-Division are indeed PPA-p-complete. However, it is unclear whether we should expect k-Necklace-Splitting and Consensus-$1/k$-Division to be PPA-k-complete when k is not a prime power. Indeed, the class PPA-k seems to exhibit some "unnatural" characteristics when k is not a prime power, such as (most likely) not being closed under Turing reductions.

No hardness result is currently known for k-Necklace-Splitting for any $k > 2$. Thus, even a PPAD-hardness result for any $k > 2$ would already be very interesting. A first step in this direction was made by Filos-Ratsikas et al. [FHSZ20] who recently proved that Consensus-1/3-Division is PPAD-hard for inverse-exponential approximation parameter ε. Improving this hardness result to also hold for inverse-polynomial ε would imply that 3-Necklace-Splitting is PPAD-hard. Even just extending the result of Filos-Ratsikas et al. [FHSZ20] to higher values of k would also be very interesting, since no hardness result is known for $k > 3$.

Filos-Ratsikas et al. [FHSZ20] have also provided a significantly simpler proof (and strengthening) of the PPA-hardness of 2-Necklace-Splitting and Consensus-1/2-Division [FG19]. However, it remains open to prove a PPA-hardness result for Consensus-1/2-Division that holds even for constant ε. Currently, only PPAD-hardness is known for this setting [FFGZ18].

Consensus-1/2-Division (usually called Consensus-Halving) has also been studied in the setting where the number of agents is constant and the valuations of the agents are more general and not necessarily additive [DFH21]. The problem of finding an exact solution with standard additive measures has also been studied. Deligkas et al. [DFMS21] proved that the problem is FIXP-hard. Batziou, Hansen and Høgh [BHH21] showed that the strong approximation problem (with measures represented by algebraic circuits) is complete for a class corresponding to the Borsuk-Ulam theorem.

Finally, it would be interesting to obtain some positive results for these problems, either for restricted settings or in terms of approximation algorithms, and also by allowing more cuts. Some results in this direction have been provided by Alon and Graur [AG20] and Filos-Ratsikas et al. [FHSZ20]. The current hardness results hold even if one allows $n + n^{1-\delta}$ cuts (instead of just n) for any constant $\delta > 0$ [FHSZ20], and it would also be interesting to see if the problems remain hard with even more cuts.

Chapter 12

List of Selected Open Problems

In this chapter, for the convenience of the reader, we collect some of the main open problems that emerge from the work presented in this thesis.

1. Does the Hairy Ball problem reduce to Brouwer's fixed point theorem, if one considers exact solutions? In other words, is the exact Hairy Ball problem FIXP-complete?

2. Is the equilibrium computation problem for the First Price auction with *common priors* PPAD-complete? Or is it easier?

3. Is computing a mixed Nash equilibrium of a network congestion game PPAD $\cap$ PLS-complete?

4. Does the problem of computing a KKT point of a polynomial remain PPAD$\cap$PLS-complete for degree-3 or even degree-2 polynomials?

5. Is the problem of computing a Tarski fixed point PPAD $\cap$ PLS-complete?

6. Can we obtain an oracle separation between EOPL and PPAD $\cap$ PLS?

7. Does Factoring lie in PPA-p for primes $p > 2$ (even under randomised reductions)?

8. Is p-Necklace-Splitting PPA-p-complete for primes $p > 2$?

Appendices

Appendix A

Proofs of Arithmetic Circuit Properties

A.1 Efficient evaluation of well-behaved arithmetic circuits (Proof of Lemma 2.1)

We restate the Lemma here for convenience.

Lemma 2.1. *Let f be a well-behaved arithmetic circuit with n inputs. Then, for any rational $x \in \mathbb{R}^n$, $f(x)$ can be computed in time* $\text{poly}(\text{size}(f), \text{size}(x))$.

Proof. Recall that an arithmetic circuit f is well-behaved if, on any directed path that leads to an output, there are at most $\log(\text{size}(f))$ true multiplication gates. Without loss of generality, we can assume that the circuit f only contains gates that are used to compute at least one of the outputs.

Let x denote the input to circuit f and for any gate g of f let $v(g)$ denote the value computed by gate g when x is provided as input to the circuit. For any gate g that is not an input gate or a constant gate, let g_1 and g_2 denote the two gates it uses as inputs. Clearly, if g is one of $\{+, -, \times, \max, \min, >\}$, $v(g)$ can be computed in polynomial time in $\text{size}(v(g_1)) + \text{size}(v(g_2))$, including transforming it into an irreducible fraction. Thus, in order to show that the circuit can be evaluated in polynomial time, it suffices to show that for all gates g of f, it holds that $\text{size}(v(g)) \leq p(\text{size}(f) + \text{size}(x))$, where p is some fixed polynomial (independent of f and x). In the rest of this proof, we show that

$$\text{size}(v(g)) \leq 6 \cdot \text{size}(f)^3 \cdot \text{size}(x).$$

It is convenient to partition the gates of the circuit depending on their depth. For any gate g in f, we let $d(g)$ denote the depth of the gate in f. The input gates and the constant gates are at depth 1. For any other gate g, we define its depth inductively as $d(g) = 1 + \max\{d(g_1), d(g_2)\}$, where g_1 and g_2 are the two input gates of g. Note that $d(g) \leq \text{size}(f)$ for all gates g in the circuit.

We also define a notion of "multiplication-depth" $md(g)$. The gates g at depth 1 all have $md(g) = 0$. For the rest of the gates, the multiplication-depth is defined inductively. For a gate g whose inputs are g_1 and g_2, we let $md(g) = 1+\max\{md(g_1), md(g_2)\}$ if g is a true multiplication gate, and $md(g) = \max\{md(g_1), md(g_2)\}$ otherwise. Since f is well-behaved, it immediately follows that $md(g) \leq \log(\text{size}(f))$ for all gates g of the circuit.

We begin by showing that for any gate g of f, it holds that

$$|v(g)| \leq 2^{\text{size}(f)^2(\text{size}(x)+\text{size}(f))}.$$

This follows from the stronger statement that

$$|v(g)| \leq 2^{d(g)\cdot 2^{md(g)}\cdot(\text{size}(x)+\text{size}(f))},$$

which we prove by induction as follows. First of all, note that any gate at depth 1 satisfies the statement, since any input or constant of the circuit is bounded by $2^{\text{size}(x)}$ or $2^{\text{size}(f)}$ respectively. Next, assume that the statement holds for all gates with depth $\leq k-1$ and consider some gate g at depth k. Let g_1 and g_2 denote its two inputs, which must satisfy that $d(g_1) \leq k-1$ and $d(g_2) \leq k-1$. If g is one of $\{\min, \max, <\}$, then the statement immediately also holds for g. If g is an addition or subtraction gate, then $|v(g)| \leq |v(g_1)|+|v(g_2)| \leq 2\max\{|v(g_1)|, |v(g_2)|\}$, which implies that the statement also hold for g, since $d(g_1), d(g_2) \leq k-1$ and $d(g) = k$. If g is a multiplication by a constant, then $|v(g)| \leq 2^{\text{size}(f)}|v(g_1)|$ (wlog g_2 is the constant), and the statement holds for g too. Finally, if g is a true multiplication gate, then $|v(g)| = |v(g_1)||v(g_2)| \leq (\max\{|v(g_1)|, |v(g_2)|\})^2$. Since $md(g) = 1 + \max\{md(g_1), md(g_2)\}$, it follows that the statement also holds for g.

Let $den(g)$ denote the absolute value of the denominator of $v(g)$ (written as an irreducible fraction). We show that for all gates g, it holds that $den(g) \leq 2^{\text{size}(f)^2(\text{size}(x)+\text{size}(f))}$. This is enough to conclude our proof. Indeed, since we also have that $|v(g)| \leq 2^{\text{size}(f)^2(\text{size}(x)+\text{size}(f))}$, it follows that the absolute value of the numerator of $v(g)$ is

$$|v(g)| \cdot den(g) \leq 2^{2\cdot\text{size}(f)^2(\text{size}(x)+\text{size}(f))}.$$

As a result, it follows that

$$\begin{aligned}\text{size}(v(g)) &\leq 2 \cdot \text{size}(f)^2(\text{size}(x) + \text{size}(f)) + \text{size}(f)^2(\text{size}(x) + \text{size}(f)) \\ &\leq 6 \cdot \text{size}(f)^3 \cdot \text{size}(x).\end{aligned}$$

It remains to show that $den(g) \leq 2^{\text{size}(f)^2(\text{size}(x)+\text{size}(f))}$, which we prove by showing that $den(g) \leq 2^{d(g)\cdot 2^{md(g)}\cdot(\text{size}(x)+\text{size}(f))}$. Let M denote (the absolute value of) the product of all denominators appearing in the input x and the description of f, i.e., the

denominators of the coordinates of the input x, and the denominators of the constants used by f. Note that $M \leq 2^{\text{size}(x)+\text{size}(f)}$. We prove by induction that for all gates g,

$$den(g) \text{ is a factor of } M^{d(g)\cdot 2^{md(g)}}$$

which in particular implies the bound on $den(g)$ above. First of all, note that any gate at depth 1 is an input or a constant, and thus satisfies the statement. Next, assume that the statement holds for all gates with depth $\leq k-1$ and consider some gate g at depth k. Let g_1 and g_2 denote its two inputs, which must satisfy that $d(g_1) \leq k-1$ and $d(g_2) \leq k-1$. If g is one of $\{\min, \max, <\}$, then it is easy to see that the statement immediately also holds for g. If g is an addition or subtraction gate, then, since $v(g_1)$ and $v(g_2)$ can both be expressed as fractions with denominator $M^{d(g)\cdot 2^{md(g)}}$, so can $v(g)$, and the statement also holds for g. If g is a multiplication by a constant, then $den(g)$ is a factor of $M^{d(g)\cdot 2^{md(g)}}$, since $den(g_1)$ is a factor of $M^{(d(g)-1)\cdot 2^{md(g)}}$ and the denominator of the constant is a factor of M (wlog assume that g_2 is the constant). Finally, if g is a true multiplication gate, then $den(g_1)$ and $den(g_2)$ are factors of $M^{d(g)\cdot 2^{md(g)-1}}$, and thus $den(g)$ is a factor of $(M^{d(g)\cdot 2^{md(g)-1}})^2 = M^{d(g)\cdot 2^{md(g)}}$ as desired. □

A.2 Lipschitz-continuity of linear arithmetic circuits (Proof of Lemma 2.2)

We restate the Lemma here for convenience.

Lemma 2.2. *Any linear arithmetic circuit $f : \mathbb{R}^n \to \mathbb{R}^m$ is $2^{\text{size}(f)^2}$-Lipschitz-continuous (w.r.t. the ℓ_∞-norm) over $\mathbb{R}^n$.*

Proof. For any gate g of the circuit f, let $L(g)$ denote the Lipschitz-constant of the function which outputs the value of g, given the input x to the circuit. As in the proof of Lemma 2.1, it is convenient to partition the gates of f according to their depth. Note that for all the gates g at depth 1, i.e., the input gates and the constant gates, it holds that $L(g) \leq 1$. We show that any gate g at depth k satisfies $L(g) \leq 2^{k\cdot\text{size}(f)}$. It immediately follows from this that f is $2^{\text{size}(f)^2}$-Lipschitz-continuous (w.r.t. the ℓ_∞-norm) over $\mathbb{R}^n$.

Consider a gate g at depth k with inputs g_1 and g_2 (which lie at a lower depth). If g is $+$ or $-$, then $L(g) \leq L(g_1)+L(g_2) \leq 2\max\{L(g_1), L(g_2)\} \leq 2\cdot 2^{(k-1)\cdot\text{size}(f)} \leq 2^{k\cdot\text{size}(f)}$. If g is max or min, then it is easy to see that $L(g) \leq \max\{L(g_1), L(g_2)\} \leq 2^{k\cdot\text{size}(f)}$. Finally, if g is $\times\zeta$, then $L(g) \leq |\zeta| \cdot L(g_1) \leq 2^{\text{size}(f)} 2^{(k-1)\cdot\text{size}(f)} = 2^{k\cdot\text{size}(f)}$, where we used the fact that $|\zeta| \leq 2^{\text{size}(f)}$. □

Appendix B

Approximation by Linear Arithmetic Circuits

In this appendix, we show that linear arithmetic circuits can approximate more general functions with very small error. We begin by restating Theorem 2.1.

Theorem 2.1. *Given a well-behaved arithmetic circuit $f : [0,1]^n \to \mathbb{R}^d$, a purported Lipschitz constant $L > 0$, and a precision parameter $\varepsilon > 0$, in polynomial time in* $\mathrm{size}(f)$, $\log L$ *and* $\log(1/\varepsilon)$, *we can construct a linear arithmetic circuit $F : [0,1]^n \to \mathbb{R}^d$ such that for any $x \in [0,1]^n$ it holds that:*

- $\|f(x) - F(x)\|_\infty \leq \varepsilon$, *or*
- *given x, we can efficiently compute $y \in [0,1]^n$ such that*

$$\|f(x) - f(y)\|_\infty > L\|x - y\|_\infty.$$

Here "efficiently" means in polynomial time in $\mathrm{size}(x)$, $\mathrm{size}(f)$, $\log L$ *and* $\log(1/\varepsilon)$.

Our proof of this result relies on existing techniques introduced by Daskalakis, Goldberg and Papadimitriou [DGP09] and Chen, Deng and Teng [CDT09], but with a modification that ensures that we only get a very small error. Indeed, using the usual so-called sampling trick with averaging does not work here. We modify the sampling trick to output the *median* instead of the average.

Since we believe that this tool will be useful in future work, we prove a more general version of Theorem 2.1. This more general version is Theorem B.1 and it is presented and proved in the next subsection, where we also explain how Theorem 2.1 is easily obtained from Theorem B.1.

Remark B.1. Note that in Theorem 2.1 the domain $[0,1]^n$ can be replaced by $[-M, M]^n$ for any $M > 0$ (in which case the running time is polynomial in the same quantities and in $\log M$). This is easy to show by using a simple bijection between $[0,1]^n$ and

$[-M, M]^n$. This also holds for the more general statement in Theorem B.1. In fact, the result holds for any convex set $S \subseteq [-M, M]^n$, as long as we can efficiently compute the projection onto S. Furthermore, the choice of the ℓ_∞-norm in the statement is not important, and it can be replaced by any other ℓ_p-norm, if f is L-Lipschitz-continuous with respect to that norm.

B.1 General statement and proof

In order to make the statement of the result as general as possible, we consider a generalisation of the notion of a polynomially computable class of functions $\mathcal{F}$ (see Section 2.2.3).

Definition B.1. A class $\mathcal{F}$ of functions is said to be *polynomially-approximately-computable* if there exists a polynomial q such that for any function $f \in \mathcal{F}$ where $f : [0,1]^n \to \mathbb{R}^d$, any point $x \in [0,1]^n$, and any precision parameter $\delta > 0$, a value $v \in \mathbb{R}^d$ such that $\|f(x) - v\|_\infty \leq \delta$ can be computed in time $q(\text{size}(f) + \text{size}(x) + \log(1/\delta))$.

The next theorem basically says that any polynomially-approximately-computable class can be approximated by linear arithmetic circuits, as long as the functions are Lipschitz-continuous.

Theorem B.1. *Let $\mathcal{F}$ be a polynomially-approximately-computable class of functions. Given $f \in \mathcal{F}$ where $f : [0,1]^n \to \mathbb{R}^d$, $L > 0$ and $\varepsilon > 0$, in polynomial time in $\text{size}(f)$, $\log L$ and $\log(1/\varepsilon)$, we can construct a linear arithmetic circuit $F : [0,1]^n \to \mathbb{R}^d$ such that for any $x \in [0,1]^n$ it holds that:*

- $\|f(x) - F(x)\|_\infty \leq \varepsilon$, *or*
- *given x, we can efficiently compute $y \in [0,1]^n$ such that*

$$\|f(x) - f(y)\|_\infty > L\|x - y\|_\infty + \varepsilon/2.$$

Here "efficiently" means in polynomial time in $\text{size}(x)$, $\text{size}(f)$, $\log L$ and $\log(1/\varepsilon)$.

Note that Theorem B.1 immediately implies Theorem 2.1, since the class of all well-behaved arithmetic circuits mapping $[0,1]^n$ to $\mathbb{R}^d$ is polynomially-approximately-computable. (In fact, it is even exactly computable.) Note that since $L\|x-y\|_\infty+\varepsilon/2 \geq L\|x-y\|_\infty$ for any $\varepsilon > 0$, we indeed immediately obtain Theorem 2.1.

Proof of Theorem B.1. First of all, note that we can assume that $d = 1$. Indeed, if $f : [0,1]^n \to \mathbb{R}^d$, then we can consider $f_1, \ldots, f_d : [0,1]^n \to \mathbb{R}$ where $f_i(x) = [f(x)]_i$, and construct linear arithmetic circuits $F_1, \ldots, F_d$ approximating $f_1, \ldots, f_d$ (as in the statement of the theorem). By constructing $F(x) = (F_1(x), \ldots, F_d(x))$, we have then obtained a linear arithmetic that satisfies the statement of the theorem. Indeed, if for some $x \in [0,1]^n$ we have $\|f(x) - F(x)\|_\infty > \varepsilon$, then it follows that there exists $i \in [n]$ such that $|f_i(x) - F_i(x)| > \varepsilon$. From this it follows that we can efficiently compute y with $|f_i(x) - f_i(y)| > L\|x - y\|_\infty + \varepsilon/2$, which implies that $\|f(x) - f(y)\|_\infty > L\|x - y\|_\infty + \varepsilon/2$. Note that $d \leq \text{size}(f)$, so this construction remains polynomial-time with respect to $\text{size}(f)$, $\log L$ and $\log(1/\varepsilon)$.

Consider any $L > 0$, $\varepsilon > 0$ and $f \in \mathcal{F}$ where $f : [0,1]^n \to \mathbb{R}$. Pick $k \in \mathbb{N}$ such that $N := 2^k \geq 4L/\varepsilon$. We consider the partition of $[0,1]^n$ into N^n subcubes of side-length $1/N$. Every $p \in [N]^n$ then represents one subcube of the partition, and we let $\widehat{p} \in [0,1]^n$ denote the centre of that subcube. Formally, for all $p \in [N]^n$, $\widehat{p} \in [0,1]^n$ is given by

$$[\widehat{p}]_i = \frac{2p_i - 1}{2N} \qquad \text{for all } i \in [n].$$

For any $p \in [N]^n$, let $\tilde{f}(\widehat{p})$ denote the approximation of $f(\widehat{p})$ with error at most $\varepsilon/16$. Note that $\tilde{f}(\widehat{p})$ can be computed in time $q(\text{size}(f) + \text{size}(\widehat{p}) + \log(16/\varepsilon))$ (where q is the polynomial associated to $\mathcal{F}$). Since $\text{size}(\widehat{p})$ is polynomial in $\log L$ and $\log(1/\varepsilon)$, we can compute a rational number $M > 0$ such that $\text{size}(M)$ is polynomial in $\text{size}(f)$, $\log L$ and $\log(1/\varepsilon)$, and it holds that $|\tilde{f}(\widehat{p})| \leq M$ for all $p \in [N]^n$. We then define

$$C(p) := \left\lfloor \left(\tilde{f}(\widehat{p}) + M \right) \frac{16}{\varepsilon} \right\rfloor + 1$$

for all $p \in [N]^n$. Note that $C(p) \in [1, 32M/\varepsilon + 1] \cap \mathbb{N}$. Pick $m \in \mathbb{N}$ such that $2^m \geq 32M/\varepsilon + 1$. Then, $C : [N]^n \to [2^m]$ and we construct a boolean circuit $\{0,1\}^{kn} \to \{0,1\}^m$ that computes C. Importantly, the Boolean circuit can be constructed in polynomial time in $\text{size}(f)$, $\log L$ and $\log(1/\varepsilon)$. Before we move on, note that for all $p \in [N]^n$, letting $V(p) := (C(p) - 1)\frac{\varepsilon}{16} - M$, it holds that

$$|f(\widehat{p}) - V(p)| \leq \left| f(\widehat{p}) - \tilde{f}(\widehat{p}) \right| + \left| \tilde{f}(\widehat{p}) - V(p) \right| \leq \frac{\varepsilon}{8} \tag{B.1}$$

since $|f(\widehat{p}) - \tilde{f}(\widehat{p})| \leq \varepsilon/16$ and

$$\begin{aligned}
&\left| \tilde{f}(\widehat{p}) - V(p) \right| \\
&= \left| \tilde{f}(\widehat{p}) + M - \left\lfloor \left(\tilde{f}(\widehat{p}) + M \right) \frac{16}{\varepsilon} \right\rfloor \frac{\varepsilon}{16} \right| \\
&\leq \left| \tilde{f}(\widehat{p}) + M - \left(\tilde{f}(\widehat{p}) + M \right) \frac{16}{\varepsilon} \frac{\varepsilon}{16} \right| + \left| \left(\tilde{f}(\widehat{p}) + M \right) \frac{16}{\varepsilon} - \left\lfloor \left(\tilde{f}(\widehat{p}) + M \right) \frac{16}{\varepsilon} \right\rfloor \right| \frac{\varepsilon}{16}
\end{aligned}$$

$\leq \frac{\varepsilon}{16}$.

Using Lemma B.1, which is our key lemma here and is stated and proved in the next subsection, we can construct a linear arithmetic circuit $F : [0,1]^n \to \mathbb{R}$ in polynomial time in size(C) (and thus in size(f), $\log L$ and $\log(1/\varepsilon)$), such that for all $x \in [0,1]^n$

$$F(x) \in \left[\min_{p \in S(x)} C(p), \max_{p \in S(x)} C(p)\right]$$

where $S(x) \subseteq [N]^n$ is such that

1. $|S(x)| \leq n+1$,
2. $\|x - \widehat{p}\|_\infty \leq 1/N$ for all $p \in S(x)$, and
3. $S(x)$ can be computed in polynomial time in size(x) and $\log N$.

We modify the linear circuit so that instead of outputting $F(x)$, it outputs $(F(x) - 1)\frac{\varepsilon}{16} - M$. Note that this is straightforward to do using the arithmetic gates at our disposal. Since $V(p) = (C(p) - 1)\frac{\varepsilon}{16} - M$, we obtain that for all $x \in [0,1]^n$

$$F(x) \in \left[\min_{p \in S(x)} V(p), \max_{p \in S(x)} V(p)\right].$$

We are now ready to complete the proof. For this it suffices to show that if $|f(x) - F(x)| > \varepsilon$, then the second point in the statement of the theorem must hold. Assume that $x \in [0,1]^n$ is such that $|f(x) - F(x)| > \varepsilon$. It immediately follows that there exists $p^* \in [N]^n$ such that $|f(x) - V(p^*)| > \varepsilon$. By Equation (B.1), it follows that $|f(x) - f(\widehat{p^*})| > \varepsilon - \varepsilon/8 = 7\varepsilon/8$. Note that we might not be able to identify p^*, since we can only approximately compute f. Thus, we instead proceed as follows. We compute $p' = \text{argmax}_{p \in S(x)} |f'(x) - f'(\widehat{p})|$, where f' denotes computation of f with error at most $\varepsilon/32$. Note that p' can be computed in polynomial time in size(x), size(f), $\log L$ and $\log(1/\varepsilon)$, since f' and $S(x)$ can be computed efficiently.

We now show that $y = \widehat{p'} \in [0,1]^n$ satisfies the second point in the statement of the theorem. First of all, note that $|f'(x) - f'(\widehat{p^*})| > 7\varepsilon/8 - 2\varepsilon/32 = 13\varepsilon/16$. By the choice of p', it must be that $|f'(x) - f'(\widehat{p'})| \geq |f'(x) - f'(\widehat{p^*})| > 13\varepsilon/16$, which implies that $|f(x) - f(\widehat{p'})| > 13\varepsilon/16 - 2\varepsilon/32 > 3\varepsilon/4$. On the other hand, since we have that $\|x - \widehat{p'}\|_\infty \leq 1/N \leq \varepsilon/4L$ (because $p' \in S(x)$), it follows that $L\|x - \widehat{p'}\|_\infty \leq \varepsilon/4$. Thus, we indeed have that $|f(x) - f(y)| > L\|x - y\|_\infty + \varepsilon/2$, as desired. Since p' can be computed efficiently, so can $y = \widehat{p'}$. □

B.1.1 Key Lemma

Let us recall some notation introduced in the proof of Theorem B.1. For $N \in \mathbb{N}$, consider the partition of $[0,1]^n$ into N^n subcubes of side-length $1/N$. Every $p \in [N]^n$ then represents one subcube of the partition, and we let $\widehat{p} \in [0,1]^n$ denote the centre of that subcube. Formally, for all $p \in [N]^n$, $\widehat{p} \in [0,1]^n$ is given by

$$[\widehat{p}]_i = \frac{2p_i - 1}{2N}$$

for all $i \in [n]$.

Lemma B.1. *Assume that we are given a Boolean circuit $C : \{0,1\}^{kn} \to \{0,1\}^m$, interpreted as a function $C : [N]^n \to [2^m]$, where $N = 2^k$. Then, in polynomial time in $\mathrm{size}(C)$, we can construct a linear arithmetic circuit $F : [0,1]^n \to \mathbb{R}$ such that for all $x \in [0,1]^n$*

$$F(x) \in \left[\min_{p \in S(x)} C(p), \max_{p \in S(x)} C(p)\right]$$

where $S(x) \subseteq [N]^n$ is such that

1. *$|S(x)| \leq n + 1$,*
2. *$\|x - \widehat{p}\|_\infty \leq 1/N$ for all $p \in S(x)$, and*
3. *$S(x)$ can be computed in polynomial time in $\mathrm{size}(x)$ and $\log N$.*

Proof. We begin by a formal definition of $S(x)$ and prove that it has the three properties mentioned in the statement of the Lemma. We then proceed with the construction of the linear arithmetic circuit.

For $N \in \mathbb{N}$, consider the partition of $[0,1]$ into N subintervals of length $1/N$. Let $I_N : [0,1] \to [N]$ denote the function that maps any point in $[0,1]$ to the index of the subinterval that contains it. In the case where a point lies on the boundary between two subintervals, i.e., $x \in B = \{1/N, 2/N, \ldots, (N-1)/N\}$, the tie is broken in favour of the smaller index. Formally,

$$I_N(x) = \min\left\{\ell \in [N] \;\middle|\; x \in \left[\frac{\ell - 1}{N}, \frac{\ell}{N}\right]\right\}.$$

We abuse notation and let $I_N : [0,1]^n \to [N]^n$ denote the natural extension of the function to $[0,1]^n$, where it is simply applied on each coordinate separately. Thus, if we consider the partition of $[0,1]^n$ into N^n subcubes of side-length $1/N$, then, for any point $x \in [0,1]^n$, $p = I_N(x) \in [N]^n$ is the index of the subcube containing x. For $x \in \mathbb{R}^n \setminus [0,1]^n$, we let $I_N(x) := I_N(y)$ where y is obtained by projecting every coordinate of x onto $[0,1]$.

Letting $\mathbf{e} \in \mathbb{R}^n$ denote the all-ones vector, we define

$$S(x) = \left\{ I_N(x + \alpha \cdot \mathbf{e}) \;\middle|\; \alpha \in \left[0, \frac{1}{2N}\right] \right\}.$$

In other words, we consider a small segment starting at x and moving up simultaneously in all dimensions, and we let $S(x)$ be the set of subcubes-indices of all the points on the segment. We can now prove the three properties of $S(x)$:

1. Note that for any $i \in [n]$, there exists at most one value $\alpha \in [0, 1/2N]$ such that $[x + \alpha \cdot \mathbf{e}]_i \in B = \{1/N, 2/N, \ldots, (N-1)/N\}$. We let α_i denote that value of α if it exists, and otherwise we let $\alpha_i = 1/2N$. Thus, we obtain $\alpha_1, \alpha_2, \ldots, \alpha_n \in [0, 1/2N]$ and we rename them β_i so that they are ordered, i.e., $\beta_1 \leq \beta_2 \leq \cdots \leq \beta_n$ and $\{\beta_i \,|\, i \in [n]\} = \{\alpha_i \,|\, i \in [n]\}$. By the definition of I_N, it is then easy to see that $\alpha \mapsto I_N(x + \alpha \cdot \mathbf{e})$ is constant on each of the intervals $[0, \beta_1]$, $(\beta_1, \beta_2]$, $(\beta_2, \beta_3]$, $\ldots$, $(\beta_{n-1}, \beta_n]$ and $(\beta_n, 1/2N]$. Since these $n+1$ intervals (some of which are possibly empty) cover the entirety of $[0, 1/2N]$, it follows that $|S(x)| \leq n + 1$.

2. Consider any $p \in S(x)$. Let $\alpha \in [0, 1/2N]$ be such that $p = I_N(y)$ where $y = x + \alpha \cdot \mathbf{e}$. For any $i \in [n]$, it holds that $x_i \leq y_i \leq x_i + 1/2N$. There are two cases to consider. If $I_N(y_i) = I_N(x_i)$, then this means that x_i lies in the subinterval of length $1/N$ centred at $[\widehat{p}]_i$, and thus, in particular, $|x_i - [\widehat{p}]_i| \leq 1/2N \leq 1/N$. The only other possibility is that $I_N(y_i) = I_N(x_i) + 1$, since $\alpha \in [0, 1/2N]$. But for this to happen, it must be that $(2I_N(x_i) - 1)/2N \leq x_i \leq 2I_N(x_i)/2N$, since $\alpha \leq 1/2N$. By definition, $[\widehat{p}]_i = (2I_N(y_i) - 1)/2N = (2I_N(x_i) + 1)/2N$, and so we again obtain that $|x_i - [\widehat{p}]_i| \leq 1/N$. Since this holds for all $i \in [n]$, it follows that $\|x - \widehat{p}\|_\infty \leq 1/N$.

3. Given $x \in [0, 1]^n$, the values $\alpha_1, \ldots, \alpha_n \in [0, 1/2N]$, defined in the first point above, can be computed in polynomial time in size(x), n and $\log N$. Then, $S(x)$ can be computed by simply evaluating $I_N(x + \alpha \cdot \mathbf{e})$ for all $\alpha \in \{\alpha_1, \ldots, \alpha_n, 1/2N\}$, which can also be done in polynomial time in size(x), n and $\log N$. Note that since $n \leq$ size(x), the computation of $S(x)$ runs in polynomial time in size(x) and $\log N$.

We can now describe how the linear arithmetic circuit $F : [0, 1]^n \to \mathbb{R}$ is constructed. Let $C : \{0, 1\}^{kn} \to \{0, 1\}^m$ be the Boolean circuit that is provided. It is interpreted as a function $C : [N]^n \to [2^m]$, where $N = 2^k$. Let $x \in [0, 1]^n$ be the input to the linear arithmetic circuit. F is constructed to perform the following steps.

Step 1: Sampling trick. In the first step, we create a sample T of points close to x. This is a standard trick that was introduced in the study of the complexity of computing Nash equilibria [DGP09; CDT09]. Here we use the so-called equi-angle sampling trick introduced by Chen, Deng and Teng [CDT09]. The sample T consists of $2n+1$ points:

$$T = \left\{ x + \frac{\ell}{4nN} \cdot \mathbf{e} \;\middle|\; \ell \in \{0, 1, 2, \ldots, 2n\} \right\}.$$

Note that these $2n+1$ points can easily be computed by F given the input x. The following two observations are important:

1. for all $y \in T$, $I_N(y) \in S(x)$ (by definition of $S(x)$),
2. Let $T_b = \{y \in T \mid \exists i \in [n] : \mathrm{dist}(y_i, B) < \frac{1}{8nN}\}$, where $B = \{1/N, 2/N, \ldots, (N-1)/N\}$ and $\mathrm{dist}(y_i, B) = \min_{t \in B} |y_i - t|$. We call these the *bad* samples, because they are too close to a boundary between two subcubes. The points in $T_g = T \setminus T_b$ are the *good* samples. It holds that $|T_b| \leq n$. This is easy to see by fixing some coordinate $i \in [n]$, and noting that there exists at most one point $y \in T$ such that $\mathrm{dist}(y_i, B) < \frac{1}{8nN}$. Indeed, since the samples are successively $1/4nN$ apart, at most one can be sufficiently close to any given boundary. Furthermore, since the samples are all $1/2N$ close, at most one boundary can be sufficiently close to any of them (for every coordinate). Thus, since there is at most one bad sample for each coordinate, there are at most n bad samples overall.

Step 2: Bit extraction. In the second step, we want to compute $I_N(y)$ for all $y \in T$. This corresponds to extracting the first k bits of each coordinate of each point $y \in T$, because $N = 2^k$. Unfortunately, bit extraction is not a continuous function and thus it is impossible to always perform it correctly with a linear arithmetic circuit. Fortunately, we will show that we can perform it correctly for *most* points in T, namely all the good points in T_g.

Consider any $y \in T$ and any coordinate $i \in [n]$. In order to extract the first bit of y_i, the arithmetic circuit computes

$$b_1 = \min\{1, \max\{0, 8nN(y_i - 1/2)\}\} =: \phi(y_i - 1/2).$$

Note that if $y_i \geq 1/2 + 1/8nN$, then $8nN(y_i - 1/2) \geq 1$ and thus $b_1 = 1$. On the other hand, if $y_i \leq 1/2 - 1/8nN$, then $8nN(y_i - 1/2) \leq -1$ and thus $b_1 = 0$. This means that if $\mathrm{dist}(y_i, B) \geq 1/8nN$, the first bit of y_i is extracted correctly. Note that $B = \{1/N, 2/N, \ldots, (N-1)/N\} = \{1/2^k, 2/2^k, \ldots, (2^k - 1)/2^k\}$.

To extract the second bit, the arithmetic circuit computes $t := y_i - b_1/2$ and

$$b_2 = \phi(t - 1/4).$$

By the same argument as above, b_2 is the correct second bit of y_i, if $|t-1/4| \geq 1/8nN$, i.e., if $|y_i - 1/4| \geq 8nN$ and $|y_i - 3/4| \geq 8nN$. Thus, if $\mathrm{dist}(y_i, B) \geq 1/8nN$, the second bit is also computed correctly, since $1/4, 3/4 \in B$.

To extract the third bit, the arithmetic circuit updates $t := t - b_2/4$ and computes $b_3 = \phi(t - 1/8)$. We proceed analogously up to the kth bit b_k. By induction and the same arguments as above, it follows that the first k bits of y_i are computed correctly by the arithmetic circuit as long as $\mathrm{dist}(y_i, B) \geq 1/8nN$. In particular, this condition always holds for $y \in T_g$.

By performing this bit extraction for each coordinate of each $y \in T$, we obtain the purported bit representation of $I_N(y)$ for each $y \in T$. The argumentation in the previous paragraphs shows that for all $y \in T_g$, we indeed obtain the correct bit representation of $I_N(y)$. For $y \in T_b$, we have no control over what happens, and it is entirely possible that the procedure outputs numbers that are not valid bits, i.e., not in $\{0, 1\}$.

Step 3: Simulation of the Boolean circuit. In the next step, for each $y \in T$, we evaluate the circuit C on the bits purportedly representing $I_N(y)$. The Boolean gates of C are simulated by the arithmetic circuit as follows:

- $\neg b := 1 - b$,
- $b \vee b' := \min\{1, b + b'\}$,
- $b \wedge b' := \max\{0, b + b' - 1\}$.

Note that if the input bits b, b' are valid bits, i.e., in $\{0, 1\}$, then the Boolean gates are simulated correctly, and the output is also a valid bit.

For $y \in T_g$, since the input bits indeed represent $I_N(y)$, the simulation of C will thus output the correct bit representation of $C(I_N(y)) \in [2^m]$. We can obtain the value $C(I_N(y)) \in [2^m]$ itself by decoding the bit representation, i.e., multiplying every bit by the corresponding power of 2 and adding all the terms together.

Let $V(y) \in \mathbb{R}$ denote the value that this step outputs for each $y \in T$. For $y \in T_g$, we have that $V(y) = C(I_N(y))$. For $y \in T_b$, there is no guarantee other than $V(y) \in \mathbb{R}$.

Step 4: Median using a sorting network. In the last step, we want to use the $|T| = 2n + 1$ values that we have computed (namely $\{V(y) \,|\, y \in T\}$) to compute the final output of our arithmetic circuit. In previous constructions of this type, in particular [DGP09; CDT09], the circuit would simply output the average of the $V(y)$. However, this is not good enough to prove our statement, because even a single bad point y can introduce an inversely polynomial error in the average.

In order to obtain a stronger guarantee, the arithmetic circuit instead outputs the *median* of the multiset $\{V(y) \mid y \in T\}$. The median of the given $2n+1$ values can be computed by constructing a so-called sorting network (see, e.g., [Knu98]). The basic operation of a sorting network can easily be simulated by the max and min gates. It is easy to construct a sorting network for $2n+1$ values that has size polynomial in n. The output of the sorting network will be the same values that it had as input, but sorted. In other words, the sorting network outputs $V_1 \leq V_2 \leq \cdots \leq V_{2n+1}$ such that $\{V_j \mid j \in [2n+1]\} = \{V(y) \mid y \in T\}$ as multisets. The final output of our arithmetic circuit is V_{n+1}, which is exactly the median of the $2n+1$ values.

Recall from step 1 that $|T_b| \leq n$ and thus $|T_g| \geq n+1$. It immediately follows that either V_{n+1} corresponds to $V(y)$ of a good sample y, or there exist $i < n+1$ and $j > n+1$ such that both V_i and V_j correspond to good samples. In other words, the output of the circuit satisfies $F(x) \in [\min_{y \in T_g} C(I_N(y)), \max_{y \in T_g} C(I_N(y))]$. As noted in step 1, $I_N(y) \in S(x)$ for all $y \in T$. Thus, we obtain that $F(x) \in [\min_{p \in S(x)} C(p), \max_{p \in S(x)} C(p)]$.

It follows that the linear arithmetic circuit F that we have constructed indeed satisfies the statement of the Lemma. Furthermore, the construction we have described can be performed in polynomial time in size(C), n, m and k. Since size$(C) \geq \max\{n, k, m\}$, it is simply polynomial time in size(C). □

Appendix C

General-Brouwer and General-Real-Localopt

In this appendix we define the computational problems General-Brouwer and General-Real-Localopt and prove that they are PPAD- and PLS-complete respectively. These two completeness results follow straightforwardly from prior work. The membership of General-Brouwer in PPAD and of General-Real-Localopt in PLS are used in this thesis to show that various problems of interest lie in PPAD ∩ PLS.

Definition C.1. **General-Brouwer**:

Input:

- precision parameter $\varepsilon > 0$,
- $(A, b) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m$ defining a bounded non-empty domain $D = \{x \in \mathbb{R}^n : Ax \leq b\}$,
- well-behaved arithmetic circuit $g : \mathbb{R}^n \to \mathbb{R}^n$,
- Lipschitz constant $L > 0$.

Output: An approximate fixed point of g on domain D. Formally, find $x \in D$ such that

$$\|\Pi_D(g(x)) - x\| \leq \varepsilon.$$

Alternatively, we also accept a violation of L-Lipschitzness of g as a solution. Namely, $x, y \in D$ such that $\|g(x) - g(y)\| > L\|x - y\|$.

Proposition C.1. General-Brouwer *is* PPAD*-complete.*

Proof. Various formulations and special cases of the problem of finding a Brouwer fixed point are known to be PPAD-complete [Pap94; CD09; EY10]. The PPAD-hardness of our General-Brouwer problem immediately follows from the PPAD-hardness of the problem on the domain $[0, 1]^2$ and when g is a linear arithmetic circuit, which is known from [Meh14].

The membership in PPAD essentially follows from Proposition 2 in [EY10], where it is shown that finding an approximate fixed point of a Brouwer function that is efficiently computable and continuous, when the domain is a bounded polytope, is in PPAD. In General-Brouwer, the function is not guaranteed to be continuous, but instead we allow violations of Lipschitz-continuity as solutions. However, it can easily be seen that the proof by Etessami and Yannakakis [EY10] also applies to this case. Alternatively, we can also use our Theorem 2.1 to approximate the circuit g by a linear arithmetic circuit (which is necessarily Lipschitz-continuous with a polynomially representable Lipschitz-constant, see Lemma 2.1) and then use [EY10, Proposition 2] directly. Note that since D is bounded, we can easily compute $M > 0$ such that $D \subseteq [-M, M]^n$ (using linear programming). Then, using Theorem 2.1 and Remark B.1, we can approximate g by a linear arithmetic circuit on the domain D. □

Definition C.2. **General-Real-Localopt**:

Input:

- precision/stopping parameter $\varepsilon > 0$,
- $(A, b) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m$ defining a bounded non-empty domain $D = \{x \in \mathbb{R}^n : Ax \leq b\}$,
- well-behaved arithmetic circuits $p : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^n$,
- Lipschitz constant $L > 0$.

Output: An approximate local optimum of p with respect to g on domain D. Formally, find $x \in D$ such that

$$p(\Pi_D(g(x))) \geq p(x) - \varepsilon.$$

Alternatively, we also accept a violation of L-Lipschitzness of p as a solution. Namely, $x, y \in D$ such that $|p(x) - p(y)| > L\|x - y\|$.

Proposition C.2. General-Real-Localopt *is* PLS*-complete.*

Proof. The PLS-hardness of General-Real-Localopt immediately follows from Theorem 2.1 in [DP11], where it is shown that the problem is PLS-complete in the special case where the domain is $[0, 1]^3$. The proof of membership in PLS for the domain $[0, 1]^3$ immediately generalises to $[0, 1]^n$, even for non-fixed n. Thus, it remains to show that we can reduce General-Real-Localopt to the special case where the domain is $[0, 1]^n$.

Note that since D is bounded, we can easily compute $M > 0$ such that $D \subseteq [-M, M]^n$ (using linear programming). We extend p and g to the whole hypercube $[-M, M]^n$ by using the projection onto D, namely $\widehat{p}(x) = p(\Pi_D(x))$ and $\widehat{g}(x) =$

$g(\Pi_D(x))$. Since $\|x - y\| \geq \|\Pi_D(x) - \Pi_D(y)\|$ for all $x, y \in \mathbb{R}^n$, it follows that any violation of L-Lipschitzness for $\widehat{p}$ immediately yields a violation for p. If $x \in [-M, M]^n$ is an ε-approximate local optimum of $\widehat{p}$ with respect to $\widehat{g}$, i.e., $\widehat{p}(\Pi_D(\widehat{g}(x))) \geq \widehat{p}(x) - \varepsilon$, then it immediately follows that $\Pi_D(x) \in D$ is an ε-approximate local optimum of p with respect to g. Thus, we have reduced the problem to the case where the domain is a hypercube $[-M, M]^n$.

The final step is to change the domain from $[-M, M]^n$ to $[0, 1]^n$, which can easily be achieved by letting $\tilde{p}(x) = \widehat{p}(2M \cdot x - M \cdot \mathbf{e})$ and $\tilde{g}(x) = (\widehat{g}(2M \cdot x - M \cdot \mathbf{e}) + M \cdot \mathbf{e})/2M$. Here $\mathbf{e} \in \mathbb{R}^n$ denotes the all-ones vector. A violation of $2ML$-Lipschitzness for $\tilde{p}$ immediately yields a violation of L-Lipschitzness for $\widehat{p}$. Furthermore, if $x \in [0, 1]^n$ is an ε-approximate local optimum of $\tilde{p}$ with respect to $\tilde{g}$, then it is easy to see that $(2M \cdot x - M \cdot \mathbf{e}) \in [-M, M]^n$ is an ε-approximate local optimum of $\widehat{p}$ with respect to $\widehat{g}$. We thus obtain an instance on the domain $[0, 1]^n$ with the functions $\tilde{p}$ and $\tilde{g}$ and $L' = 2ML$ instead of L. By using the same arguments as in [DP11, Theorem 2.1], it follows that the problem lies in PLS. Note that we do not actually need to construct arithmetic circuits that compute $\tilde{p}$ and $\tilde{g}$ (from the given circuits for p and g), because it suffices to be able to compute the functions in polynomial time for the arguments in [DP11] to go through. □

Bibliography

[AB09] S. Arora and B. Barak. *Computational complexity : a modern approach*. Cambridge University Press, 2009 (cit. on p. 10).

[ABB20] J. Aisenberg, M. L. Bonet and S. Buss. "2-D Tucker is PPA complete". In: *Journal of Computer and System Sciences* 108 (2020), pp. 92–103. DOI: 10.1016/j.jcss.2019.09.002 (cit. on pp. 25, 199, 210, 220, 225).

[ABF+21] G. Amanatidis, G. Birmpas, A. Filos-Ratsikas, A. Hollender and A. A. Voudouris. "Maximum Nash welfare and other stories about EFX". In: *Theoretical Computer Science* 863 (2021), pp. 69–85. DOI: 10.1016/j.tcs.2021.02.020 (cit. on p. 9).

[AD54] K. J. Arrow and G. Debreu. "Existence of an Equilibrium for a Competitive Economy". In: *Econometrica* 22.3 (1954), pp. 265–290. DOI: 10.2307/1907353 (cit. on p. 1).

[AFL86] N. Alon, P. Frankl and L. Lovász. "The chromatic number of Kneser hypergraphs". In: *Transactions of the American Mathematical Society* 298.1 (1986), pp. 359–370. DOI: 10.2307/2000624 (cit. on pp. 194, 233).

[AG20] N. Alon and A. Graur. "Efficient Splitting of Measures and Necklaces". In: *arXiv preprint* (2020). URL: https://arxiv.org/abs/2006.16613 (cit. on p. 242).

[AKV04] T. Abbott, D. Kane and P. Valiant. "On Algorithms for Nash Equilibria". 2004. URL: http://web.mit.edu/tabbott/Public/final.pdf (cit. on p. 22).

[Alo87] N. Alon. "Splitting necklaces". In: *Advances in Mathematics* 63.3 (1987), pp. 247–253. DOI: 10.1016/0001-8708(87)90055-7 (cit. on pp. 7, 194, 195, 234–236, 239, 240).

[Alo88] N. Alon. "Some recent combinatorial applications of Borsuk-type theorems". In: *Algebraic, Extremal and Metric Combinatorics* (1988), pp. 1–12. DOI: 10.1017/cbo9780511758881.003 (cit. on p. 234).

[Alo90] N. Alon. "Non-constructive proofs in combinatorics". In: *Proceedings of the International Congress of Mathematicians, Kyoto, Japan*. 1990, pp. 1421–1429 (cit. on p. 234).

[ARV08] H. Ackermann, H. Röglin and B. Vöcking. "On the impact of combinatorial structure on congestion games". In: *Journal of the ACM* 55.6 (2008), 25:1–25:22. DOI: 10.1145/1455248.1455249 (cit. on p. 27).

[Ath01] S. Athey. "Single Crossing Properties and the Existence of Pure Strategy Equilibria in Games of Incomplete Information". In: *Econometrica* 69.4 (2001), pp. 861–889. DOI: 10.1111/1468-0262.00223 (cit. on pp. 5, 76–78, 81, 85).

[AW86] N. Alon and D. B. West. "The Borsuk-Ulam Theorem and Bisection of Necklaces". In: *Proceedings of the American Mathematical Society* 98.4 (1986), pp. 623–628. DOI: 10.2307/2045739 (cit. on p. 234).

[AZ20a] A. A. Ahmadi and J. Zhang. "Complexity aspects of local minima and related notions". In: *arXiv preprint* (2020). URL: https://arxiv.org/abs/2008.06148 (cit. on p. 112).

[AZ20b] A. A. Ahmadi and J. Zhang. "On the complexity of finding a local minimizer of a quadratic function over a polytope". In: *arXiv preprint* (2020). URL: https://arxiv.org/abs/2008.05558 (cit. on p. 111).

[BBM17] D. Bergemann, B. Brooks and S. Morris. "First-Price Auctions With General Information Structures: Implications for Bidding and Revenue". In: *Econometrica* 85.1 (2017), pp. 107–143. DOI: 10.3982/ecta13958 (cit. on pp. 76, 77).

[BCE+98] P. Beame, S. Cook, J. Edmonds, R. Impagliazzo and T. Pitassi. "The relative complexity of NP search problems". In: *Journal of Computer and System Sciences* 57.1 (1998), pp. 3–19. DOI: 10.1006/jcss.1998.1575 (cit. on pp. 5, 19–21, 24, 25, 35, 40–42, 135, 175, 179, 190).

[Ber99] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999 (cit. on p. 119).

[BFH09] F. Brandt, F. Fischer and M. Holzer. "Symmetries and the complexity of pure Nash equilibrium". In: *Journal of Computer and System Sciences* 75.3 (2009), pp. 163–177. DOI: 10.1016/j.jcss.2008.09.001 (cit. on p. 27).

[BG94] M. Bellare and S. Goldwasser. "The Complexity of Decision Versus Search". In: *SIAM Journal on Computing* 23.1 (1994), pp. 97–119. DOI: 10.1137/S0097539792228289 (cit. on p. 11).

[BG97] P. Battigalli and D. Guaitoli. "Conjectural Equilibria and Rationalizability in a Game with Incomplete Information". In: *Decisions, Games and Markets*. Ed. by P. Battigalli, A. Montesano and F. Panunzi. Springer, 1997, pp. 97–124. DOI: 10.1007/978-1-4615-6337-2_4 (cit. on p. 77).

[BGH+21] G. Birmpas, J. Gan, A. Hollender, F. J. Marmolejo-Cossío, N. Rajgopal and A. A. Voudouris. "Optimally Deceiving a Learning Leader in Stackelberg Games". In: *Journal of Artificial Intelligence Research* 72 (2021), pp. 507–531. DOI: 10.1613/jair.1.12542 (cit. on p. 9).

[BGM92] P. Battigalli, M. Gilli and M. C. Molinari. "Learning and Convergence to Equilibrium in Repeated Strategic Interactions: An Introductory Survey". In: *Ricerche Economiche* 46 (1992), pp. 335–378 (cit. on p. 77).

[BHH21] E. Batziou, K. A. Hansen and K. Høgh. "Strong Approximate Consensus Halving and the Borsuk-Ulam Theorem". In: *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP)*. 2021, 24:1–24:20. DOI: 10.4230/LIPIcs.ICALP.2021.24 (cit. on pp. 32, 242).

[BIK+96] P. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi and P. Pudlák. "Lower bounds on Hilbert's Nullstellensatz and propositional proofs". In: *Proceedings of the London Mathematical Society* s3-73.1 (1996), pp. 1–26. DOI: 10.1112/plms/s3-73.1.1 (cit. on pp. 175, 185–187).

[BIQ+17] A. Belovs, G. Ivanyos, Y. Qiao, M. Santha and S. Yang. "On the polynomial parity argument complexity of the combinatorial Nullstellensatz". In: *Proceedings of the 32nd Conference on Computational Complexity (CCC)*. 2017, 30:1–30:24. DOI: 10.4230/LIPIcs.CCC.2017.30 (cit. on p. 26).

[BJ12] S. R. Buss and A. S. Johnson. "Propositional proofs and reductions between NP search problems". In: *Annals of Pure and Applied Logic* 163.9 (2012), pp. 1163–1182. DOI: 10.1016/j.apal.2012.01.015 (cit. on pp. 20, 21, 43, 44, 128, 176, 180, 188, 189, 192, 193).

[BM04] J. Buresh-Oppenheim and T. Morioka. "Relativized NP search problems and propositional proof systems". In: *Proceedings of the 19th IEEE Conference on Computational Complexity (CCC)*. 2004, pp. 54–67. DOI: 10.1109/CCC.2004.1313795 (cit. on p. 21).

[BM20] S. Bubeck and D. Mikulincer. "How to Trap a Gradient Flow". In: *Proceedings of the 33rd Conference on Learning Theory (COLT)*. 2020, pp. 940–960. URL: http://proceedings.mlr.press/v125/bubeck20b.html (cit. on p. 113).

[Boo71] W. M. Boothby. "On two classical theorems of algebraic topology". In: *The American Mathematical Monthly* 78.3 (1971), pp. 237–249. DOI: 10.1080/00029890.1971.11992736 (cit. on p. 46).

[Bor33] K. Borsuk. "Drei Sätze über die n-dimensionale euklidische Sphäre". In: *Fundamenta Mathematicae* 20 (1933), pp. 177–190. DOI: 10.4064\%2Ffm-20-1-177-190 (cit. on p. 25).

[BPR15] N. Bitansky, O. Paneth and A. Rosen. "On the Cryptographic Hardness of Finding a Nash Equilibrium". In: *Proceedings of the 56th Symposium on Foundations of Computer Science (FOCS)*. 2015, pp. 1480–1498. DOI: 10.1109/focs.2015.94 (cit. on p. 22).

[BR11] K. Bhawalkar and T. Roughgarden. "Welfare Guarantees for Combinatorial Auctions with Item Bidding". In: *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2011, pp. 700–709. DOI: 10.1137/1.9781611973082.55 (cit. on p. 80).

[BR21] Y. Babichenko and A. Rubinstein. "Settling the complexity of Nash equilibrium in congestion games". In: *Proceedings of the 53rd ACM Symposium on Theory of Computing (STOC)*. 2021, pp. 1426–1437. DOI: 10.1145/3406325.3451039 (cit. on pp. 134, 135).

[BR92] A. Blum and R. L. Rivest. "Training a 3-node neural network is NP-complete". In: *Neural Networks* 5.1 (1992), pp. 117–127. DOI: 10.1016/S0893-6080(05)80010-3 (cit. on p. 110).

[Bro11] L. E. J. Brouwer. "Über Abbildung von Mannigfaltigkeiten". In: *Mathematische Annalen* 71 (1911), pp. 97–115. DOI: 10.1007/BF01456931 (cit. on pp. 5, 24, 46, 47).

[BSS81] I. Bárány, S. B. Shlosman and A. Szücs. "On a topological generalization of a theorem of Tverberg". In: *Journal of the London Mathematical Society* 2.1 (1981), pp. 158–164. DOI: 10.1112/jlms/s2-23.1.158 (cit. on pp. 7, 194, 195, 215, 216, 234, 236, 240).

[CD09] X. Chen and X. Deng. "On the complexity of 2D discrete fixed point problem". In: *Theoretical Computer Science* 410.44 (2009), pp. 4448–4456. DOI: 10.1016/j.tcs.2009.07.052 (cit. on pp. 24, 49, 148, 214, 215, 257).

[CDDT09] X. Chen, D. Dai, Y. Du and S.-H. Teng. "Settling the Complexity of Arrow-Debreu Equilibria in Markets with Additively Separable Utilities". In: *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2009, pp. 273–282. DOI: 10.1109/FOCS.2009.29 (cit. on pp. 3, 24).

[CDHS20] Y. Carmon, J. C. Duchi, O. Hinder and A. Sidford. "Lower bounds for finding stationary points I". In: *Mathematical Programming* 184 (2020), pp. 71–120. DOI: 10.1007/s10107-019-01406-y (cit. on p. 113).

[CDO15] X. Chen, D. Durfee and A. Orfanou. "On the Complexity of Nash Equilibria in Anonymous Games". In: *Proceedings of the 47th ACM Symposium on Theory of Computing (STOC)*. 2015, pp. 381–390. DOI: 10.1145/2746539.2746571 (cit. on p. 24).

[CDT09] X. Chen, X. Deng and S.-H. Teng. "Settling the complexity of computing two-player Nash equilibria". In: *Journal of the ACM* 56.3 (2009), 14:1–14:57. DOI: 10.1145/1516512.1516516 (cit. on pp. 3, 5, 7, 24, 31, 64, 161, 194, 248, 254, 255).

[CH13] S. Chawla and J. D. Hartline. "Auctions with unique equilibria". In: *Proceedings of the 14th ACM conference on Electronic Commerce (EC)*. 2013, pp. 181–196. DOI: 10.1145/2492002.2483188 (cit. on p. 76).

[CHK+19] A. R. Choudhuri, P. Hubáček, C. Kamath, K. Pietrzak, A. Rosen and G. N. Rothblum. "Finding a Nash equilibrium is no easier than breaking Fiat-Shamir". In: *Proceedings of the 51st ACM Symposium on Theory of Computing (STOC)*. 2019, pp. 1103–1114. DOI: 10.1145/3313276.3316400 (cit. on p. 22).

[Chw89] M. S.-Y. Chwe. "The Discrete Bid First Auction". In: *Economics Letters* 31.4 (Dec. 1989), pp. 303–306. DOI: 10.1016/0165-1765(89)90019-0 (cit. on p. 76).

[CKK+15] I. Caragiannis, C. Kaklamanis, P. Kanellopoulos, M. Kyropoulou, B. Lucier, R. P. Leme and É. Tardos. "Bounding the inefficiency of outcomes in generalized second price auctions". In: *Journal of Economic Theory* 156 (Mar. 2015), pp. 343–388. DOI: 10.1016/j.jet.2014.04.010 (cit. on p. 80).

[CKS16] G. Christodoulou, A. Kovács and M. Schapira. "Bayesian Combinatorial Auctions". In: *Journal of the ACM* 63.2 (Apr. 2016). DOI: 10.1145/2835172 (cit. on p. 80).

[CP14] Y. Cai and C. Papadimitriou. "Simultaneous Bayesian auctions and computational complexity". In: *Proceedings of the 15th ACM Conference on Economics and Computation (EC)*. June 2014, pp. 895–910. DOI: 10.1145/2600057.2602877 (cit. on p. 78).

[CPY17] X. Chen, D. Paparas and M. Yannakakis. "The Complexity of Non-Monotone Markets". In: *Journal of the ACM* 64.3 (2017), 20:1–20:56. DOI: 10.1145/3064810 (cit. on pp. 24, 64).

[CRW19] V. Chatziafratis, T. Roughgarden and J. R. Wang. "On the Computational Power of Online Gradient Descent". In: *Proceedings of the 32nd Conference on Learning Theory (COLT)*. 2019, pp. 624–662. URL: http://proceedings.mlr.press/v99/chatziafratis19a.html (cit. on p. 109).

[CS08] V. Conitzer and T. Sandholm. "New complexity results about Nash equilibria". In: *Games and Economic Behavior* 63.2 (2008), pp. 621–641. DOI: 10.1016/j.geb.2008.02.015 (cit. on p. 78).

[CSVY08] B. Codenotti, A. Saberi, K. Varadarajan and Y. Ye. "The Complexity of Equilibria: Hardness Results for Economies via a Correspondence with Games". In: *Theoretical Computer Science* 408.2–3 (2008), pp. 188–198. DOI: 10.1016/j.tcs.2008.08.007 (cit. on p. 24).

[Cur18] E. Curtin. "Another Short Proof of the Hairy Ball Theorem". In: *The American Mathematical Monthly* 125.5 (2018), pp. 462–463. DOI: 10.1080/00029890.2018.1436836 (cit. on p. 46).

[CWG10] G. Cai, P. R. Wurman and X. Gong. "A Note on Discrete Bid First-Price Auction With General Value Distribution". In: *International Game Theory Review* 12.01 (2010), pp. 75–81. DOI: 10.1142/s0219198910002520 (cit. on p. 76).

[DEF+21] X. Deng, J. R. Edmonds, Z. Feng, Z. Liu, Q. Qi and Z. Xu. "Understanding PPA-completeness". In: *Journal of Computer and System Sciences* 115 (2021), pp. 146–168. DOI: 10.1016/j.jcss.2020.07.006 (cit. on p. 25).

[DFH21] A. Deligkas, A. Filos-Ratsikas and A. Hollender. "Two's Company, Three's a Crowd: Consensus-Halving for a Constant Number of Agents". In: *Proceedings of the 22nd ACM Conference on Economics and Computation (EC)*. 2021, pp. 347–368. DOI: 10.1145/3465456.3467625 (cit. on pp. 8, 242).

[DFK17] X. Deng, Z. Feng and R. Kulkarni. *Octahedral Tucker is PPA-complete*. Tech. rep. TR17-118. Electronic Colloquium on Computational Complexity (ECCC), 2017. URL: https://eccc.weizmann.ac.il/report/2017/118/ (cit. on p. 26).

[DFMS21] A. Deligkas, J. Fearnley, T. Melissourgos and P. G. Spirakis. "Computing exact solutions of consensus halving and the Borsuk-Ulam theorem". In: *J. Comput. Syst. Sci.* 117 (2021), pp. 75–98. DOI: 10.1016/j.jcss.2020.10.006 (cit. on pp. 32, 70, 242).

[DGP09] C. Daskalakis, P. W. Goldberg and C. H. Papadimitriou. "The complexity of computing a Nash equilibrium". In: *SIAM Journal on Computing* 39.1 (2009), pp. 195–259. DOI: 10.1137/070699652 (cit. on pp. 3, 5, 7, 22, 24, 64, 161, 194, 248, 254, 255).

[DM13] D. Dumrauf and B. Monien. "On the PLS-complexity of maximum constraint assignment". In: *Theoretical Computer Science* 469 (2013), pp. 24–52. DOI: 10.1016/j.tcs.2012.10.044 (cit. on p. 27).

[Dol83] A. Dold. "Simple proofs of some Borsuk-Ulam results". In: *Contemp. math* 19 (1983), pp. 65–69 (cit. on pp. 195, 201, 233).

[Dol92] V. L. Dol'nikov. "A generalization of the ham sandwich theorem". In: *Mathematical Notes* 52.2 (1992), pp. 771–779. DOI: 10.1007/BF01236771 (cit. on p. 233).

[DP11] C. Daskalakis and C. Papadimitriou. "Continuous local search". In: *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2011, pp. 790–804. DOI: 10.1137/1.9781611973082.62 (cit. on pp. 3, 17, 19, 26–32, 106, 124–127, 129, 135, 258, 259).

[DQS12] X. Deng, Q. Qi and A. Saberi. "Algorithmic solutions for envy-free cake cutting". In: *Operations Research* 60.6 (2012), pp. 1461–1476. DOI: 10.1287/opre.1120.1116 (cit. on pp. 3, 24).

[DQY20] C. Dang, Q. Qi and Y. Ye. "Computations and Complexities of Tarski's Fixed Points and Supermodular Games". In: *arXiv preprint* (2020). URL: https://arxiv.org/abs/2005.09836 (cit. on p. 135).

[DSZ21] C. Daskalakis, S. Skoulakis and M. Zampetakis. "The Complexity of Constrained Min-Max Optimization". In: *Proceedings of the 53rd ACM Symposium on Theory of Computing (STOC)*. 2021, pp. 1466–1478. DOI: 10.1145/3406325.3451125 (cit. on pp. 107, 109).

[DTZ18] C. Daskalakis, C. Tzamos and M. Zampetakis. "A converse to Banach's fixed point theorem and its CLS-completeness". In: *Proceedings of the 50th ACM Symposium on Theory of Computing (STOC)*. 2018, pp. 44–50. DOI: 10.1145/3188745.3188968 (cit. on pp. 30, 109).

[EG79] M. Eisenberg and R. Guy. "A Proof of the Hairy Ball Theorem". In: *The American Mathematical Monthly* 86.7 (1979), pp. 571–574. DOI: 10.1080/00029890.1979.11994857 (cit. on p. 46).

[EMS09] G. Escamocher, P. B. Miltersen and R. Santillan R. "Existence and Computation of Equilibria of First-price Auctions with Integral Valuations and Bids". In: *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2009, pp. 1227–1228. URL: https://dl.acm.org/doi/10.5555/1558109.1558225 (cit. on pp. 76, 77, 80, 104).

[EPRY20] K. Etessami, C. Papadimitriou, A. Rubinstein and M. Yannakakis. "Tarski's Theorem, Supermodular Games, and the Complexity of Equilibria". In: *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS)*. 2020, 18:1–18:19. DOI: 10.4230/LIPIcs.ITCS.2020.18 (cit. on p. 135).

[EY10] K. Etessami and M. Yannakakis. "On the Complexity of Nash Equilibria and Other Fixed Points". In: *SIAM Journal on Computing* 39.6 (2010), pp. 2531–2597. DOI: 10.1137/080720826 (cit. on pp. 5, 18, 31, 32, 49, 57, 59, 60, 62, 66, 70, 90, 257, 258).

[FFGL20] M. Feldman, H. Fu, N. Gravin and B. Lucier. "Simultaneous auctions without complements are (almost) efficient". In: *Games and Economic Behavior* 123 (Sept. 2020), pp. 327–341. DOI: 10.1016/j.geb.2015.11.009 (cit. on p. 80).

[FFGZ18] A. Filos-Ratsikas, S. K. S. Frederiksen, P. W. Goldberg and J. Zhang. "Hardness Results for Consensus-Halving". In: *Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*. 2018, 24:1–24:16. DOI: 10.4230/LIPIcs.MFCS.2018.24 (cit. on pp. 64, 235, 236, 241).

[FG18] A. Filos-Ratsikas and P. W. Goldberg. "Consensus Halving is PPA-complete". In: *Proceedings of the 50th ACM Symposium on Theory of Computing (STOC)*. 2018, pp. 51–64. DOI: 10.1145/3188745.3188880 (cit. on pp. 4, 26, 235, 236).

[FG19] A. Filos-Ratsikas and P. W. Goldberg. "The Complexity of Splitting Necklaces and Bisecting Ham Sandwiches". In: *Proceedings of the 51st ACM Symposium on Theory of Computing (STOC)*. 2019, pp. 638–649. DOI: 10.1145/3313276.3316334 (cit. on pp. 7, 26, 194, 233–236, 241).

[FGH+21] A. Filos-Ratsikas, Y. Giannakopoulos, A. Hollender, P. Lazos and D. Poças. "On the Complexity of Equilibrium Computation in First-Price Auctions". In: *Proceedings of the 22nd ACM Conference on Economics and Computation (EC)*. 2021, pp. 454–476. DOI: 10.1145/3465456.3467627 (cit. on pp. 7, 80, 82).

[FGHS21] J. Fearnley, P. W. Goldberg, A. Hollender and R. Savani. "The Complexity of Gradient Descent: CLS = PPAD ∩ PLS". In: *Proceedings of the 53rd ACM Symposium on Theory of Computing (STOC)*. 2021, pp. 46–59. DOI: 10.1145/3406325.3451052 (cit. on p. 8).

[FGMS17] J. Fearnley, S. Gordon, R. Mehta and R. Savani. "CLS: New Problems and Completeness". In: *arXiv preprint* (2017). URL: https://arxiv.org/abs/1702.06017 (cit. on pp. 30, 109).

[FGMS20] J. Fearnley, S. Gordon, R. Mehta and R. Savani. "Unique End of Potential Line". In: *Journal of Computer and System Sciences* 114 (2020), pp. 1–35. DOI: 10.1016/j.jcss.2020.05.007 (cit. on pp. 13, 19, 30, 135).

[FHSZ20] A. Filos-Ratsikas, A. Hollender, K. Sotiraki and M. Zampetakis. "Consensus-Halving: Does it Ever Get Easier?" In: *Proceedings of the 21st ACM Conference on Economics and Computation (EC)*. 2020, pp. 381–399. DOI: 10.1145/3391403.3399527 (cit. on pp. 8, 241, 242).

[FHSZ21] A. Filos-Ratsikas, A. Hollender, K. Sotiraki and M. Zampetakis. "A Topological Characterization of Modulo-p Arguments and Implications for Necklace Splitting". In: *Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2021, pp. 2615–2634. DOI: 10.1137/1.9781611976465.155 (cit. on pp. 8, 208, 219, 220).

[FISV06] K. Friedl, G. Ivanyos, M. Santha and Y. F. Verhoeven. "Locally 2-dimensional Sperner problems complete for the Polynomial Parity Argument classes". In: *Italian Conference on Algorithms and Complexity (CIAC)*. 2006, pp. 380–391. DOI: 10.1007/11758471_36 (cit. on p. 25).

[FL86] D. Fudenberg and D. Levine. "Limit Games and Limit Equilibria". In: *Journal of Economic Theory* 38.2 (Apr. 1986), pp. 261–279. DOI: 10.1016/0022-0531(86)90118-3 (cit. on p. 77).

[FPT04] A. Fabrikant, C. Papadimitriou and K. Talwar. "The complexity of pure Nash equilibria". In: *Proceedings of the 36th ACM Symposium on Theory of Computing (STOC)*. 2004, pp. 604–612. DOI: 10.1145/1007352.1007445 (cit. on pp. 27, 134).

[FS15] J. Fearnley and R. Savani. "The Complexity of the Simplex Method". In: *Proceedings of the 47th ACM Symposium on Theory of Computing (STOC)*. 2015, pp. 201–208. DOI: 10.1145/2746539.2746558 (cit. on p. 109).

[FS21] J. Fearnley and R. Savani. "A faster algorithm for finding Tarski fixed points". In: *38th International Symposium on Theoretical Aspects of Computer Science (STACS)*. 2021, 29:1–29:16. DOI: 10.4230/LIPIcs.STACS.2021.29 (cit. on p. 135).

[FT81] R. M. Freund and M. J. Todd. "A constructive proof of Tucker's combinatorial lemma". In: *Journal of Combinatorial Theory, Series A* 30.3 (1981), pp. 321–325. DOI: 10.1016/0097-3165(81)90027-3 (cit. on pp. 7, 195, 203, 225, 227).

[FW16] R. Frongillo and J. Witkowski. “A Geometric Method to Construct Minimal Peer Prediction Mechanisms”. In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. 2016, pp. 502–508. DOI: 10.5555/3015812.3015888 (cit. on p. 77).

[GGM07] G. Gottlob, G. Greco and T. Mancini. “Complexity of Pure Equilibria in Bayesian Games”. In: *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI)*. 2007, pp. 1294–1299. DOI: 10.5555/1625275.1625485 (cit. on p. 78).

[GH21] P. W. Goldberg and A. Hollender. “The Hairy Ball problem is PPAD-complete”. In: *Journal of Computer and System Sciences* 122 (2021), pp. 34–62. DOI: 10.1016/j.jcss.2021.05.004 (cit. on pp. 7, 47, 63).

[GHI+20] P. W. Goldberg, A. Hollender, A. Igarashi, P. Manurangsi and W. Suksompong. “Consensus halving for sets of items”. In: *Proceedings of the 16th International Conference on Web and Internet Economics (WINE)*. 2020, pp. 384–397. DOI: 10.1007/978-3-030-64946-3_27 (cit. on pp. 8, 64, 68).

[GHS20] P. W. Goldberg, A. Hollender and W. Suksompong. “Contiguous Cake Cutting: Hardness Results and Approximation Algorithms”. In: *Journal of Artificial Intelligence Research* 69 (2020), pp. 109–141. DOI: 10.1613/jair.1.12222 (cit. on p. 9).

[GKRS19] M. Göös, P. Kamath, R. Robere and D. Sokolov. “Adventures in Monotone Complexity and TFNP”. In: *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*. 2019, 38:1–38:19. DOI: 10.4230/LIPIcs.ITCS.2019.38 (cit. on pp. 19, 21).

[GKSZ20] M. Göös, P. Kamath, K. Sotiraki and M. Zampetakis. “On the Complexity of Modulo-q Arguments and the Chevalley-Warning Theorem”. In: *Proceedings of the 35th Computational Complexity Conference (CCC)*. 2020, 19:1–19:42. DOI: 10.4230/LIPIcs.CCC.2020.19 (cit. on pp. 173, 194).

[GLS67] J. H. Griesmer, R. E. Levitan and M. Shubik. “Toward a Study of Bidding Processes Part IV – Games With Unknown Costs”. In: *Naval Research Logistics* 14.4 (1967), pp. 415–433. DOI: 10.1002/nav.3800140402 (cit. on pp. 76, 77).

[GP18] P. W. Goldberg and C. H. Papadimitriou. “Towards a unified complexity theory of total functions”. In: *Journal of Computer and System Sciences* 94 (2018), pp. 167–192. DOI: 10.1016/j.jcss.2017.12.003 (cit. on pp. 19, 20).

[GP74] V. Guillemin and A. Pollack. *Differential topology*. Prentice-Hall, 1974 (cit. on p. 47).

[GPS16] S. Garg, O. Pandey and A. Srinivasan. “Revisiting the Cryptographic Hardness of Finding a Nash Equilibrium”. In: *Proceedings of the 36th International Cryptology Conference (CRYPTO)*. 2016, pp. 579–604. DOI: 10.1007/978-3-662-53008-5_20 (cit. on p. 22).

[Gri01] M. Grigni. “A Sperner lemma complete for PPA”. In: *Information Processing Letters* 77.5–6 (2001), pp. 255–259. DOI: 10.1016/S0020-0190(00)00152-6 (cit. on p. 25).

[GW85] C. H. Goldberg and D. B. West. “Bisection of Circle Colorings”. In: *SIAM Journal on Algebraic Discrete Methods* 6.1 (1985), pp. 93–106. DOI: 10.1137/0606010 (cit. on p. 234).

[Hah73] F. Hahn. *On the Notion of Equilibrium in Economics: An Inaugural Lecture [By] F.H. Hahn*. Cambridge University Press, 1973 (cit. on p. 77).

[Har12] J. D. Hartline. "Bayesian Mechanism Design". In: *Foundations and Trends in Theoretical Computer Science* 8.3 (2012), pp. 143–263. DOI: 10.1561/0400000045 (cit. on p. 76).

[Har67] J. C. Harsanyi. "Games With Incomplete Information Played by "Bayesian" Players, I–III: Part I. The Basic Model". In: *Management Science* 14.3 (1967), pp. 159–182. DOI: 10.1287/mnsc.1040.0270 (cit. on pp. 76, 77).

[HNY17] P. Hubácek, M. Naor and E. Yogev. "The Journey from NP to TFNP Hardness". In: *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*. 2017, 60:1–60:21. DOI: 10.4230/LIPIcs.ITCS.2017.60 (cit. on p. 21).

[Hol21] A. Hollender. "The classes PPA-k: Existence from arguments modulo k". In: *Theoretical Computer Science* 885 (2021), pp. 15–29. DOI: 10.1016/j.tcs.2021.06.016 (cit. on p. 8).

[HY20] P. Hubácek and E. Yogev. "Hardness of continuous local search: Query complexity and cryptographic lower bounds". In: *SIAM Journal on Computing* 49.6 (2020), pp. 1128–1172. DOI: 10.1137/17M1118014 (cit. on pp. 22, 148, 156).

[Imp95] R. Impagliazzo. "A personal view of average-case complexity". In: *Proceedings of the 10th Structure in Complexity Theory Conference (CCC)*. 1995, pp. 134–147. DOI: 10.1109/SCT.1995.514853 (cit. on p. 21).

[Ish21] T. Ishizuka. "The complexity of the parity argument with potential". In: *Journal of Computer and System Sciences* 120 (2021), pp. 14–41. DOI: 10.1016/j.jcss.2021.03.004 (cit. on pp. 20, 44, 135).

[Jeř16] E. Jeřábek. "Integer factoring and modular square roots". In: *Journal of Computer and System Sciences* 82.2 (2016), pp. 380–394. DOI: 10.1016/j.jcss.2015.08.001 (cit. on pp. 4, 20, 21, 193).

[Jeř17] E. Jeřábek. Theoretical Computer Science Stack Exchange. https://cstheory.stackexchange.com/q/37794 (version: 2017-03-20). 2017 (cit. on pp. 172, 178, 180).

[JKKZ21] R. Jawale, Y. T. Kalai, D. Khurana and R. Zhang. "SNARGs for Bounded Depth Computations and PPAD Hardness from Sub-Exponential LWE". In: *Proceedings of the 53rd ACM Symposium on Theory of Computing (STOC)*. 2021, pp. 708–721. DOI: 10.1145/3406325.3451055 (cit. on p. 22).

[JLS21] A. Jain, H. Lin and A. Sahai. "Indistinguishability Obfuscation from Well-Founded Assumptions". In: *Proceedings of the 53rd ACM Symposium on Theory of Computing (STOC)*. 2021, pp. 60–73. DOI: 10.1145/3406325.3451093 (cit. on p. 22).

[Joh11] A. S. Johnson. "Reductions and propositional proofs for total NP search problems". PhD thesis. UC San Diego, 2011. URL: https://escholarship.org/uc/item/89r774x7 (cit. on pp. 172, 173, 180, 185, 188–190, 192, 193).

[JPY88] D. S. Johnson, C. H. Papadimitriou and M. Yannakakis. "How easy is local search?" In: *Journal of Computer and System Sciences* 37.1 (1988), pp. 79–100. DOI: 10.1016/0022-0000(88)90046-3 (cit. on pp. 3, 19, 26, 27).

[JR01] G. A. Jehle and P. J. Reny. *Advanced Microeconomic Theory*. Financial Times/Prentice Hall, 2001 (cit. on p. 77).

[JT04] T. Jarvis and J. Tanton. "The Hairy Ball Theorem via Sperner's Lemma". In: *The American Mathematical Monthly* (2004), pp. 599–603. DOI: 10.1080/00029890.2004.11920120 (cit. on pp. 46, 47, 50).

[KL93] E. Kalai and E. Lehrer. "Rational Learning Leads to Nash Equilibrium". In: *Econometrica* 61.5 (1993), pp. 1019–1045. DOI: 10.2307/2951492 (cit. on p. 77).

[KL95] E. Kalai and E. Lehrer. "Subjective Games and Equilibria". In: *Games and Economic Behavior* 8.1 (1995), pp. 123–163. DOI: 10.1016/s0899-8256(05)80019-3 (cit. on p. 77).

[Kne55] M. Kneser. "Aufgabe 360". In: *Jahresbericht der Deutschen Mathematiker-Vereinigung* 2.27 (1955), pp. 3–16 (cit. on p. 194).

[Knu98] D. E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley Professional, 1998 (cit. on p. 256).

[KNY19] I. Komargodski, M. Naor and E. Yogev. "White-Box vs. Black-Box Complexity of Search Problems: Ramsey and Graph Property Testing". In: *Journal of the ACM* 66.5 (2019), 34:1–34:28. DOI: 10.1145/3341106 (cit. on p. 21).

[KPR+13] S. Kintali, L. J. Poplawski, R. Rajaraman, R. Sundaram and S.-H. Teng. "Reducibility among Fractional Stability Problems". In: *SIAM Journal on Computing* 42.6 (2013), pp. 2063–2113. DOI: 10.1137/120874655 (cit. on p. 24).

[Kre89] M. W. Krentel. "Structure in locally optimal solutions". In: *Proceedings of the 30th Symposium on Foundations of Computer Science (FOCS)*. 1989, pp. 216–221. DOI: 10.1109/SFCS.1989.63481 (cit. on pp. 3, 27).

[Kre90] M. W. Krentel. "On finding and verifying locally optimal solutions". In: *SIAM Journal on Computing* 19.4 (1990), pp. 742–749. DOI: 10.1137/0219052 (cit. on p. 27).

[Kri09] V. Krishna. *Auction Theory*. 2nd. Academic Press, 2009 (cit. on pp. 76, 78, 81).

[KTK80] M. K. Kozlov, S. P. Tarasov and L. G. Khachiyan. "The polynomial solvability of convex quadratic programming". In: *USSR Computational Mathematics and Mathematical Physics* 20.5 (1980), pp. 223–228. DOI: 10.1016/0041-5553(80)90098-1 (cit. on pp. 30, 112).

[Kuh60] H. W. Kuhn. "Some combinatorial lemmas in topology". In: *IBM Journal of research and development* 4.5 (1960), pp. 518–524. DOI: 10.1147/rd.45.0518 (cit. on p. 197).

[LB10] B. Lucier and A. Borodin. "Price of Anarchy for Greedy Auctions". In: *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Jan. 2010, pp. 537–553. DOI: 10.1137/1.9781611973075.46 (cit. on p. 80).

[Leb06] B. Lebrun. "Uniqueness of the equilibrium in first-price auctions". In: *Games and Economic Behavior* 55.1 (Apr. 2006), pp. 131–151. DOI: 10.1016/j.geb.2005.01.006 (cit. on p. 80).

[Leb96] B. Lebrun. "Existence of an Equilibrium in First Price Auctions". In: *Economic Theory* 7.3 (1996), pp. 421–443. URL: http://www.jstor.org/stable/25054935 (cit. on pp. 76–78).

[Leb99] B. Lebrun. "First Price Auctions in the Asymmetric N Bidder Case". In: *International Economic Review* 40.1 (1999), pp. 125–142. URL: https://www.jstor.org/stable/2648842 (cit. on pp. 76, 77).

[Lov78] L. Lovász. "Kneser's conjecture, chromatic number, and homotopy". In: *Journal of Combinatorial Theory, Series A* 25.3 (1978), pp. 319–324. DOI: 10.1016/0097-3165(78)90022-5 (cit. on p. 194).

[LP00] A. Lizzeri and N. Persico. "Uniqueness and Existence of Equilibrium in Auctions with a Reserve Price". In: *Games and Economic Behavior* 30.1 (Jan. 2000), pp. 83–114. DOI: 10.1006/game.1998.0704 (cit. on pp. 76, 77).

[LT10] R. P. Leme and E. Tardos. "Pure and Bayes-Nash Price of Anarchy for Generalized Second Price Auction". In: *Proceedings of the 51st Symposium on Foundations of Computer Science (FOCS)*. 2010, pp. 735–744. DOI: 10.1109/focs.2010.75 (cit. on p. 80).

[LŽ06] M. de Longueville and R. T. Živaljević. "The Borsuk-Ulam-property, Tucker-property and constructive proofs in combinatorics". In: *Journal of Combinatorial Theory, Series A* 113.5 (2006), pp. 839–850. DOI: 10.1016/j.jcta.2005.08.002 (cit. on p. 235).

[Mat08] J. 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 (cit. on pp. 25, 195, 197, 200, 202, 226).

[McG16] P. McGrath. "An extremely short proof of the Hairy Ball theorem". In: *The American Mathematical Monthly* 123.5 (2016), pp. 502–503. DOI: 10.4169/amer.math.monthly.123.5.502 (cit. on p. 46).

[Meh14] R. Mehta. "Constant Rank Bimatrix Games Are PPAD-hard". In: *Proceedings of the 46th ACM Symposium on Theory of Computing (STOC)*. 2014, pp. 545–554. DOI: 10.1145/2591796.2591835 (cit. on pp. 24, 62, 257).

[Meu08] F. Meunier. "Discrete Splittings of the Necklace". In: *Mathematics of Operations Research* 33.3 (2008), pp. 678–688. DOI: 10.1287/moor.1080.0311 (cit. on p. 234).

[Meu14] F. Meunier. "Simplotopal maps and necklace splitting". In: *Discrete Mathematics* 323 (2014), pp. 14–26. DOI: 10.1016/j.disc.2014.01.008 (cit. on pp. 195, 222, 235–237, 240).

[Mil78] J. Milnor. "Analytic proofs of the "hairy ball theorem" and the Brouwer fixed point theorem". In: *The American Mathematical Monthly* 85.7 (1978), pp. 521–524. DOI: 10.1080/00029890.1978.11994635 (cit. on pp. 46, 59, 60).

[MK87] K. G. Murty and S. N. Kabadi. "Some NP-complete problems in quadratic and nonlinear programming". In: *Mathematical Programming* 39.2 (1987), pp. 117–129. DOI: 10.1007/BF02592948 (cit. on pp. 110, 111).

[MMRS94] R. C. Marshall, M. J. Meurer, J.-F. Richard and W. Stromquist. "Numerical Analysis of Asymmetric First Price Auctions". In: *Games and Economic Behavior* 7.2 (1994), pp. 193–220. DOI: 10.1006/game.1994.1045 (cit. on pp. 76, 77).

[MMSS17] F. Meunier, W. Mulzer, P. Sarrabezolles and Y. Stein. "The rainbow at the end of the line—a PPAD formulation of the colorful Carathéodory theorem with applications". In: *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2017, pp. 1342–1351. DOI: 10.1137/1.9781611974782.87 (cit. on p. 135).

[MN12] F. Meunier and B. Neveu. "Computing solutions of the paintshop-necklace problem". In: *Computers and Operations Research* 39.11 (2012), pp. 2666–2678. DOI: 10.1016/j.cor.2012.01.014 (cit. on p. 234).

[Mor01] T. Morioka. "Classification of search problems and their definability in bounded arithmetic". MA thesis. University of Toronto, 2001. URL: https://www.collectionscanada.ca/obj/s4/f2/dsk3/ftp04/MQ58775.pdf (cit. on pp. 21, 26).

[MP91] N. Megiddo and C. H. Papadimitriou. "On total functions, existence theorems and computational complexity". In: *Theoretical Computer Science* 81.2 (1991), pp. 317–324. DOI: 10.1016/0304-3975(91)90200-L (cit. on pp. 2, 12, 18, 76, 112).

[MR00] E. Maskin and J. Riley. "Equilibrium in Sealed High Bid Auctions". In: *The Review of Economic Studies* 67.3 (2000), pp. 439–454. URL: http://www.jstor.org/stable/2566961 (cit. on pp. 76–78, 80, 81).

[MR03] E. Maskin and J. Riley. "Uniqueness of equilibrium in sealed high-bid auctions". In: *Games and Economic Behavior* 45.2 (2003), pp. 395–409. DOI: 10.1016/S0899-8256(03)00150-7 (cit. on p. 80).

[MS09] F. Meunier and A. Sebő. "Paintshop, odd cycles and necklace splitting". In: *Discrete Applied Mathematics* 157.4 (2009), pp. 780–793. DOI: 10.1016/j.dam.2008.06.017 (cit. on p. 234).

[MS94] P. Milgrom and C. Shannon. "Monotone Comparative Statics". In: *Econometrica* 62.1 (1994), pp. 157–180. DOI: 10.2307/2951479 (cit. on p. 77).

[Mun84] J. R. Munkres. *Elements of algebraic topology*. Westview Press, 1984. DOI: 10.1201/9780429493911 (cit. on p. 197).

[Mye81] R. B. Myerson. "Optimal Auction Design". In: *Mathematics of Operations Research* 6.1 (1981), pp. 58–73. DOI: 10.1287/moor.6.1.58 (cit. on pp. 75, 77).

[Mye97] R. B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1997 (cit. on pp. 76, 77).

[Nas50] J. F. Nash. "Equilibrium points in n-person games". In: *Proceedings of the National Academy of Sciences* 36.1 (1950), pp. 48–49. DOI: 10.1073/pnas.36.1.48 (cit. on p. 1).

[Pap90] C. H. Papadimitriou. "On graph-theoretic lemmata and complexity classes". In: *Proceedings of the 31st Symposium on Foundations of Computer Science (FOCS)*. 1990, pp. 794–801. DOI: 10.1109/FSCS.1990.89602 (cit. on pp. 24, 41).

[Pap92] C. H. Papadimitriou. "The complexity of the Lin-Kernighan heuristic for the traveling salesman problem". In: *SIAM Journal on Computing* 21.3 (1992), pp. 450–465. DOI: 10.1137/0221030 (cit. on p. 27).

[Pap94] C. H. Papadimitriou. "On the complexity of the parity argument and other inefficient proofs of existence". In: *Journal of Computer and System Sciences* 48.3 (1994), pp. 498–532. DOI: 10.1016/S0022-0000(05)80063-7 (cit. on pp. 3, 6, 18–22, 24, 25, 41, 49, 50, 66, 172–175, 179, 188, 225, 234, 257).

[Plu92] M. Plum. "Characterization and Computation of Nash-Equilibria for Auctions with Incomplete Information". In: *International Journal of Game Theory* 20.4 (Dec. 1992), pp. 393–418. DOI: 10.1007/bf01271133 (cit. on pp. 76, 77).

[Poi85] H. Poincaré. "Sur les courbes définies par les équations différentielles (III)". In: *Journal de Mathématiques Pures et Appliquées* 4.1 (1885), pp. 167–244 (cit. on pp. 5, 46, 47).

[PST20] R. Paes Leme, B. Sivan and Y. Teng. "Why Do Competitive Markets Converge to First-Price Auctions?" In: *Proceedings of The World Wide Web Conference (WWW)*. 2020, pp. 596–605. DOI: 10.1145/3366423.3380142 (cit. on p. 76).

[Pud15] P. Pudlák. "On the complexity of finding falsifying assignments for Herbrand disjunctions". In: *Archive for Mathematical Logic* 54.7-8 (2015), pp. 769–783. DOI: 10.1007/s00153-015-0439-6 (cit. on p. 18).

[PV20] R. Pass and M. Venkitasubramaniam. "Is it Easier to Prove Theorems that are Guaranteed to be True?" In: *Proceedings of the 61st Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 1255–1267. DOI: 10.1109/FOCS46700.2020.00119 (cit. on p. 21).

[RG21] I. Rasooly and C. Gavidia-Calderon. "The importance of being discrete: on the (in-)accuracy of continuous approximations in auction theory". In: *CoRR* abs/2006.03016 (2021). arXiv: 2006.03016. URL: http://arxiv.org/abs/2006.03016 (cit. on p. 76).

[RS81] J. G. Riley and W. F. Samuelson. "Optimal Auctions". In: *The American Economic Review* 71.3 (1981), pp. 381–392. URL: https://www.jstor.org/stable/1802786 (cit. on pp. 76, 77).

[RSS21] A. Rosen, G. Segev and I. Shahaf. "Can PPAD Hardness be Based on Standard Cryptographic Assumptions?" In: *Journal of Cryptology* 34 (2021). DOI: 10.1007/s00145-020-09369-6 (cit. on p. 22).

[Rub18] A. Rubinstein. "Inapproximability of Nash equilibrium". In: *SIAM Journal on Computing* 47.3 (2018), pp. 917–959. DOI: 10.1137/15M1039274 (cit. on pp. 5, 24, 64, 65).

[Rus95] W. S. Russell. "Polynomial interpolation schemes for internal derivative distributions on structured grids". In: *Applied Numerical Mathematics* 17.2 (1995), pp. 129–171. DOI: 10.1016/0168-9274(95)00014-L (cit. on pp. 156, 159).

[RW94] A. Rubinstein and A. Wolinsky. "Rationalizable Conjectural Equilibrium: Between Nash and Rationalizability". In: *Games and Economic Behavior* 6.2 (Mar. 1994), pp. 299–311. DOI: 10.1006/game.1994.1016 (cit. on p. 77).

[RZ04] P. J. Reny and S. Zamir. "On the Existence of Pure Strategy Monotone Equilibria in Asymmetric First-Price Auctions". In: *Econometrica* 72.4 (July 2004), pp. 1105–1125. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-0262.2004.00527.x (cit. on pp. 76, 77, 81).

[Sar91] K. S. Sarkaria. "A generalized van Kampen-Flores theorem". In: *Proceedings of the American Mathematical Society* 111.2 (1991), pp. 559–565. DOI: 10.2307/2048349 (cit. on p. 194).

[Sch91] A. A. Schäffer. "Simple local search problems that are hard to solve". In: *SIAM journal on Computing* 20.1 (1991), pp. 56–87. DOI: 10.1137/0220004 (cit. on pp. 3, 27).

[Sip06] M. Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology Boston, 2006 (cit. on p. 10).

[Spe28] E. Sperner. "Neuer Beweis für die Invarianz der Dimensionszahl und des Gebietes". In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 6 (1928), pp. 265–272. DOI: 10.1007/BF02940617 (cit. on p. 24).

[SS03] F. W. Simmons and F. E. Su. "Consensus-halving via theorems of Borsuk-Ulam and Tucker". In: *Mathematical social sciences* 45.1 (2003), pp. 15–25. DOI: 10.1016/S0165-4896(02)00087-2 (cit. on pp. 195, 235–237, 240).

[SSB17] S. Schuldenzucker, S. Seuken and S. Battiston. "Finding Clearing Payments in Financial Networks with Credit Default Swaps is PPAD-complete". In: *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*. 2017, 32:1–32:20. DOI: 10.4230/LIPIcs.ITCS.2017.32 (cit. on p. 64).

[Str80] W. Stromquist. "How to Cut a Cake Fairly". In: *The American Mathematical Monthly* 87.8 (1980), pp. 640–644. DOI: 10.1080/00029890.1980.11995109 (cit. on p. 1).

[SV08] A. Skopalik and B. Vöcking. "Inapproximability of pure Nash equilibria". In: *Proceedings of the 40th ACM Symposium on Theory of Computing (STOC)*. 2008, pp. 355–364. DOI: 10.1145/1374376.1374428 (cit. on p. 27).

[SZZ18] K. Sotiraki, M. Zampetakis and G. Zirdelis. "PPP-Completeness with Connections to Cryptography". In: *Proceedings of the 59th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2018, pp. 148–158. DOI: 10.1109/FOCS.2018.00023 (cit. on p. 4).

[Tod76] M. J. Todd. *The computation of fixed points and applications*. Springer, 1976 (cit. on pp. 56, 57).

[Tuc45] A. W. Tucker. "Some Topological Properties of Disk and Sphere". In: *Proceedings of the First Canadian Math. Congress, Montreal*. University of Toronto Press, 1945, pp. 286–309 (cit. on p. 25).

[Vav93] S. A. Vavasis. "Black-Box Complexity of Local Minimization". In: *SIAM Journal on Optimization* 3.1 (1993), pp. 60–80. DOI: 10.1137/0803004 (cit. on p. 113).

[Vic61] W. Vickrey. "Counterspeculation, Auctions and Competitive Sealed Tenders". In: *Journal of Finance* 16.1 (Mar. 1961), pp. 8–37. DOI: 10.1111/j.1540-6261.1961.tb02789.x (cit. on pp. 75, 76).

[VY11] V. V. Vazirani and M. Yannakakis. "Market Equilibrium under Separable, Piecewise-Linear, Concave Utilities". In: *Journal of the ACM* 58.3 (2011), 10:1–10:25. DOI: 10.1145/1970392.1970394 (cit. on p. 24).

[WP12] J. Witkowski and D. C. Parkes. "Peer prediction without a common prior". In: *Proceedings of the 13th ACM Conference on Electronic Commerce*. 2012, pp. 964–981. DOI: 10.1145/2229012.2229085 (cit. on p. 77).

[WSZ20] Z. Wang, W. Shen and S. Zuo. "Bayesian Nash Equilibrium in First-Price Auction with Discrete Value Distributions". In: *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2020, pp. 1458–1466. URL: https://dl.acm.org/doi/abs/10.5555/3398761.3398929 (cit. on pp. 78, 80).

[ZV90] R. T. Zivaljević and S. T. Vrećica. "An extension of the ham sandwich theorem". In: *Bulletin of the London Mathematical Society* 22.2 (1990), pp. 183–186. DOI: 10.1112/blms/22.2.183 (cit. on p. 233).