On TFNP Classes: Approaches from Fixed Point Theory and Algorithmic Game Theory

石塚, 天

出版情報：Kyushu University, 2022, 博士（数理学）, 課程博士
バージョン：
権利関係：

KYUSHU UNIVERSITY

DOCTORAL THESIS

On TFNP Classes: Approaches from Fixed Point Theory and Algorithmic Game Theory

Author:
Takashi ISHIZUKA

Supervisor:
Naoyuki KAMIYAMA

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy [Mathematics]

in the

Graduate School of Mathematics

July 11, 2022

KYUSHU UNIVERSITY

Abstract

Graduate School of Mathematics

Doctor of Philosophy [Mathematics]

On TFNP Classes: Approaches from Fixed Point Theory and Algorithmic Game Theory

by Takashi ISHIZUKA

While NP-hardness has played a crucial role in guaranteeing the intractability of many fascinating problems, there still are left many natural problems that are seemingly not effortless to solve and are not known to be NP-hard. The complexity class TFNP captures the complexity of such problems. This class is made up of total search problems, and the correctness of solutions is polynomial-time verifiable. The TFNP classes, like PLS (Polynomial Local Search) and PPAD (Polynomial Parity Argument for Digraphs), constitute an essential and fascinating complexity-theoretical research field. This thesis provides new results for search problems belonging to TFNP classes.

In the first part, we consider the complexity of search problems on an exponentially-large graph whose vertices have positive values. We show the robustness of the definition of EOPL by END OF POTENTIAL LINE. Furthermore, we provide new PPA$\cap$PLS-complete problems.

The rest of this thesis focuses on the computational aspects of some search problems from the viewpoints of Fixed Point Theory and Algorithmic Game Theory. In the second part, we consider the complexity of finding a fixed point whose existence is guaranteed by Caritsti's fixed point theorem. We prove the PLS completeness of this problem. Thus, Caristi's fixed point theorem characteries the complexity class PLS. Furthermore, we discuss the complexity of computing a fixed point whose existence is guaranteed by Brøndsted's fixed point theorem.

The final part deal with the complexity of equilibrium computation. First, we consider the hardness of distinguishing the existence of uniform Nash equilibria on a two-player strategic-form game. Second, we consider the complexity of finding a pure Nash equilibrium on a discrete preference game and a network coordination game; it is well known that these graphical games always have a pure Nash equilibrium. Finally, we study the complexity of finding a stable fractional matching on a hypergraphic preference system, a generalization of a roommate matching model. We provide a tractable-intractable boundary for this problem.

Acknowledgements

I wish to thank my supervisor, Naoyuki Kamiyama, who collaborated with me on much of this research. After I proposed that I wanted to study the computational complexity around PPAD, he led me into the wonderful TFNP world with careful guidance.

I want to express gratefulness to the members of the Quantum Information Science Subgroup of Computing Theory Research Group of NTT Communication Science Laboratories, especially Seiichiro Tani and Yuki Takeuchi. It has been a great pleasure to discuss with Dr. Tani and Dr. Takeuchi for the past two years, which was an excellent inspiration for me. It was Dr. Tani who hosted my internship and accompanied my (slow and awkward) first steps in exploring the quantum algorithm and complexity theory.

I would also like to express my appreciation for the staff at the department of my university, who were always very helpful with various administrative tasks. I received the fellowship from JSPS; my research projects were supported by JSPS KAKENHI Grant Numbers JP21J10845 and JST ACT-X Grant Number JPMJAX2101, Japan.

I wish to thank Christos Papadimitriou, who hosted my visit supported by JSPS Research Fellowship for Young Scientists, which was not realized due to the COVID-19 situation.

I am grateful to my examiners, Yoshihiro Mizoguchi, Koji Nuida, and Yukiko Yamauchi, for carefully reading my dissertation and providing some comments and suggestions that helped improve its presentation.

To make my college-life marvelous, I am deeply thankful to everyone I came to an acquaintance with via enjoying Go and other tabletop games, especially members of the Kyushu University Go Club and my local board-game friends. Finally, my Ph.D. graduation would not have been realized without the love and support from my family.

Contents

Dedicated to the benefactors for my degree.

Part I

Overview

Chapter 1

Introduction

An algorithm has become one of the fundamental concepts indispensable to our lives. We cannot imagine a world without algorithms (at least the author thinks so). Algorithms represent how to compute problems we want to solve; anyone can find a solution according to a suitable algorithm. Usually, algorithms are carried out on computers. Hence, it is an essential quest to explore an algorithm that efficiently solves a problem appearing in the real world.

Unfortunately, we do not precisely catch the notion of computation or algorithms, although computers are widely utilized. An excellent example is the "P versus NP" problem[1], which is an unsolved problem listed in the million-dollar prize problems of the Clay Math Institute[2]. Some researchers claim that "This is the most important open problem for mathematics and computer sciences." Interestingly, Scott Aaronson has formed an opinion of this open problem in his book [Aar13] as follows:

> Look, this P versus NP question, what can I say? People like to describe it as "probably the central unsolved problem of theoretical computer science." That's a comical understatement. P vs. NP is one of the deepest questions that human begins have ever asked.

Hence, the "P versus NP" problem is a significant open problem worth considering. It is an essential and fascinating issue to discuss the existence of efficient algorithms to solve practical problems. Computational Complexity Theory is a realm that treats the "P versus NP" problem as a central topic. Both terms P and NP are the set of decision problems. Experts on Computational Complexity Theory primarily have paid attention to the complexity of decision problems, a problem that can be answered with YES or NO.

Have we forgotten something crucial? Obviously, we can envision a variety of computational tasks which can be answered with neither YES nor NO in the practical world. For example, we want to search for the shortest route to arrive at the destination; we need to allocate students to their desired courses; we wish to divide a positive integer into some primes; and so on. We call such problems, which require finding an actual solution that is not necessarily YES or NO, *search problems*.

Most of the physical world's problems we are interested in are total search problems. A search problem is *total* if every instance has at least one solution. For instance, the shortest route to arrive at a destination always exists; every positive

[1] Note that the class P is the set of polynomial-time solvable decision problems. On the other hand, the class NP contains many decision problems that are concerned hard to compute.

[2] See http://www.claymath.org/millennium-problems/p-vs-np-problem.

integer greater than one can be factorized into prime factors. Many natural problems with at least one solution are seemingly not NP-hard. Moreover, it is unknown an efficient algorithm for these problems. The complexity class TFNP, introduced by Megiddo and Papadimitriou [MP91], captures the complexity of such computational problems. It is widely known that computational problems that are important in applied fields belong to TFNP.

Economics is known as a major application of TFNP. Equilibrium is an important and key concept in economics. Their field has a motive to compute an equilibrium when their existence is guaranteed. These computational problems, often called equilibrium computation, have been widely studied in our realm [CD07; Das09; Yan09]. A Nash equilibrium is a basic example of equilibrium theory. However, we had long been puzzled about how to capture the computational complexity of finding it. Nash's theorem [NJ50] made it unsuitable for handling in the context of decision problems; furthermore, any efficient algorithm to compute a Nash equilibrium is unknown.

Some TFNP classes resolve such an obstacle. Papadimitriou [Pap94b] has proven that computing a Nash equilibrium on a strategic-form game is a PPAD-complete problem; the complexity class PPAD is a set of total search problems that the existence of solutions is guaranteed by the parity argument for digraphs. Various other equilibria (e.g., pure Nash equilibria, market equilibria, stable solution, etc.) are characterized by TFNP classes such as PPAD, PPA, and PLS. These proofs are based on Fixed Point Theory; some fixed point theorems guarantee the existence of equilibria. Thus, we use the problem of computing a fixed point to clarify the complexity of equilibrium computation.

The complexity class TFNP and its subclasses have been a game-changer for discussing problems that we want to deal with in the real world in computer science. However, the complexity of TFNP classes also remains elusive; the hardness of handling TFNP is closely related to the "P versus NP" problem [MP91; Pap94b].

We will anatomize the complexity of TFNP classes from the perspective of Fixed Point Theory and Algorithmic Game Theory.

1.1 Total Search Problem in NP

Total search problems in NP are not unfamiliar to us. In fact, they are all around us in our daily lives. For example, cryptography, stable configuration of neural networks, stable allocations, and equilibria on economic models. The computational complexity aspect of total search problems has widely been utilized.

Primarily, Johnson, Papadimitriou, and Yannakakis [JPY88] defined the class PLS as a class of search problems that can be solved by a local search method. A local search method is a procedure: While there is an improvement among the current neighbor, we change the current candidate to it.

A few years later, Megiddo and Papadimitriou [MP91] formulated the complexity class TFNP, consisting of total search problems in NP. Search problems in TFNP have the following properties: (i) Every input instance always has a solution; (ii) their correctness is efficiently verifiable. The class TFNP contains some of the most fundamental, elegant, and intriguing computational problems: factoring, computing a Nash equilibrium, finding a local optimum, etc. These problems have no known

efficient algorithms, but it seems to be not NP-hard either. If TFNP has an NP-hard problem, then we have NP = coNP [MP91; Pap94a].

Unfortunately, TFNP is a *semantic* class[3]. Generally, such classes seem to have complete problems like language classes RP, ZPP, and BPP [Pap94b]. Thus, we focus on *syntactic* subclasses of TFNP. The following four classes formulated by existence proofs are best-well known:

PLS: Every acyclic digraph has a sink (*Local Search*).

PPP: Every function mapping $\{0,1,2,\ldots,N\}$ to $\{1,2,\ldots,N\}$ must have a collision (*Pigeonhole Principle*).

PPA: Every graph with an odd-degree vertex must have another odd-degree node (*Parity Argument*).

PPAD: Every digraph with a vertex whose in-degree and out-degree are different must have another such vertex (*Parity Argument for Digraph*).

These Papadimitriou's classes make it possible to classify total search problems according to existence proofs.

This thesis will discuss the complexity of TFNP classes from the three aspects. First, we consider the fundamental theory of computation. We deal with the problem belonging to both PPA and PLS. Second, we consider the complexity of fixed-point computation, a computational problem that lies at the interface of Theoretical Computer Science and Pure Mathematics. Finally, we focus on the complexity of equilibrium computation, a computational problem located at the interface of Theoretical Computer Science and Economics.

1.2 Outline of the Thesis and Main Contributions

1.2.1 Oganization

Chapter 2 presents notations that are used throughout this thesis. The rest of this thesis is made up of three parts:

Part II Fundamental Theory of Computational Complexity,

Part III Fixed Point Theory,

Part IV Algorithmic Game Theory,

In Part II, we organize central results around TFNP classes. Chapter 3 presents some classes contained in TFNP and briefly reviews recent research streams of TFNP complexity classes. After that, we consider the complexity of the problems on an exponentially large graph that every vertex has a potential value in Chapter 4. We prove the robustness of END OF POTENTIAL LINE and provide a PPA ∩ PLS-complete problem.

[3] We say that a complexity class is semantic if there is no easy way of identifying non-deterministic Turing machines that define problems in that class (see page 255 of the Papadimitriou book [Pap94b]).

Part III considers the complexity of finding a fixed point. Chapter 5 surveys the results of fixed-point computation[4]. In order to think about the real-valued problems, we introduce the arithmetic circuit in Section 5.1. To deal with TFNP problems, we use the *well-behaved* notion introduced by Fearnley, Goldberg, Hollender, and Savani [Fea+21]. Chapter 6 shows the complexity of computing a Caristi's fixed point; we prove that Caristi's fixed point theorem [Car76] characterizes the complexity class PLS.

Finally, we consider the computational problems related to Game Theory in Part IV. The first chapter of this part, Chapter 7, presents the research interaction between Computational Complexity Theory and Game Theory. This chapter presents a list of some studies on the interface of these two fields. Chapter 8 shows a connection between two-player strategic-form games and graph-theoretical techniques. We prove that checking the existence of uniform Nash equilibria is NP-complete even in the two-player setting. Chapter 9 considers a succinctly represented multi-player game called a discrete preference game. It is known that this game always has a pure Nash equilibrium [CKO18; Lol+19]. However, finding it is not easy. We provide an unfamiliar tractable case of computing a pure Nash equilibrium on a discrete preference game. We consider, in Chapter 10, a problem of finding a stable fractional matching on a hypergraphic preference system, which is a generalization of a roommate matching model. It is known that this problem is PPAD-complete [Kin+13]. We present a computational boundary for this search problem; we show that the low degree of a hypergraph setting is easy, but the high degree setting is very hard.

Remark that we have organized this thesis so that the first chapter of each part, Chapters 3, 5, and 7, will present us with a mainstream of this realm. The other parts summarize our contributions; detailed contributions are given in the following section.

1.2.2 Main Contributions

Part II: Fundamental Theory of Computational Complexity

We relax the problem called LEAF (see Definition 3.15), which is a canonical PPA-complete problem. In Section 4.4.1, we introduce the new problem called POTENTIAL LEAF (see Definition 4.31). This problem is the simplest generalization variant of END OF POTENTIAL LINE. We prove that this problem is also EOPL-complete.

We also relax the problem called ODD (see Definition 3.17), a generalization of LEAF. Informally speaking, LEAF is associated with undirected graphs with a degree of at most two. On the other hand, ODD is associated with undirected graphs which possibly contain a vertex with a degree more than two. The parity argument also works for undirected graphs with a higher degree, i.e., ODD is also PPA-complete [Pap94b]. We introduce the new problem called POTENTIAL ODD and show that this problem is EOPL-complete. However, if there is a vertex with a degree of at least four, then POTENTIAL ODD is PPA$\cap$PLS-complete. Roughly speaking, we deal with the complexity of the problem of finding an unknown odd-degree vertex or a local maximum or minimum vertex on an exponentially large undirected graph

[4]Chapter 5 is a revised version of the author's survey in Japanese (see here: https://doi.org/10.14943/101654).

with potential. The PPAD-completeness of IMBALANCE is proved by Goldberg and Hollender [GH19]. In Chapter 4, we expand their elegant result to END OF POTENTIAL LINE, and we show that multiple source variants of END OF POTENTIAL LINE are also EOPL-complete. From this fact, we can immediately show that POTENTIAL IMBALANCE is an EOPL-complete problem. Therefore, we obtain an important and fascinating complexity gap.

Part III: Fixed Point Theory

We focus on the computational aspects of Caristi's fixed point theorem. In other words, we consider the complexity of computing a Caristi's fixed point. It is mathematically known that Caristi's fixed point theorem is an order-theoretic fixed point theorem and a generalization of Banach's fixed point theorem. In fact, we can prove the existence of Banach's fixed points by using Caristi's fixed point theorem [GD03].

Chapter 6 presents that Caristi's fixed point theorem characterizes PLS. Specifically, we show that the problem of finding a Caristi's fixed point is PLS-complete. Remark that the complexity class PLS has been known since the 1980s, but a fixed point theorem that characterizes this class was not known.

Furthermore, we consider a variant of computing a Caristi's fixed point. The existence of a solution to this problem is guaranteed by Brøndsted's fixed point theorem. We provide, in Chapter 6, an upper bound and a lower bound for the problem of finding a Brøndsted's fixed point. More specifically, we show that this problem is CLS-hard and belongs to PPAD.

Part IV: Algorithmic Game Theory

In this part, we compile the three papers which are closely related to Algorithmic Game Theory. First, Chapter 8 focuses on uniform Nash equilibria for a two-player strategic-form game. We study the complexity of computing a uniform Nash equilibrium on a bimatrix game. In general, it is known that such a problem is NP-complete even if a game is a win-lose bimatrix game [BIL08]. However, if the bipartite digraph defined by a win-lose bimatrix game is planar, then there is a polynomial-time algorithm to find a uniform Nash equilibrium [AOV07]. It is still open how hard it is to compute a uniform Nash equilibrium on a bimatrix game that is planar but not win-lose. This paper presents the NP-hardness for the problem of deciding whether a given planar bimatrix game has uniform Nash equilibria even if every component of both players' payoff matrices consists of three types of non-zero elements.

Second, Chapter 9 considers two types of graphical games which always have pure Nash equilibria. We present the polynomial-time computability of equilibrium computation for discrete preference games and network coordination games beyond $O(\log n)$-treewidth and tree metric spaces. The first result shows that we can efficiently find a pure Nash equilibrium for a discrete preference game on a grid graph and a product metric of some metric spaces. On a tree metric space, it is known that we can find it in polynomial time regardless of the structure of a players' network. To prove our polynomial-time computability, we provide a new characterization of a discrete preference game and slightly generalize the recent result in [Lol+19]. We also discuss the complexity of finding a pure Nash equilibrium for a two-strategic network coordination game whose cost functions satisfy submodularity. In this case,

if every cost function is symmetric, then the games are reducible to a discrete preference game on a path metric.

Finally, Chapter 10 studies the complexity of computing a stable hypergraphic fractional matching in a hypergraphic preference system. Aharoni and Fleiner [AF03] have proven that there exists a stable fractional matching in every hypergraphic preference system. Furthermore, Kintali, Poplawski, Rajaraman, Sundaram, and Teng [Kin+13] have shown that the problem of finding a stable fractional matching in a hypergraphic preference system is PPAD-complete. In Chapter 10, we consider the complexity of the problem of finding a stable fractional matching in a hypergraphic preference system whose maximum degree is bounded by some constant. Note that the proof by Kintali, Poplawski, Rajaraman, Sundaram, and Teng [Kin+13] implies the PPAD-completeness of the problem of finding a stable fractional matching in a hypergraphic preference system whose maximum degree is five. We prove that this problem is PPAD-complete even if the maximum degree is three in Section 10.3, this problem can be solved in polynomial time if the maximum degree is two in Section 10.4, and the problem of finding an approximate stable fractional matching in a hypergraphic preference is PPAD-complete in Section 10.5.

1.3 List of Papers

The results presented in this thesis can be found in the following papers:

Chapter 4 is based on Takashi Ishizuka. "The complexity of parity argument with potential," Journal of Computer and System Sciences, 2021 [Ish21b].

DOI: https://doi.org/10.1016/j.jcss.2021.03.004

Chapter 6 is based on Takashi Ishizuka."On the complexity of Caristi's fixed point theory," Information Processing Letters, 2021 [Ish21a].

DOI: https://doi.org/10.1016/j.ipl.2021.106119

Chapter 8 is based on Takashi Ishizuka and Naoyuki Kamiyama. "NP-hardness of Computing Uniform Nash Equilibria on Planar Bimatrix Game," arXiv:2205.03117, 2022 [IK22a].

DOI: https://doi.org/10.48550/arXiv.2205.03117

Chapter 9 is based on Takashi Ishizuka and Naoyuki Kamiyama. "On Finding Nash Equilibria of Discrete Preference Games and Network Coordination Games," arXiv:2207.01523, 2022 [IK22b].

DOI: https://doi.org/10.48550/arXiv.2207.01523

Chapter 10 is based on Takashi Ishizuka and Naoyuki Kamiyama. "On the Complexity of Stable Fractional Hypergraph Matching," Proceedings of the 29th International Symposium on Algorithms and Computation, 2018 [IK18].

DOI: https://doi.org/10.4230/LIPIcs.ISAAC.2018.11

Chapter 2

Preliminaries

We now present the notations that are used throughout this thesis.

2.1 Notation

2.1.1 Standard Notation

We denote by $\mathbb{Z}$ and $\mathbb{R}$ the sets of integers and real numbers, respectively. Furthermore, we denote by $\mathbb{Z}_{\geq 0}, \mathbb{Z}_{>0}$, and $\mathbb{R}_{\geq 0}$ the sets of non-negative integers, positive integers, and non-negative real numbers, respectively. For each positive integer n, we define $[n] := \{1, 2, \ldots, n\}$. For each finite set X, we denote by $|X|$ the number of elements contained in X.

2.1.2 Metric Spaces

A space (L, d) is a metric space if the function $d\colon L \times L \to \mathbb{R}_{\geq 0}$ satisfies the following conditions: for all $x, y, z \in L$

(i) $d(x, y) = 0$ if and only if $x = y$;

(ii) $d(x, y) = d(y, x)$;

(iii) $d(x, y) \leq d(x, z) + d(z, y)$.

Specifically, we refer to a metric space (L, d) whose distance satisfies that $d(x, y) = 1$ whenever $x \neq y$ as a discrete metric.

A graph metric is represented by an edge-weighted undirected graph. The distance between any pair of points is the weight of the minimum weight path in the graph between the corresponding vertices. A graph metric is a tree or path metric if the graph is a tree or a path, respectively.

For some $\ell \in \mathbb{Z}_{>0} \cup \{\infty\}$, a strategy space $\mathscr{M} = (L, d)$ is the ℓ-product metric space of k metric spaces $\mathscr{M}_1 = (L_1, d_1), \ldots, \mathscr{M}_k = (L_k, d_k)$ if $L = L_1 \times \cdots \times L_k$ and for any two points $x = (x^1, \ldots, x^k)$ and $y = (y^1, \ldots, y^k)$ in L, the distance $d(x, y)$ is defined as $\|(d_1(x^1, y^1), \ldots, d_k(x^k, y^k))\|_\ell^\ell$, where $\|\cdot\|_\ell$ means the ℓ-norm if $\ell \in \mathbb{Z}_{>0}$; otherwise we define $d(x, y) = \max_{t \in [k]} \{d_t(x^t, y^t)\}$.

Let (M, d) and (M', d') be metric spaces. A sequence $\{x_n\}_{n \in \mathbb{Z}_{\geq 0}}$ over M is called a Cauchy sequence if for every $\varepsilon > 0$, there exists a number $N(\varepsilon) \in \mathbb{Z}_{\geq 0}$ such that $d(x_n, x_m) < \varepsilon$ whenever $n, m \geq N(\varepsilon)$. A metric space (M, d) is complete if every

Cauchy sequence converges in M. A function $f : M \to \mathbb{R}_{\geq 0_+}$ is lower semicontinuous if for each $a \in \mathbb{R}_{\geq 0}$, $\{x \in M \ ; \ f(x) \leq a\}$ is closed. A function $f : M \to M'$ is λ-Lipschitz continuous with respect to (d, d') if for every pair of points $x, y \in M$, $d'(f(x), f(y)) \leq \lambda d(x, y)$. It is easy to see that every function $f : M \to \mathbb{R}_{\geq 0}$ which is λ-Lipschitz continuous is a lower semicontinuous function.

2.1.3 Languages and Relations

We denote by Σ the finite set of symbols. Each finite sequence of symbols in Σ is called a string. That is, for each string s with respect to Σ, there exists a positive integer n such that $s = s_1 s_2 \ldots s_n$ where $s_i \in \Sigma$ for all $i \in [n]$. Here, we define by Σ^* a set of all finite strings and define by Σ^n a set of all strings of length n. Throughout this thesis, let $\Sigma = \{0, 1\}$. The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. For each pair of strings x, y in Σ^n, we denote by $\|x - y\|$ the Hamming distance between x and y. A language is defined as a subset of Σ^*, and a relation is defined as a subset of $\Sigma^* \times \Sigma^*$.

An instance of a search problem typically contains a few functions. Specifically, a problem on a graph is presented via a function that computes the neighbors of a given vertex. Here, we consider mostly the white-box model; this is an explicit computational model in which each function $f : \Sigma^n \to \Sigma^m$ is given explicitly by a Boolean circuit with n inputs and m outputs. Throughout this thesis, we assume that each function given by an instance is represented by a Boolean circuit, and each Boolean circuit is encoded by a binary representation.

2.1.4 Implicit Graphs

We consider a graph produced by some Boolean circuits that compute the neighborhood of a given vertex. In an implicit graph, we can effortlessly obtain only local property around each given vertex, and it probably takes exponential effort to obtain the whole structure of the graph.

We consider mostly graphs with potential in this thesis. Each vertex in such a graph has a non-negative value as a potential. A function computing the potential of a given vertex is called a potential function.

Directed Graphs

We consider the given two functions $S, P : \Sigma^n \to \Sigma^{dn}$, where d and n are positive integers. Here, for each vertex x in Σ^n, we view as $S(x) = \{y_1, \ldots, y_d\}$, where each string y_i is in Σ^n, similarly P do. Thus, two functions S and P output the set of at most d vertices[1]. Then we consider the implicit digraph $G(S, P) = (\Sigma^n, E)$. Each element of Σ^n is called a vertex, and the set Σ^n is called a vertex set. The functions S and P are called a successor function and a predecessor function, respectively. For each pair of vertices (x, y) in $\Sigma^n \times \Sigma^n$, (x, y) is called a valid arc if it holds that $y \in S(x)$ and $x \in P(y)$. Furthermore, a vertex x in Σ^n is called a self-loop if $S(x) = \{x\}$ and

[1] We suppose that when a vertex has a degree strictly less than d, two functions S and P output the remaining points as itself without loss of generality.

$P(x) = \{x\}$. The set E consists of all valid arcs with respect to S and P, i.e., we define $E := \{(x,y) \in \Sigma^n \times \Sigma^n; x \neq y,\ y \in S(x),\ \text{and}\ x \in P(y)\}$. For each vertex v in Σ^n, an arc (u,v) in E is called an incoming arc of v, and an arc (v,w) in E is called an outgoing arc of v. Furthermore, we denote by $E^-(v)$ the set of all incoming arcs of v, and denote by $E^+(v)$ the set of all outgoing arcs of v. For every vertex v in Σ^n, we define the in-degree of v as $\delta^-(v) = |E^-(v)|$, the out-degree of v as $\delta^+(v) = |E^+(v)|$, and the degree of v as $\delta(v) = \delta^-(v) + \delta^+(v)$. Finally, we define the degree of the implicit graph $G(S,P)$ as $\deg(G) = \max_{v \in \Sigma^n} \delta(v)$.

In particular, when $d = 1$, the implicit graph $G(S,P)$ is a directed graph with in-degree/out-degree at most one. Then every valid arc (x,y) in E satisfies that $S(x) = y$ and $P(y) = x$. Furthermore, a vertex s in Σ^n is called a source if it satisfies that $S(P(s)) \neq s$, and a vertex t in Σ^n is called a sink if it satisfies that $P(S(t)) \neq t$.

Undirected Graphs

When we are given a function $N : \Sigma^n \to \Sigma^{dn}$, we consider the implicit graph $G(N) = (\Sigma^n, E)$, where d and n are positive integers. For each string x in Σ^n, we view as $N(x) = \{y_1, \ldots, y_d\}$, where each string y_i is contained in Σ^n. The function N is called the neighborhood function, and the set Σ^n is called the vertex set. A set $\{x,y\} \subseteq \Sigma^n$ satisfying that $x \neq y$ is called a valid edge if it holds that $y \in N(x)$ and $x \in N(y)$. A vertex x in Σ^n is called an isolated vertex if it holds that $N(x) \subseteq \{x\}$. Then the set E consists of all valid edges with respect to N, i.e., we define $E := \{\{x,y\} \subseteq \Sigma^n; x \neq y, y \in N(x),\ \text{and}\ x \in N(y)\}$. For each vertex v in Σ^n, the degree of x is defined as $\deg_G(x) = |\{y \in \Sigma^n; \{x,y\} \in E\}|$. Furthermore, the degree of the graph G is defined as $\deg(G) = \max_{v \in \Sigma^n} \deg(v)$.

In particular, when $d = 2$, the implicit graph $G(N)$ is an undirected graph with a degree at most two. Then the vertex x with degree one is called a leaf. It is well known that the total number of all leaves on $G(N)$ is always even [Pap94b].

Part II

Fundamental Theory of Computational Complexity

Chapter 3

Theory of Computation

3.1 What is Computation?

We begin with a brief overview of various computational complexity-theoretical notions that will be needed for the remainder of this thesis. We assume that the reader has sufficient knowledge about the theory of computation. So, we intend only that this overview will establish our notation and highlight the main concepts we will need. Readers not familiar with the theory of computation are referred to the books such as [AB09; Sip97; Pap94a].

A decision problem associated with a language $L \subseteq \Sigma^*$ is defined as follows:

Definition 3.1. The decision problem L:
Input:

- a string $x \in \Sigma^*$.

Task: Decide whether

- x belongs to L.

We say that a decision problem L belongs to the complexity class NP if there is a polynomial-time bounded non-deterministic Turing machine that distinguishes whether every input string is in L. We say that a decision problem L belongs to the complexity class P if there is a polynomial-time bounded deterministic Turing machine that distinguishes whether every input string is in L. Remark that we often say that a problem L is efficiently solvable if $L \in$ P.

Let A, B be two problems. A polynomial-time reduction from A to B consists of a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ satisfying that $x \in A$ if and only if $f(x) \in B$. Here, a function $f : \Sigma^* \to \Sigma^*$ is said to be polynomial-time computable if we have a polynomial-time bounded deterministic Turing machine that halts after outputting a string $f(x)$ for every input string $x \in \Sigma^*$. When two problems A and B are polynomial-time reducible to each other, we say that A and B are polynomially equivalent.

A decision problem L is said to be NP-hard if every decision problem in NP is polynomial-time reducible to L. A language L is NP-complete if L is in NP and NP-hard. Note that the problem 3-SAT (see Definition 3.2) is one of the well-known NP-complete problems [Coo71; Lev73].

Definition 3.2. 3-SAT:
Input:

- a 3-CNF formula $\phi : \{0,1\}^n \to \{0,1\}$.

Task: Decide whether

- there exists an assignment $x \in \{0,1\}^n$ such that $\phi(x) = 1$.

In other words, decision problems in NP always have a concise witness to every *yes*-instance while it may be no succinct witness for a *no*-instance. The complexity class coNP is defined as the complement of the class NP. Thus, every *no*-instance for a decision problem in coNP always has a brief witness certificating it is a no-instance. The problem UNSAT is a coNP-complete problem.

Definition 3.3. UNSAT:
Input:

- a 3-CNF formula $\phi : \{0,1\}^n \to \{0,1\}$.

Task: Decide whether

- there exist no assignments $x \in \{0,1\}^n$ such that $\phi(x) = 1$.

Therefore, decision problems in NP have succinct witnesses, whereas decision problems in coNP have succinct disqualifications. These facts imply that the class NP$\cap$coNP is the set of all decision problems that have both: for every *yes*-instance, there is a concise witness; on the other hand, there is a short disproof for every *no*-instance. Examples of decision problems that are known as the membership of NP$\cap$coNP but unknown as polynomial-time computability are the problem Simple Stochastic Game [Con92] and the problem Arrival [Doh+17].

The complexity class TFNP is a functional analog of NP$\cap$coNP [MP91]. From the above observation, each instance of a problem in NP$\cap$coNP always has a short indication that convinces us it is a *yes*- or *no*-instance. We consider the problem of finding such an indication as a total search problem.

3.2 What is a Search Problem?

A computational search problem is defined by using a relation. Let $R \subseteq \Sigma^* \times \Sigma^*$ be a relation. R is said to be polynomial-time computable if we can decide whether $(x,y) \in R$ in polynomial time for each pair of strings $(x,y) \in \Sigma^* \times \Sigma^*$. We call that R is a polynomial-balanced relation if there exists some polynomial p such that for each pair of strings $(x,y) \in R$, $|y| \leq p(|x|)$. Throughout this thesis, we assume that every relation is polynomial-time computable and a polynomial-balanced relation. We define a search problem with respect to a relation R as follows [Pap94b].

Definition 3.4. A search problem R:
Input:

- a string x in Σ^n.

Task: Return

- a string y in Σ^* such that (x,y) is in R if such a y exists

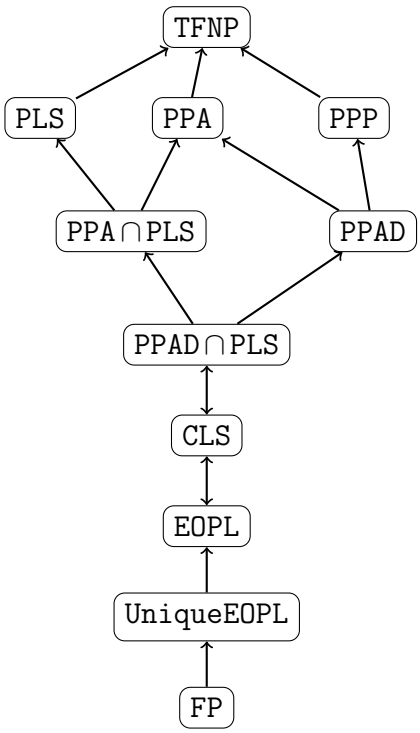

FIGURE 3.1: The relationship of TFNP classes.

- a string "*no*" otherwise.

Here, to simplify notation, a search problem with respect to R is referred to as a problem R. The class of all search problems that are polynomial-time computable and polynomial-time balanced relations is called the class FNP. We denote by FP subclass of FNP that contains all search problems which can be solved in polynomial time. Class TFNP is the subclass of FNP containing all search problems with totality. We say that a search problem R has totality if for every instance $x \in \Sigma^*$, there always exists a string $y \in \Sigma^*$ such that $(x,y) \in R$. It is clear that FP $\subseteq$ TFNP $\subseteq$ FNP [Pap94b]. However, it is still open whether these inclusions are strict. Megiddo and Papadimitriou [MP91] have shown that there is an FNP-complete problem in TFNP if and only if NP $=$ coNP.

A polynomial-time reduction from problem A to problem B is a polynomial-time computable function f which maps an instance x of A to an instance $f(x)$ of B, plus another polynomial-time computable function g which maps every solution y of $f(x)$ to a solution $g(y,x)$ of x.

3.3 Complexity Classes of Search Problems

The following subsections introduce several subclasses of TFNP. See Figure 3.1 for their relationship.

3.3.1 Class PLS

The complexity class PLS is the class of all search problems whose totality is proved by local search algorithms. This class was introduced by Johnson, Papadimitriou,

and Yannakakis [JPY88]. The formal definition of this class is defined as the set of all problems which are reducible to the problem LOCALOPT in polynomial time.

Definition 3.5. LOCALOPT
Input:

- two Boolean circuits computing $f : \Sigma^n \to \Sigma^n$ and $p : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$.

Task: Find

- a string x in Σ^n such that $p(x) \geq p(f(x))$.

Definition 3.6 (Class PLS)**.** The complexity class PLS consists of all search problems that are reducible to LOCALOPT in polynomial time.

It is well known that the class PLS contains many essential and fascinating search problems. In particular, the following two problems, MINFLIP and MAXFLIP, are the first PLS-complete problems [JPY88].

Definition 3.7. MINFLIP
Input:

- a Boolean circuit C with n inputs and m outputs that computes a function $f : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$.

Task: Find

- a string x in Σ^n such that for every $y \in \Sigma^n$ with $\|x - y\| = 1$, $C(x) \leq C(y)$

Definition 3.8. MAXFLIP
Input:

- a Boolean circuit C with n inputs and m outputs that computes a function $f : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$.

Task: Find

- a string x in Σ^n such that for every $y \in \Sigma^n$ with $\|x - y\| = 1$, $C(x) \geq C(y)$.

Theorem 3.9 (Johnson, Papadimitriou, and Yannakakis [JPY88])**.** MINFLIP *and* MAXFLIP *are* PLS*-complete.*

The complexity class PLS has been widely studied. It is known that PLS contains some search problems related to a local search method as complete problems. See [Bor16] for a detailed list of PLS-complete problems.

3.3.2 Class PPAD

The complexity class PPAD is the class of all search problems whose totality is guaranteed by the parity argument for digraphs. This class is also introduced by Papadimitriou [Pap94b].

Definition 3.10. END OF LINE
Input:

- two Boolean circuits $S, P : \Sigma^n \to \Sigma^n$
- a standard source $\pi \in \Sigma^n$, i.e., $P(\pi) = \pi \neq S(\pi)$.

Task: Find a vertex x in Σ^n satisfying at least one of the following:

(R1) $P(S(x)) \neq x$, i.e., x is a sink, and

(R2) $S(P(x)) \neq x \neq \pi$, i.e., x is a non-standard source

Definition 3.11 (Class PPAD)**.** The complexity class PPAD consists of all search problems that can be reduced to END OF LINE in polynomial time.

Note that the problem END OF LINE is a total search problem because every finite path has a sink and a source. An input instance attaches a known source, so a sink must exist on the given digraph. Moreover, the correctness of solutions is polynomial-time checkable; END OF LINE is a TFNP problem. These facts imply that the class PPAD is a subclass of TFNP.

For every valid instance of END OF LINE, we are given a digraph $G(S,P)$ with in-degree/out-degree at most one and a standard source $\pi \in \Sigma^n$, i.e., $S(P(\pi)) \neq \pi$. Of course, the principle that guarantees the totality of END OF LINE also works for higher degree digraphs. Every digraph with an unbalanced vertex must have another unbalanced vertex. Here, a vertex is unbalanced if its out-degree and in-degree are different. Such a problem is called IMBALANCE, and Hollender and Goldberg [HG18] have shown that IMBALANCE is polynomially equivalent to END OF LINE. Thus, IMBALANCE is PPAD-complete.

Definition 3.12. IMBALANCE
Input:

- a successor circuit $S : \Sigma^n \to \Sigma^{dn}$
- a predecessor circuit $P : \Sigma^n \to \Sigma^{dn}$
- an unbalanced vertex π in Σ^n such that $\delta^+(\pi) \neq \delta^-(\pi)$.

Task: Find

- another unbalanced vertex x on the implicit digraph $G(S,P)$, i.e., it satisfies that $\delta^+(x) \neq \delta^-(x)$ and $x \neq \pi$.

Theorem 3.13 (Hollender and Goldberg [HG18])**.** IMBALANCE *is* PPAD-*complete.*

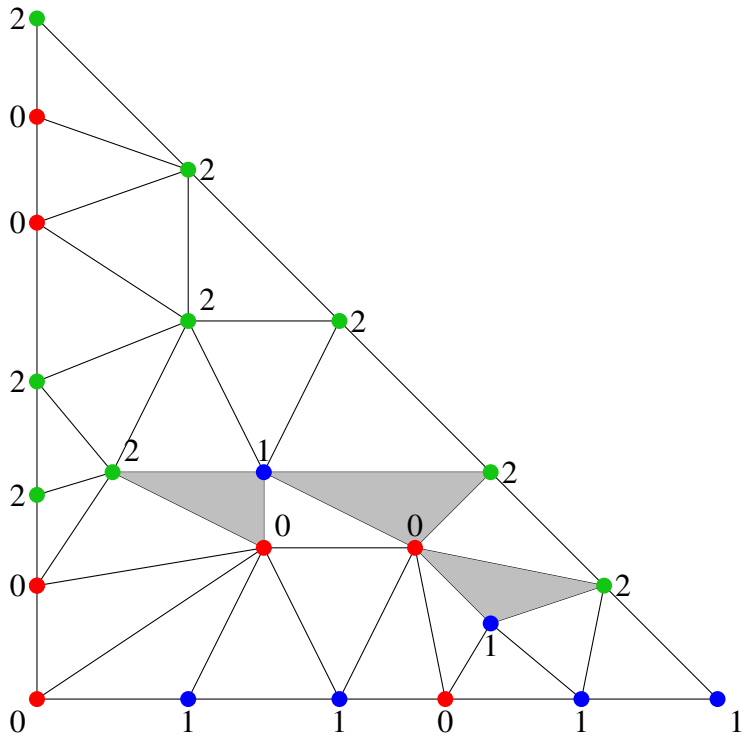

FIGURE 3.2: An Example of Sperner's lemma.

The complexity class PPAD has many complete problems related to various fields (e.g., Fixed Point Theory, Game Theory, and Graph Theory). The first natural PPAD-complete problem is SPERNER, a computational problem based on Sperner's lemma.

Now, we describe the two-dimensional case of Sperner's lemma [Spe28]. Consider a coloring to a triangulation on a triangle; this coloring assigns one of 0, 1, and 2 as a color to each vertex. A coloring is admissible if it satisfies the following conditions: (i) The three vertices on the original triangle have different colors; and (ii) every a-b edge has only a or b, where $a, b \in \{0, 1, 2\}$ and $a \neq b$. Here, we call an edge on the original triangle an a-b edge if its endpoints are a vertex with a and a vertex with b on the original triangle. A small triangle is said to be trichromatic if each vertex has a different color from the other. Sperner [Spe28] has shown that any admissible coloring of any triangulation of a simplex contains at least one trichromatic simplex. We illustrate an example of a two-dimensional admissible coloring in Figure 3.2. Every gray triangle in this figure is trichromatic.

The problem SPERNER is formally defined as follows:

Definition 3.14. SPERNER
Input:

- a Boolean circuit that computes an admissible coloring.

Task:

- Find a trichromatic simplex.

It is not hard to see that SPERNER belongs to TFNP; the existence of solutions is guaranteed by Sperner's lemma [Spe28]. Papadimitriou [Pap94b] proved that

the computational problem based on Sperner's lemma is PPAD-complete even in the three-dimensional setting. Chen and Deng [CD06] have shown that SPERNER is PPAD-complete. Note that solving SPERNER in polynomial time is trivial by using a binary search method for the one-dimensional setting.

Other known PPAD-complete problems include the following: finding a fixed point of Brouwer's and Kakutani's fixed point theorems [Pap94b], computing a variety of equilibria, for example, Nash equilibria [Pap94b], Fisher market equilibria [OPR16], Arrow-Debreu equilibria [Che+09], and approximate competitive equilibrium from equal incomes [OPR16], finding a core of balanced games [Kin+13], finding a stable path [Kin+13], computing a fractional stable hypergraphic matching [Kin+13], and finding a fractional stable kernel [Kin+13].

Remark that the problem END OF LINE can be naturally generalized to the problem on an undirected graph. Such a problem makes up the complexity class PPA which contains PPAD. The next section explains the complexity class PPA.

3.3.3 Class PPA

Papadimitriou [Pap94b] introduced the complexity class PPA. This class consists of all search problems whose totality is guaranteed by the parity argument. Formally, we define the class PPA as the set of all search problems that are reducible to LEAF in polynomial time.

Definition 3.15. LEAF
Input:

- a Boolean circuits $N : \Sigma^n \to \Sigma^{2n}$
- a known leaf $\pi \in \Sigma^n$ on $G(N)$.

Task: Find

- another leaf x on $G(N)$, i.e., it satisfies that $\deg_G(x) = 1$ and $x \neq \pi$.

Definition 3.16 (Class PPA)**.** The complexity class PPA consists of all search problems that are reducible to LEAF in polynomial time.

Let $G(N)$ be an implicit graph given by a valid instance of LEAF. Notice that every vertex on the graph $G(N)$ has a degree at most two. The parity argument guarantees that there must exist an unknown leaf when we have a known leaf. This principle also works for higher degree graphs: A graph that has a known odd-degree vertex must have at least one unknown odd-degree vertex. We consider the search problem ODD, a natural generalization of LEAF. Furthermore, Papadimitriou [Pap94b] has shown that ODD is also a PPA-complete problem.

Definition 3.17. ODD
Input:

- a Boolean circuits $N : \Sigma^n \to \Sigma^{dn}$
- an odd-degree vertex π on $G(N)$.

Task: Find one of the following:

- another odd-degree vertex on $G(N)$, i.e., it satisfies that $x \neq \pi$ and $\deg_G(x) = 2k-1$ for some $k \in \mathbb{N}$.

Theorem 3.18 (Papadimitriou [Pap94b])**.** ODD *is* PPA*-complete.*

The complexity PPA has been widely studied. It is well known that PPA has many complete problems. For example, CONSENSUSHALVING, NECKLACESPLITTING, DISCRETEHAMSANDWICH, and OCTAHEDRALTUCKER [FRG18; FRG19; DFK17; ABB20].

In the previous section, we mentioned that the computational problem SPERNER is PPAD-complete. Deng, Edmonds, Feng, Liu, Qi, and Xu [Den+21] have proven that the problems related to Sperner's lemma on an orientable and a non-orientable two-dimensional spaces are PPAD- and PPA-complete, respectively. They also provided oracle model complexity; both problems require $\Theta(N^{d-1})$ queries.

3.3.4 Class PPP

The pigeonhole principle is a well-known combinatorial existence principle. This principle states that every function $f : [N+1] \to [N]$ must have a collision, i.e., there exist two distinct points $x, y \in [N+1]$ such that $f(x) = f(y)$, where N is a positive integer.

For the computational purpose, we assume that a function f is polynomial-time computable and mapping from Σ^n to Σ^n for some positive integer n. In order to guarantee the totality of the search problem, we usually use the following statement: For every polynomial-time computable function $f : \Sigma^n \to \Sigma^n \setminus \{0^n\}$, there must exist two distinct strings $x, y \in \Sigma^n$ such that $f(x) = f(y)$. However, it is hard to check whether there are strings $x \in \Sigma^n$ such that $f(x) = 0^n$ in polynomial time. Therefore, we consider every string $x \in \Sigma^n$ satisfying that $f(x) = 0^n$ as an additional solution. Such a principle is called a *polynomial pigeonhole principle*.

The class PPP is a class for problems whose totality is guaranteed by the polynomial pigeonhole principle. The formal definition is as follows:

Definition 3.19. PIGEONHOLE CIRCUIT
Input:

- a Boolean circuits $C : \Sigma^n \to \Sigma^n$

Task: Find one of the following:

(R1) two distinct strings $s, y \in \Sigma^n$ such that $C(x) = C(y)$

(V1) a string $x \in \Sigma^n$ such that $C(x) = 0^n$.

Definition 3.20. The complexity class PPP consists of all search problems that are reducible to PIGEONHOLE CIRCUIT in polynomial time.

The complexity class PPP was introduced by Papadimitriou [Pap94b]. It is easy to see that PPP is contained in TFNP.

Papadimitriou [Pap94b] has shown that PPP is closely related to much more familiar issues. For each positive integer n, a one-way permutation is defined as a

polynomial-time function $\pi : \Sigma^n \to \Sigma^n$ satisfying that $\pi(x) \neq \pi(y)$ whenever $x \neq y$, such that for every probabilistic polynomial-time computable function $\tau : \Sigma^n \to \Sigma^n$, every non-negative integer k, and sufficiently large integer n, if we pick a string $w \in \Sigma^n$ at random, then it holds that

$$\Pr[\tau(\pi(w)) = w] \leq n^{-k},$$

where $\Pr[\tau(x) = y]$ means the probability that τ outputs $y \in \Sigma^n$ if it is given a string $x \in \Sigma^n$ as an input (see Section 10.6 of [Sip97]). Roughly speaking, it seems to be required exponential time to compute a string $x \in \Sigma^n$ for a given string $\pi(x)$.

The definition of the PPP implies that there are no one-way permutations if PIGEONHOLE CIRCUIT can be solved in polynomial time.

Theorem 3.21 (Papadimitriou [Pap94b])**.** *If* PPP $=$ FP*, then there are no one-way permutations.*

It is believed that PPP has worked to progress our knowledge of algorithmic aspects of Cryptography, just as PPAD has played a crucial role in advancing our algorithmic understanding of Game Theory [Ban+19].

Ban, Jain, Papadimitriou, Psomas, and Rubinstein [Ban+19] have proven that the problems MINKOWSKI and DIRICHLET belong to PPP, and the problems EQUAL SUMS and DIRICHLET are reducible to MINKWOSKI. Sotiraki, Zampetakis, and Zirdelis [SZZ18] have shown that the problems BLICHFELDT and CSIS are PPP-complete.

There is another complexity class based on the pigeonhole principle, called PWPP (*Polynomial Weak Pigeonhole Principle*). This class is defined as the set of total search problems that are reducible to WEAK PIGEON [Jeř16].

Definition 3.22. WEAK PIGEON
Input:

- a Boolean circuits $C : \Sigma^n \to \Sigma^m$, where $n > m$.

Task: Find one of the following:

- two distinct strings $s, y \in \Sigma^n$ such that $C(x) = C(y)$

Definition 3.23. The complexity class PWPP is a class of search problems that are polynomial-time reducible to WEAK PIGEON.

It is not hard to see that PWPP is contained in PPP. Jeřábek [Jeř16] have shown that FACTORING lies in PPA and PWPP under *randomized* reductions. Komargodski, Naor, and Yogeev [KNY19] have proven that the problem based on Ramsey's theorem, called RAMSEY, is PWPP-hard. Sotiraki, Zampetakis, and Zirdelis [SZZ18] have proven that the problem WC-SIS is PWPP-complete. Recently, Huváček and Václavek [HV21] have shown that the complexity of the discrete logarithm problem, called DLOG, is PWPP-complete.

3.3.5 Class PPAD∩PLS

We describe in this section the complexity class PPAD∩PLS. To define this class, we define the problem EITHER SOLUTION($\mathscr{A},\mathscr{B}$), introduced by Daskalakis and Papadimitriou [DP11] to consider the complexity of the problems belonging to both PPAD and PLS. Furthermore, Daskalakis and Papadimitriou [DP11] defined the complexity class CLS (see Definition 5.4) contained in PPAD∩PLS. Informally speaking, the goal of EITHER SOLUTION($\mathscr{A},\mathscr{B}$) is to find a solution to either $I_{\mathscr{A}}$ or $I_{\mathscr{B}}$ when we are given instances $I_{\mathscr{A}}$ of $\mathscr{A}$ and $I_{\mathscr{B}}$ of $\mathscr{B}$. The formal definition is as follows:

Definition 3.24. EITHER SOLUTION($\mathscr{A},\mathscr{B}$)
Input:

- a valid instance $I_{\mathscr{A}}$ of $\mathscr{A}$ and
- a valid instance $I_{\mathscr{B}}$ of $\mathscr{B}$.

Task: Find either

- a solution to $I_{\mathscr{A}}$ or
- a solution to $I_{\mathscr{B}}$.

Daskalakis and Papadimitriou [DP11] showed that if $\mathscr{A}$ is a PPAD-complete problem and $\mathscr{B}$ is a PLS-complete problem, then the problem EITHER SOLUTION($\mathscr{A},\mathscr{B}$) is PPAD∩PLS-complete. Thus, they proved the following theorem.

Theorem 3.25 (Daskalakis and Papadimitriou [DP11])**.** EITHER SOLUTION(END OF LINE, LOCALOPT) *is a* PPAD∩PLS*-complete problem.*

3.3.6 Class EOPL

The complexity class EOPL is introduced by Fearnley, Gordon, Mehta, and Savani [Fea+20]. This class consists of all search problems that are reducible to END OF POTENTIAL LINE.

Definition 3.26. END OF POTENTIAL LINE
Input:

- two Boolean circuits $S,P:\Sigma^n\to\Sigma^n$
- a Boolean circuit $V:\Sigma^n\to\{0,1,\ldots,2^m-1\}$
- a standard source π in Σ^n such that $P(x)=x\neq S(x)$ and $V(\pi)=0$.

Task: Find a vertex x in Σ^n satisfying at least one of the following:

(R1) $P(S(x))\neq x$, i.e., x is a sink, and

(R2) $S(P(x))\neq x\neq\pi$, i.e., x is a non-standard source

(V1) $S(x)\neq x$, $P(S(x))=x$, and $V(S(x))-V(x)\leq 0$

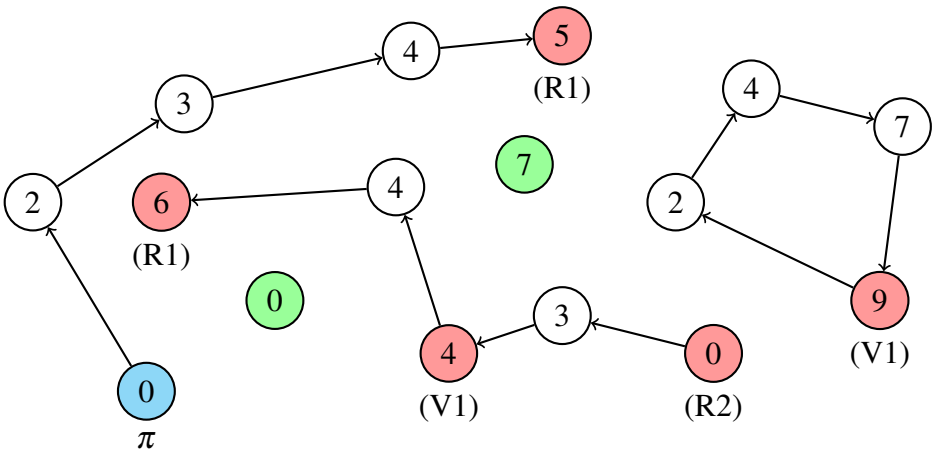

FIGURE 3.3: An example of the instance of END OF POTENTIAL LINE. This figure shows all solutions in red, and the corresponding solution form is labeled. Furthermore, the blue vertex implies the standard source, and it has the label "π." The green vertices are isolated (or self-loops); that is, these are not solutions for END OF POTENTIAL LINE. In this example, we have two lines and one cycle. The mainline starts at the vertex labeled "π" and ends at the red vertex that has weight five and the label (R1). Every valid arc on this line is strictly increasing. Therefore, the mainline does not contain violating solutions. On the other hand, another line starts at the red vertex which was a weight of zero and the label (R2) and ends at the red vertex which has a weight of 6, and the label (R1). This line contains a non-increasing valid arc, and thus, there is a violating solution, that has the label (V1). Finally, every cycle on the instance of END OF POTENTIAL LINE always has a non-increasing arc. This implies that a cycle has a violating solution.

Definition 3.27 (Class EOPL)**.** The complexity class EOPL consists of all search problems that can be reduced to END OF POTENTIAL LINE in polynomial time.

Notice that END OF POTENTIAL LINE has violating solutions. Let (n, m, S, P, V, π) be a valid instance of END OF POTENTIAL LINE, and let $G(S,P) = (\Sigma^n, E)$ be the implicit digraph produced by S and P. Each vertex x on $G(S,P)$ has a potential assigned by the potential function V. Then this problem requires that for every valid arc (x,y) in E, it holds that $V(x) < V(y)$, i.e., every valid arc on $G(S,P)$ is an increasing arc. Therefore, any non-increasing valid arc is a violation. We illustrate an example of END OF POTENTIAL LINE in Figure 3.3.

By definition, it is easy to see that END OF POTENTIAL LINE is polynomial-time reducible to END OF LINE. This implies that EOPL is contained in PPAD. Notice that every valid arc is strictly increasing; we can apply a local search method to find a solution of END OF POTENTIAL LINE, which implies that EOPL also lies on PLS. Therefore, the following statement follows:

Proposition 3.28. $\texttt{EOPL} \subseteq \texttt{PPAD} \cap \texttt{PLS}$.

Surprisingly, Göös, Hollender, Jain, Mayster, Pires, Robere, and Tao [Göö+22] have shown that the inverse reduction follows, that is, EOPL is equal to $\texttt{PPAD} \cap \texttt{PLS}$.

Theorem 3.29 (Göös, Hollender, Jain, Mayster, Pires, Robere, and Tao [Göö+22])**.** $\texttt{EOPL} = \texttt{PPAD} \cap \texttt{PLS}$.

Furthermore, Fearnley, Gordon, Mehta, and Savani [Fea+20] considered the variant of END OF POTENTIAL LINE. They add extra properties to an instance of END OF POTENTIAL LINE. In the original problem, see Definition 3.26, every instance has one form of violating solution, meaning that the promise is unsatisfying.

Definition 3.30. UNIQUE END OF POTENTIAL LINE
Input:

- two Boolean circuits $S, P : \Sigma^n \to \Sigma^n$
- a Boolean circuit $V : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$
- a standard source π in Σ^n such that $P(x) = x \neq S(x)$ and $V(\pi) = 0$.

Task: Find a vertex x in Σ^n satisfying at least one of the following:

(R1) $P(S(x)) \neq x$, i.e., x is a sink,

(V1) $S(P(x)) \neq x \neq \pi$, i.e., x is a non-standard source

(V2) $S(x) \neq x$, $P(S(x)) = x$, and $V(S(x)) - V(x) \leq 0$

(V3) two strings $x, y \in \Sigma^n$ such that $x \neq y$, $x \neq S(x) \neq y$, and either $V(x) = V(y)$ or $V(x) < V(y) < V(S(x))$.

Definition 3.31. The complexity class UNIQUE END OF POTENTIAL LINE is the set of all search problems that are reducible to UNIQUE END OF POTENTIAL LINE in polynomial time.

3.3.7 Further Classes for Search Problems Outside of NP

Hitherto, we deal with the complexity classes inside NP, more precisely TFNP classes. In this section, we briefly mention the classes outside of FNP. First and foremost, Etessami and Yannkakis [EY10] focused on the complexity of search problems that seemed to be not in FNP. They considered the complexity of finding an exact or a strong approximate solution to a real-valued search problem and introduced the class FIXP, which will be discussed in Section 5.3.1.

Here, we consider the complexity class defined by Boolean circuits. First, Kleinberg, Korten, Mitropolsky, and Papadimitriou [Kle+21] introduced a new complexity class called PEPP (*Polynomial Empty Pigeonhole Principle*). The existence of solutions to search problems belonging to PEPP is guaranteed by the pigeonhole principle: For every function $f : [n] \to [n+1]$, there must exist a point x such that $f(y) \neq x$ for every $y \in [n+1]$. The complexity class PEPP is formally defined by the problem EMPTY [Kle+21]:

Definition 3.32. EMPTY
Input:

- a Boolean function $C : \Sigma^n \setminus \{0^n\} \to \Sigma^n$

Task:

- Find a string $y \in \Sigma^n$ such that $C(x) \neq y$ for every $x \in \Sigma^n \setminus \{0^n\}$.

Definition 3.33. The complexity class PEPP is the set of search problems that are polynomial-time reducible to EMPTY.

Naturally, it is unknown how to check the correctness of a solution efficiently. It is not a realistic approach to find an efficient method for verifying the correctness of a solution to EMPTY; this is an FNP-complete problem. Kleinberg, Korten, Mitropolsky, and Papadimitriou [Kle+21] also provided some total search problems belonging to PEPP. For example, REMOTE POINT, REMOTE VERTEX, δ-RIGID MATRIX COMPLETION, and RAMSEY-ERÖS COMPLETION. Furthermore, they also introduced the parameterized analog of PEPP.

Another research line around the outside of FNP is quantum computational complexity. Recently, Massar and Santha [MS21b; MS21a] have studied a quantum counterpart of TFNP. They introduced the complexity class FQMA and TFQMA and have shown some quantum total search problems belonging to TFQMA. However, the following open questions have remained: (i) subclasses that contain complete problems, and (ii) the classes for quantum search problems whose witness can be represented by classical.

In the rest of this section, we discuss the complexity of problems that the correctness of a solution is efficiently checkable by a randomized algorithm. Thus, we consider it a functional analog of the classical Merlin-Arthur model. Informally speaking, we focus on the class of search problems having the following properties:

(a) There is some polynomial p such that the length of a witness for every input instance x is bounded by $p(|x|)$ if it exists;

(b) we have a polynomial-time bounded randomized algorithm to verify the correctness of an outputted solution.

Note that we usually consider *promise problems* when we deal with probabilistic computational models. First, we define an (a,b)-classical verification procedure. Here, let $a,b : \mathbb{Z}_{\geq 0} \to [0,1]$ be polynomial-time computable functions satisfying that $a(n) - b(n) \geq 1/q(n)$ for every $n \in \mathbb{Z}_{\geq 0}$, where q is a strictly positive polynomial. Let M be a polynomial-time bounded probabilistic Turing machine that takes $x \in \Sigma^*$ and $y \in \Sigma^{p(|x|)}$ as inputs, where p is some polynomial. We call M an (a,b)-classical verification procedure if for every input $x \in \Sigma^*$, it holds one of the following:

(i) There exists $y \in \Sigma^{p(|x|)}$ such that $\Pr[M(x,y) = 1] \geq a(|x|)$;

(ii) for every $y \in \Sigma^{p(|x|)}$, $\Pr[M(x,y) = 1] \leq b(|x|)$.

Second, we define *accepting* and *rejecting* subsets. Let M be an (a,b)-classical verification procedure. For an input $x \in \Sigma^*$, the accepting subset

$$R^{\geq a}(x) = \{y \in \Sigma^{p(|x|)} \; ; \; \Pr[M(x,y) = 1] \geq a(|x|) \, ;$$

and the rejecting subset

$$R^{\leq b}(x) = \{y \in \Sigma^{p(|x|)} \; ; \; \Pr[M(x,y) = 1] \leq b(|x|)\}.$$

Finally, we define the complexity class $\mathtt{FMA}(a,b)$ as the class made up of the set $R = \{(R^{\geq a}(x), R^{\leq b}(x))\}$. A search problem with respect to R is formulated as: given an input $x \in \Sigma^*$, find $y \in R^{\geq a}(x)$ if $R^{\geq a}(x) \neq \emptyset$; otherwise report "*no*." The complexity class $\mathtt{TFMA}$ is the subset of $\mathtt{FMA}$ satisfying that $R^{\geq a}(x) \neq \emptyset$ for every input string $x \in \Sigma^*$.

It is not hard to see the following property:

Proposition 3.34. $\mathtt{FMA}(a,b) = \mathtt{FMA}(1,1/2)$.

Proof. Apply the same technique provided by [GZ11; ZF87] to show the perfect completeness of the Merlin-Arthur protocol. □

The complexity class $\mathtt{TF}\Sigma_2^{\mathtt{P}}$, defined by Kleinberg, Korten, Mitropolsky, and Papadimitriou [Kle+21], is the class for $R \subseteq \Sigma^* \times \Sigma^*$ satisfying the followings: There is some polynomial $p : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ such that (i) for every $x \in \Sigma^*$, there exists a string $y \in \Sigma^{p(|x|)}$ such that $(x,y) \in R$; and (ii) there is a polynomial-time Turing machine M such that $(x,y) \in R$ if and only if $M(x,y,z) = 1$ for every $z \in \Sigma^{p(|x|)}$.

Theorem 3.35. *The following property holds:*

(1) $\mathtt{FMA} \subseteq \mathtt{FMA}(a,1/2) \subseteq \mathtt{FMA}(1,<1) \subseteq \mathtt{TF}\Sigma_2^{\mathtt{P}}$.

(2) $\textsc{Empty} \subseteq \mathtt{FMA}(1,<1)$.

Here, the complexity class $\mathtt{FMA}(1,<1)$ *is defined by a* $(1,<1)$*-classical verification procedure.*

Proof. The first property immediately follows from the definition.

We present a $(1,<1)$-classical verification procedure to prove the second result. Let a Boolean circuit $C : \Sigma^n \setminus \{0^n\} \to \Sigma^n$ be an instance of $\textsc{Empty}$. When we are given a string $y \in \Sigma^n$, we execute the $(1,<1)$-classical verification procedure shown in Algorithm 1 to check the correctness of y. It is easy to see that Algorithm 1 verify the correctness of the outputting solution. □

Algorithm 1 A $(1,<1)$-classical verification procedure for $\textsc{Empty}$

```
Pick a string x ∈ Σ^n \ {0^n} uniformly at random.
if Is C(x) ≠ y then
    Return accept
else
    Return reject
end if
```

Finally, we mention the gapped class of $\mathtt{FMA}$. The complexity class $\mathtt{gapFMA}(a,b)$ is the set of search problems in $\mathtt{FMA}(a,b)$ satisfying that $R^{\geq a}(x) \cup R^{\leq b}(x) = \Sigma^{p(|x|)}$ for every $x \in \Sigma^*$. Moreover, the class $\mathtt{gapTFMA}(a,b)$ is the class of search problems in $\mathtt{TFMA}(a,b)$ satisfying that $R^{\geq a}(x) \cup R^{\leq b}(x) = \Sigma^{p(|x|)}$ for every $x \in \Sigma^*$. An interesting open question is that $\mathtt{MA} = \exists\mathtt{BPP}$ when $\mathtt{FMA}(a,b) = \mathtt{gapFMA}(a,b)$.

Chapter 4

On the Complexity of Parity Argument with Potential

4.1 Basics

We are interested in the complexity class TFNP: a set of total search problems belonging to FNP [MP91; Pap94b]. Every problem in TFNP satisfies the following two properties: Every instance always has a solution, and we can verify whether a solution is correct in polynomial time. The class TFNP contains some of the most fundamental, elegant, and intriguing computational problems. The most famous examples are factoring, computing a Nash equilibrium, and finding a local optimum. These problems have no known polynomial-time algorithms, but it seems to be not NP-hard either. In fact, if TFNP has an NP-hard problem, then we have NP = coNP.

Unfortunately, TFNP is a "*semantic*" class. Generally, such classes are unlikely to have complete problems, as are language classes such as RP, ZPP, and BPP [Pap94b]. Thus, we study "*syntactic*" subclass of TFNP. The best-well known such subclasses are PLS, PPP, PPA, and PPAD [Pap94b]. These are classes of search problems whose totality is guaranteed by the corresponding mathematical lemma.

The subclasses of TFNP have been studied productively and extensively and have been shown to have many interesting search problems as complete problems. Section 3.2 gives details of some complexity classes contained in TFNP.

The above subclasses of TFNP are usually defined in terms of the corresponding search problem on an exponentially-large graph, which is represented by functions. Note that a graph in a problem is presented via a function that computes the neighbors of a given vertex on the graph[1], not an adjacency matrix. Hence, we need exponential effort to get the structure of the entire graph.

For example, the class PPA is defined as a set of problems that are reducible to LEAF, in which an undirected graph on 2^n vertices with a degree at most two is presented via a function computing neighbors of a given vertex. It is known that PPA has the following complete problems: CONSENSUS HALVING, NECKLACESPLITTING, DISCRETE HAMSANDWICH, and OCTAHEDRAL TUCKER [FRG18; FRG19; DFK17; ABB20]. Similarly, the class PPAD is defined as a set of problems that are reducible to END OF LINE, in which a digraph on 2^n vertices with in-degree/out-degree at most one is given by two functions that compute a successor and a predecessor of

[1] If a given graph is a digraph, it is represented via two functions, a function that computes outgoing arcs and a function that computes incoming arcs. We call the former a successor function/circuit and the latter a predecessor function/circuit.

a given vertex. PPAD includes a search problem that seems easier than PPA. The most famous PPAD-complete problem is the problem of finding a Nash equilibrium [DGP09; Pap94b].

Fearnley, Gordon, Mehta, and Savani [Fea+20] introduced search problems on digraphs with potential and produced the new computational complexity class EOPL (see Definition 3.27). Recently, Göös, Hollender, Jain, Maystre, Pires, Robere, and Tao [Göö+22] have shown a surprising collapse: $\mathtt{EOPL} = \mathtt{PPAD} \cap \mathtt{PLS}$. It is known that EOPL contains some fascinating total search problems, e.g., solving the Linear Complementarity Problem for P-matrices, solving parity games, and solving simple stochastic games [Fea+20].

As mentioned in Section 3.3.6, the canonical EOPL-complete problem can be viewed as a weighted relaxation of a PPAD problem. A natural question worth considering is: How easier the problem is by adding a potential condition to a problem known as a PPA-complete problem.

Recently, Hollender and Goldberg [HG18] have studied the robustness of the classification by END OF LINE, a problem of finding a sink or an unknown source when we are given a successor circuit, a predecessor circuit, and one standard source, find a sink or a non-standard source. The problem END OF LINE characterizes the class PPAD. Thus, this problem is a canonical PPAD-complete problem. Hollender and Goldberg [HG18] considered combinatorial principles related to PPAD, leading to the following problems, on digraphs with a degree at most two:

- given k sources and $l \neq k$ sinks, find another sink or source;
- given k sources and $l < k$ sinks, find $k - l$ other sinks; and
- given k sources, find k sinks or k other sources.

They proved that these above problems are also PPAD-complete. Moreover, they showed that the problem IMBALANCE, in which given a digraph and an unbalanced vertex, i.e., a vertex with in-degree $\neq$ out-degree, find another unbalanced vertex, is also PPAD-complete. These facts imply that the classification by END OF LINE is very robust.

Hollender and Goldberg [HG18] left an open question: Is a multiple-source variant of END OF POTENTIAL LINE also EOPL-complete? We resolve this question in this thesis, and we show that the definition of the class EOPL based on END OF POTENTIAL LINE is robust.

In this thesis, we extend their results to END OF POTENTIAL LINE. In this problem, given a successor circuit, a predecessor circuit, a potential function, and one standard source, the objective is to find one of a sink, a non-standard source, and a non-increasing arc. We first consider combinatorial principles related to EOPL, leading to the following problems, on the graphs with potential with a degree of at most two:

- given k sources, find another degree-1 vertex or a non-increasing arc; and
- given k sources, find k distinct vertices that are at least one of a sink, other source, and a non-increasing arc.

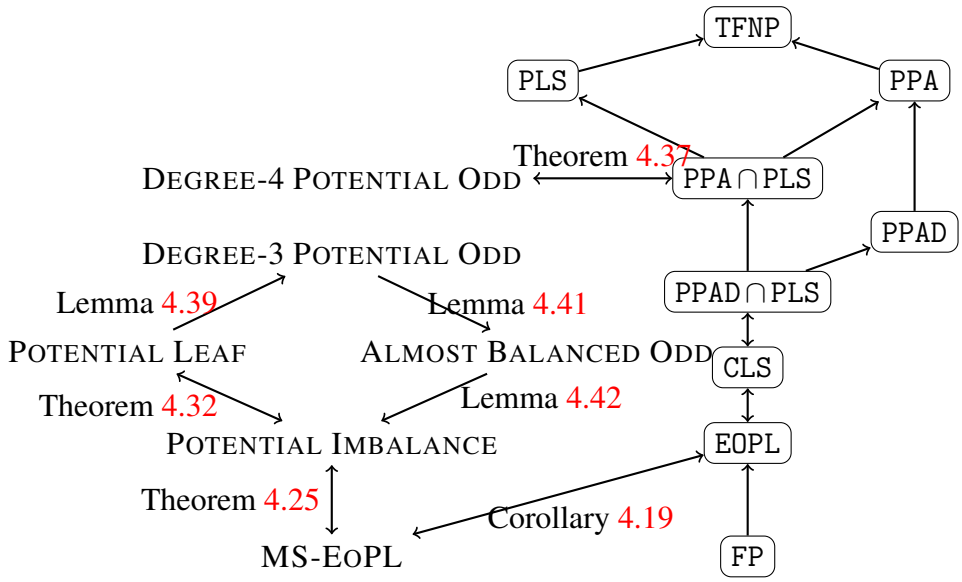

FIGURE 4.1: The relationship of search problems and complexity classes.

We show that these variants of END OF POTENTIAL LINE can be classified in terms of the original problem; that is, these problems are also EOPL-complete. Furthermore, we consider the problem of generalizing END OF POTENTIAL LINE to higher degree digraphs. In this problem, given a digraph with potential and one unbalanced vertex, the objective is to find another unbalanced vertex or a non-increasing arc. Naturally, this problem also belongs to EOPL.

Finally, we prove that the problem, in which we are given a graph with potential on $[2^n]$ with a degree at most three and an odd-degree vertex, and the goal is to seek another odd-degree or a local optimum vertex, is also EOPL-complete. However, even if the problem for a given graph with potential and the maximum degree at most four is not always EOPL-complete. Probably, such problems are much harder than END OF POTENTIAL LINE.

4.1.1 Our Contribution

An overview of our results and the known relationship of complexity classes is depicted in Figure 4.1. In this figure, each arrow $\alpha \to \beta$ denotes that there is a polynomial-time reduction from α to β.

In Section 4.2.1, we introduce the notion called the normalization to simplify arguments in this thesis. Some search problems require that every instance has several properties. The best-known example of such a problem is BROUWER. In this problem, the function given by the instance is required Lipschitz continuous. Generally, it seems hard to decide whether a given function is Lipschitz continuous. However, given a witness, it is easy to check it. We often add every witness as a solution called a violation. Informally speaking, our normalization is to transform from an instance that has violations to another instance satisfying the required conditions. The formal definition is given in Section 4.2.1.

Directed Graphs with Potential In Section 4.3, we extend the elegant results by Hollender and Goldberg [HG18] to END OF POTENTIAL LINE, and we show the robustness of EOPL. We introduce the new variant of END OF POTENTIAL LINE. We call this problem MULTIPLE-SOURCE END OF POTENTIAL LINE. We prove that this problem is also EOPL-complete. Furthermore, we introduce the new problem of generalizing END OF POTENTIAL LINE to higher degree digraphs, which is called POTENTIAL IMBALANCE. We also prove that this problem is EOPL-complete.

Undirected Graphs with Potential In Section 4.4, to study the hardness of the parity argument with potential, we introduce the complexity class PPA$\cap$PLS. This class consists of all search problems belonging to both PPA and PLS (see Section 4.2.3). We define POTENTIAL ODD; this problem is a generalization of POTENTIAL IMBALANCE to an undirected graph with potential. We show that POTENTIAL ODD is, generally, PPA$\cap$PLS-complete. Specifically, if the maximum degree on the given graph is at most three, then POTENTIAL ODD belongs to EOPL, i.e., this problem is EOPL-complete. However, even if the maximum degree on the graph is four, then POTENTIAL ODD is PPA$\cap$PLS-complete.

4.2 Preliminaries

Generally, every search problem has a solution set $\mathscr{O}(\mathscr{I})$ for each instance $\mathscr{I}$. In this thesis, we consider that every problem has two types of solutions (possibly empty). One is called a regular solution, and the other is called violating a solution. Some search problems require their instance to satisfy some conditions. While it is easy to decide whether a given instance is valid, it is hard to check whether a given instance satisfies the required conditions most of the time. However, it is effortless to verify whether a given instance violates the required conditions if we obtain a witness. We often add a witness as a solution, and such a solution is called a violating solution. On the other hand, a regular solution means a desired solution to the search problem originally. For instance, CONTINUOUS LOCALOPT (see Definition 5.3) requires that two functions f and p are λ-Lipschitz continuous. It seems to be hard that we efficiently verify whether f and p are λ-Lipschitz continuous. Therefore, we treat the violation as a solution. Note that this configuration ensures the totality of CONTINUOUS LOCALOPT.

Formally, when we are given an instance $\mathscr{I}$ of the search problem R that has violating solutions, the solution set $\mathscr{O}(\mathscr{I})$ to $\mathscr{I}$ is denoted as follows.

$$\mathscr{O}(\mathscr{I}) = \mathscr{O}_R(\mathscr{I}) \cup \mathscr{O}_V(\mathscr{I}),$$

where $\mathscr{O}_R(\mathscr{I})$ is a set of all regular solutions and $\mathscr{O}_V(\mathscr{I})$ is a set of all violating solutions. For instance, the set of solutions for CONTINUOUS LOCALOPT consists of all ε-approximate local optima and all witnesses of λ-Lipschitz continuity for given functions. The set $\mathscr{O}_R(\cdot)$ contains all ε-approximate local optima, and the set $\mathscr{O}_V(\cdot)$ contains all witnesses of non-λ-Lipschitz continuity for given functions.

4.2.1 Normalization

In this section, we introduce a normalization of search problems. Informally speaking, a normalization of a search problem that has both regular solutions and violating solutions is defined as a transformation from a given valid instance to another valid instance that satisfies the required conditions. Formally, it is defined as follows.

Definition 4.1 (Normalization)**.** A search problem R has both regular solutions and violating solutions. A normalization of the problem R is a polynomial-time computable function f which maps an instance $\mathscr{I}$ of R to another instance $f(\mathscr{I})$ of R which has no violations, i.e., $\mathscr{O}_V(f(\mathscr{I}))$ is empty, plus another polynomial-time computable function g which maps every solution y of $f(\mathscr{I})$ to a solution $g(y,\mathscr{I})$ of $\mathscr{I}$.

The rest of this section gives us an example of a normalizable search problem. We show that END OF POTENTIAL LINE can be normalized.

Proposition 4.2. END OF POTENTIAL LINE *can be normalized.*

Proof. Let $\mathscr{I} = (n,m,S,P,V,\pi)$ be a valid instance of END OF POTENTIAL LINE. Without loss of generality, we assume that the standard source π is not a solution of $\mathscr{I}$ because we can check it in polynomial time. Now, we construct another instance $\mathscr{J} = (n,m,S',P',V',\pi)$ that has no violations.

First, we construct a successor circuit $\bar{S}$ and a predecessor circuit $\bar{P}$ such that they always compute valid arcs or self-loops. For each vertex x in Σ^n, we define

$$\bar{S}(x) := \begin{cases} S(x) & \text{if } S(x) \neq x \text{ and } P(S(x)) = x, \\ x & \text{otherwise,} \end{cases}$$

and

$$\bar{P}(x) := \begin{cases} P(x) & \text{if } P(x) \neq x \text{ and } S(P(x)) = x, \\ x & \text{otherwise.} \end{cases}$$

From the above construction, if $y = \bar{S}(x) \neq x$, then $\bar{P}(y) = x$ for each $x \in \Sigma^n$, similarly, if $y = \bar{P}(x) \neq x$, then $\bar{S}(y) = x$ for each $x \in \Sigma^n$.

Next, we construct a successor circuit S' and a predecessor circuit P'. For each vertex x in Σ^n, we define

$$S'(x) := \begin{cases} \bar{S}(x) & \text{if } \bar{S}(x) \neq x \text{ and } V(\bar{S}(x)) > V(x), \\ x & \text{otherwise,} \end{cases}$$

and

$$P'(x) := \begin{cases} \bar{P}(x) & \text{if } \bar{P}(x) \neq x \text{ and } V(\bar{P}(x)) < V(x), \\ x & \text{otherwise.} \end{cases}$$

By the definitions of S' and P', if $S'(x) \neq x$, then $S'(x) = \bar{S}(x) \neq x$ and $V(\bar{S}(x)) > V(x)$. This implies that it satisfies that $\bar{P}(\bar{S}(x)) = x$. Since $V(S'(x)) = V(\bar{S}(x)) > V(x)$, it

holds that $P'(S'(x)) = P'(\bar{S}(x)) = \bar{P}(\bar{S}(x)) = x$. Similarly, if $P'(x) \neq x$, then it holds that $S'(P'(x)) = x$ and $V(x) > V(\bar{P}(x)) = V(P'(x))$. Therefore, every valid arc with respect to S' and P' is always a strictly increasing arc.

Finally, we define the new potential function V' as follows.

$$V'(x) := \begin{cases} 0 & \text{if } S'(x) = x = P'(x), \\ V(x) & \text{otherwise.} \end{cases}$$

From the above constructions, it is easy to see that the standard source π in Σ^n satisfies that $P'(\pi) = \pi \neq S'(x)$ and $V'(\pi) = 0$. Hence, the tuple $\mathscr{J} = (n, m, S', P', V', \pi)$ is a valid instance of END OF POTENTIAL LINE. Furthermore, it is not hard to see that $\mathscr{J}$ has no violating solutions. In the rest of this proof, we show that when we obtain a solution of $\mathscr{J}$, we can convert it to a solution of $\mathscr{I}$ in polynomial time.

We first consider when we obtain a solution x such that $P'(S'(x)) \neq x$. Then it must satisfy that $S'(x) = x$. Therefore, it holds that $P'(S'(x)) = P'(x) \neq x$. If $\bar{S}(x) \neq x$, then $V(\bar{S}(x)) \leq V(x)$ by the definition of S'. Hence, x is a violating solution for the original instance since $V(\bar{S}(x)) = V(S(x))$. On the other hand, if $\bar{S}(x) = x$, then $\bar{P}(\bar{S}(x)) \neq x$ since $P'(x) \neq x$. This implies that x satisfies $P(S(x)) \neq x$, and thus, x is a sink of the original instance.

Next, we consider when we obtain a solution x such that $S'(P'(x)) \neq x \neq \pi$. Then it must satisfy that $P'(x) = x$. Therefore, it holds that $S'(P'(x)) = S'(x) \neq x$. If $\bar{P}(x) \neq x$, then $V(x) \leq V(\bar{P}(x))$. Therefore, it satisfies that $S(P(x)) \neq P(x)$, $P(S(P(x))) = P(x)$, and $V(x) \leq V(P(x))$ since $\bar{P}(x) = P(x) \neq x$. This implies that the vertex $\bar{P}(x)$ is a violating solution for the original instance. On the other hand, if $\bar{P}(x) = x$, then $\bar{S}(\bar{P}(x)) \neq x$. Therefore, it holds that $S(P(x)) \neq x \neq \pi$. This implies that x is a non-standard source of the original instance.

From the above arguments, we complete the normalization of END OF POTENTIAL LINE. □

By definition, any normalization is polynomial-time computable. Without loss of generality, we assume that every given instance is already normalized if it can be normalized.

Applying Proposition 4.2, without loss of generality, we assume that every instance of END OF POTENTIAL LINE satisfies the following properties:

- if $S(x) \neq x$, then $P(S(x)) = x$ and $V(S(x)) > V(x)$;
- if $P(x) \neq x$, then $S(P(x)) = x$ and $V(x) > V(P(x))$.

4.2.2 The Problem: EITHER SOLUTION($\mathscr{A}, \mathscr{B}$)

In this section, we describe the problem EITHER SOLUTION($\mathscr{A}, \mathscr{B}$) to formulate the complexity class PPA∩PLS (see Section 4.2.3 for details). This search problem is introduced by Daskalakis and Papadimitriou [DP11] to consider the complexity of the class PPAD∩PLS[2]. Informally speaking, in this problem, given two instances that

[2]The complexity class PPAD∩PLS is defined as a set of all search problems belonging to both PPAD and PLS.

one is an instance of $\mathscr{A}$, and the other is an instance of $\mathscr{B}$, then find a solution for either $\mathscr{A}$ or $\mathscr{B}$.

Daskalakis and Papadimitriou [DP11] have shown that if $\mathscr{A}$ is a PPAD-complete problem and $\mathscr{B}$ is a PLS-complete problem, then the problem EITHER SOLUTION($\mathscr{A}$, $\mathscr{B}$) is PPAD$\cap$PLS-complete, and thus, they proved the following theorem.

Theorem 4.3 (Daskalakis and Papadimitriou [DP11])**.** EITHER SOLUTION(END OF LINE, LOCALOPT) *is a* PPAD$\cap$PLS*-complete problem.*

Now, we present a more general statement than the result of Daskalakis and Papadimitriou [DP11] mentioned above. Specifically, we show that the problem EITHER SOLUTION($\mathscr{A}$,$\mathscr{B}$) is an $\mathscr{C}_A \cap \mathscr{C}_B$-complete problem if A is $\mathscr{C}_A$-complete, and B is $\mathscr{C}_B$-complete.

Theorem 4.4. *Let $\mathscr{C}_A$ and $\mathscr{C}_B$ be subclasses of* FNP *which have complete problems.* EITHER SOLUTION(A, B) *is $\mathscr{C}_A \cap \mathscr{C}_B$-complete if A is a $\mathscr{C}_A$-complete problem, and B is a $\mathscr{C}_B$-complete problem.*

Proof. First, we show that EITHER SOLUTION(A, B) belongs to both $\mathscr{C}_A$ and $\mathscr{C}_B$. We assume that we are given a valid instance $\mathscr{I}_A$ of A and a valid instance $\mathscr{I}_B$ of B. Immediately, the problem A belongs to $\mathscr{C}_A$; we can obtain a polynomial-time reduction to A. Similarly, we can construct a polynomial-time reduction to B. Therefore, EITHER SOLUTION(A, B) is contained in $\mathscr{C}_A$ and $\mathscr{C}_B$.

To show the hardness, we consider any problem C belonging to $\mathscr{C}_A \cap \mathscr{C}_B$. There is a polynomial-time reduction f from C to A since A is a $\mathscr{C}_A$-complete problem. Thus, we can generate a valid instance $f(x)$ of A from a given instance of C. Similarly, there is a polynomial-time reduction g from C to B, that is, we can produce a valid instance $g(x)$ of B from a given instance of C. Hence, the tuple $(f(x), g(x))$ is a valid instance of EITHER SOLUTION(A, B), that is, we have a polynomial-time reduction from C to EITHER SOLUTION(A, B). □

From Theorem 4.4, the following statement straightforwardly follows.

Lemma 4.5. EITHER SOLUTION(MAXFLIP, MINFLIP) *is* PLS*-complete.*

In the problem EITHER SOLUTION(MAXFLIP, MINFLIP), when we are given two Boolean circuits C and D, we find either a local maximum of C or a local minimum of D. A natural question arises. Next, we consider the complexity when $C = D$. That is, we consider the problem, which is called FLIP, of finding a local maximum or a local minimum for a given Boolean circuit C. Furthermore, we show that FLIP is also PLS-complete. The formal definition of FLIP is as follows.

Definition 4.6. FLIP
Input:

- a Boolean circuit C with n inputs and m outputs computing a function $f : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$.

Task: Find a bit-string $x \in \Sigma^n$ satisfying at least one of the following:

(R1) $C(x) \geq C(y)$ for every $y \in \Sigma^n$ with $\|x - y\| = 1$, and

(R2) $C(x) \leq C(y)$ for every $y \in \Sigma^n$ with $\|x-y\| = 1$.

Theorem 4.7. FLIP *is a* PLS*-complete problem.*

Proof. Assume that two Boolean circuits C and D as an instance of EITHER SOLUTION(MAXFLIP, MINFLIP), where C and D compute functions $f : \Sigma^{n_1} \to \{0,1,\ldots,2^{m_1}-1\}$ and $g : \Sigma^{n_2} \to \{0,1,\ldots,2^{m_2}-1\}$, respectively. Furthermore, the Boolean circuit C with n_1 inputs and m_1 outputs is an instance of MAXFLIP, and the Boolean circuit D with n_2 inputs and m_2 outputs is an instance of MINFLIP.

Now, we define the new Boolean circuit $E : \Sigma^{n_1+n_2} \to \Sigma^{m_1+m_2}$ as follows. We consider the first n_1-bits of input x in $\Sigma^{n_1+n_2}$ to E as input to C, and the remaining n_2-bits as input to D. Furthermore, we define the first n_1-bits as x_1 and the remaining n_2-bits as x_2, and thus, denote x as (x_1,x_2). Similarly, we denote y in $\Sigma^{m_1+m_2}$ as (y_1,y_2), where y_1 is the first m_1-bits of y and y_2 is the remaining m_2-bits of y. Then for every bit-string $x = (x_1,x_2)$ in $\Sigma^{n_1+n_2}$, we define as $E(x_1,x_2) = (C(x_1),D(x_2))$. In particular, the circuit E computes $2^{m_2}f(x_1)+g(x_2)$ for every bit-string $x = (x_1,x_2)$ in $\Sigma^{n_1+n_2}$. It is easy to see that E is polynomial-time computable. Here, the Boolean circuit E is a valid instance of FLIP.

What remains is to show that we can obtain an original solution from a solution for E. In the rest of this proof, we denote by $N(x_1,x_2) = N_1(x_1,x_2) \cup N_2(x_1,x_2)$ the neighborhood of $x = (x_1,x_2) \in \Sigma^{n_1+n_2}$, where $N_1(x_1,x_2) = \{(y_1,x_2) \in \Sigma^{n_1+n_2}; \|x_1 - y_1\| = 1\}$ and $N_2(x_1,x_2) = \{(x_1,y_2) \in \Sigma^{n_1+n_2}; \|x_2 - y_2\| = 1\}$. Here, the function N_1 outputs the neighbors of a given bit-string x_1 in Σ^{n_1}, and the function N_2 outputs the neighbors of a given bit-string x_2 in Σ^{n_2}. Notice that for every bit-string $x = (x_1,x_2)$ in $\Sigma^{n_1+n_2}$, the neighbors of x agree with the union of $N_1(x_1)$ and $N_2(x_2)$.

First, we consider when we obtain a local maximum solution $(x_1,x_2) \in \Sigma^{m_1+m_2}$ for E, i.e., it satisfies that $E(x_1,x_2) \geq E(y_1,y_2)$ for every $(y_1,y_2) \in N(x_1,x_2)$. Then x_1 is a local maximum solution for C because it holds that $C(x_1) \geq C(y_1)$ for every $y_1 \in N(x_1,x_2)$. Second, we consider when we obtain a local minimum solution $(x_1,x_2) \in \Sigma^{m_1+m_2}$ for E, i.e., it satisfies that $E(x_1,x_2) \leq E(y_1,y_2)$ for every $(y_1,y_2) \in N(x_1,x_2)$. Then x_2 is a local minimum solution for D because it holds that $D(x_2) \leq D(y_2)$ for every $y_2 \in N(x_1,x_2)$. Hence, we can extract a solution of the original instance from a solution of the new instance.

From the above arguments, we complete the polynomial-time reduction from EITHER SOLUTION(MAXFLIP, MINFLIP) to FLIP. Therefore, the theorem follows. □

4.2.3 Class PPA∩PLS

In this section, we introduce the computational complexity class PPA∩PLS. This class is defined as a set of all search problems that are contained in PPA and PLS.

Definition 4.8 (Class PPA∩PLS)**.** The complexity class PPA∩PLS consists of all search problems that belong to both PPA and PLS.

Clearly, this class has following relationship with other well-known classes:

(a) EOPL ⊆ PPAD∩PLS ⊆ PPA∩PLS.

(b) PPA∩PLS ⊆ PPA.

(c) $\mathtt{PPA} \cap \mathtt{PLS} \subseteq \mathtt{PLS}$.

(d) $\mathtt{PPAD} \subseteq \mathtt{PPA} \cap \mathtt{PLS}$ if and only if $\mathtt{PPAD} \subseteq \mathtt{PLS}$.

Furthermore, the next theorem straightforwardly follows from Theorem 4.4.

Theorem 4.9. EITHER SOLUTION(ODD, FLIP) *is* $\mathtt{PPA} \cap \mathtt{PLS}$*-complete.*

4.3 Multi Source Problems

In this section, we discuss the new variants of END OF POTENTIAL LINE. The most typical modification worth considering is perhaps the following: what if the implicit digraph associated END OF POTENTIAL LINE has two or more standard sources instead of one. The objective remains the same: Find either a sink, a non-standard source, or a non-increasing arc. The existence of at least two standard sources implies that there must exist at least two sinks. Such a problem has more candidate solutions than the original problem. Hence, a new problem might seem more effortless.

As mentioned in the introduction, Hollender and Goldberg [HG18] studied and showed that the multiple-source variants of END OF LINE also belong to PPAD. Surprisingly, these problems are PPAD-complete. Thus, the classification by END OF LINE is robust. The natural question worth considering is whether a similar statement holds for END OF POTENTIAL LINE. Throughout this section, we prove that this claim is mostly correct.

4.3.1 The Problem: kS-EOPL

We consider the following variant of END OF POTENTIAL LINE that the goal is to find one solution when we are given k standard sources, where k is some constant.

Definition 4.10. kS-END OF POTENTIAL LINE (abbreviated kS-EOPL)
Input:

- two Boolean circuits $S, P : \Sigma^n \to \Sigma^n$
- a Boolean circuit $V : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$
- a set of k standard sources $\Pi = \{\pi_1, \ldots, \pi_k\}$ such that $P(\pi) = \pi \neq S(\pi)$ and $V(\pi) = 0$ for each π in Π.

Task: Find a vertex x in Σ^n satisfying at least one of the following:

(R1) $P(S(x)) \neq x$, i.e., x is a sink

(R2) $S(P(x)) \neq x \notin \Pi$, and

(V1) $S(x) \neq x$, $P(S(x)) = x$, and $V(S(x)) - V(x) \leq 0$

Immediately, we can see that kS-EOPL is EOPL-hard. For every given instance of END OF POTENTIAL LINE, we create k copies of it. Therefore, the following lemma straightforwardly follows.

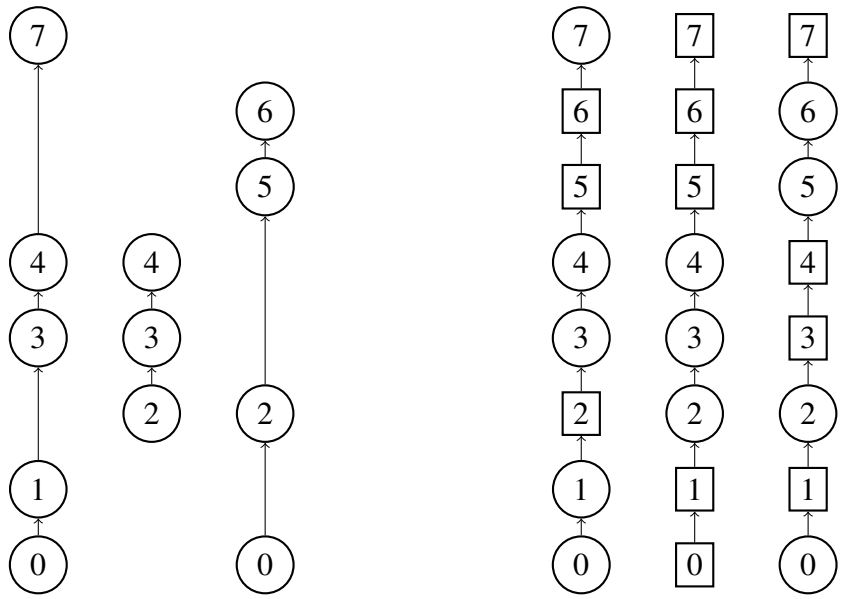

FIGURE 4.2: The left digraph is the original instance. The right digraph is a metered line based on the left digraph. A vertex of a rectangle represents a dummy. Note that each source and each sink on the right digraph have the same potential 0 and 7, respectively.

Lemma 4.11. *kS-EOPL is* EOPL*-hard.*

Next, we prove that kS-EOPL belongs to EOPL. To prove this, we construct a polynomial-time reduction from kS-EOPL to END OF POTENTIAL LINE. Our reduction is via an instance of END OF METERED LINE as an intermediate instance. The problem END OF METERED LINE is introduced by Hubáček and Yogev [HY17]. Every END OF METERED LINE instance has an excellent property: the potential increase exactly one along each valid arc. In other words, the potential on a vertex means the distance from some source. This property plays a crucial role in our reduction.

To construct a polynomial-time reduction from kS-EOPL to END OF POTENTIAL LINE, we have to settle the following two issues:

(1) the bundling of k standard sources into one standard source;

(2) computing the potential of each point efficiently.

In the multiple source variant of END OF LINE, we only had to resolve the issue (1). Hollender and Goldberg [HG18] resolved this problem and showed the containment of MULTIPLE-SOURCE ENDOFLINE in PPAD. They bundled k standard sources into one source by regarding a set of some vertices as one vertex. Recall that the digraph given by an instance has an exponential number of vertices; it seems that we can not efficiently determine whether vertices that make up a new vertex are on the same path. Furthermore, not all paths have the same length. For example, look at the path on the left of Figure 4.2; the length is different for each path. To settle this problem, Hollender and Goldberg [HG18] simulated k paths by retracing the paths sometimes backward. However, their exciting technique did not establish that we can efficiently compute the potential because the turn back the path became an obstacle.

The straightforward way to settle the above two issues (1) and (2) is to make all paths the same length and ensure each vertex that makes up a new vertex is always at the same distance from its source. It is possible in the kS-EOPL because the instance has a potential function. Recall that every instance of END OF METERED LINE has an excellent property: The potential increase exactly one along each path. Fearnley, Gordon, Mehta, and Savani [Fea+20] gave us a polynomial-time reduction from END OF POTENTIAL LINE to END OF METERED LINE. We realize a polynomial-time reduction from kS-EOPL to END OF POTENTIAL LINE via an END OF METERED LINE instance. Moreover, we add some additional vertices to make the length of all paths the same.

The outline of our reduction is as follows. This reduction consists of two steps. In the first step, we construct an intermediate instance like END OF METERED LINE, see Figure 4.2. In the second step, we create a valid instance of END OF POTENTIAL LINE from the intermediate instance. We regard a list of k distinct vertices as a single vertex. Specifically, all k vertices that make up a valid vertex on the END OF POTENTIAL LINE instance have the same potential. This configuration allows us to compute the potential of the new vertex efficiently. The new standard source is the set of all k standard sources of the intermediate instance. This completes the illustration of our reduction from kS-EOPL to END OF POTENTIAL LINE. We can efficiently extract a solution from a solution to the new instance. Details are given in the proof of Lemma 4.12.

Lemma 4.12. kS-EOPL *is reducible to* END OF POTENTIAL LINE *in polynomial time.*

Proof. Let $\mathscr{I} = (n, m, S, P, V, \Pi)$ be a valid instance of kS-EOPL. Applying Proposition 4.2, we assume that this instance is already normalized without loss of generality, that is, $\mathscr{I}$ satisfies the following conditions:

- if $S(x) \neq x$ then $P(S(x)) = x$ and $V(S(x)) > V(x)$;
- if $P(x) \neq x$ then $S(P(x)) = x$ and $V(x) > V(P(x))$.

In this proof, we denote the set of k distinct standard sources as $\Pi = \{\pi_1, \pi_2, \ldots, \pi_k\}$.

First, we construct an intermediate instance $\mathscr{I}_M = (2n+h, h, S', P', V', \Pi')$, where $h = 2n+m$. Let $\mathscr{U} := \Sigma^n \times \Sigma^n \times \{0, 1, \ldots, 2^h - 1\}$ be the new vertex set. The intermediate instance $\mathscr{I}_M$ is also an valid instance of kS-EOPL satisfying the following properties:

- if $S'(u) \neq u$, then $P'(S'(u)) = u$ and $V'(S'(u)) = V'(u) + 1$;
- if $P'(u) \neq u$, then $S'(P'(u)) = u$, and $V'(u) = V'(P'(u)) + 1$;
- every self-loop vertex u, i.e., $S'(u) = u = P'(u)$ has $V'(u) = 0$;

where u is a vertex in $\mathscr{U}$. Furthermore, we can efficiently convert a solution to $\mathscr{I}$ from a solution to $\mathscr{I}_M$.

We show the algorithms of the successor circuit $S' : \mathscr{U} \to \mathscr{U}$ and the predecessor circuit $P' : \mathscr{U} \to \mathscr{U}$, in Algorithm 2, and Algorithm 3, respectively. Our algorithm is based on the reduction from END OF POTENTIAL LINE to END OF METERED

Algorithm 2 Compute $S'(x,y,z)$

1: If $x = y$ then
   1. if $S(x) \neq x$ and $P(S(x)) = x$ then
      (a) if $V(S(x)) - 1 > V(x) = z$ then return $(x, S(x), z+1)$,
      (b) if $V(S(x)) - 1 = V(x) = z$ then return $(S(x), S(x), z+1)$,
   2. if $S(P(x)) \neq x$, i.e., x is a source, then
      (a) if $z < V(x)$ then return $(x, x, z+1)$.
   3. if $P(S(x)) \neq x$, i.e., x is a sink, then
      (a) if $2^h - 1 > z \geq V(x)$ then return $(x, x, z+1)$.

2: If $x \neq y$ then
   1. if $S(x) = y$, $P(y) = x$, and $V(x) < V(y)$ then
      (a) if $V(x) + 1 \leq z < V(y) - 1$ then return $(x, y, z+1)$,
      (b) if $V(x) < z = V(y) - 1$ then return $(y, y, z+1)$.

3: In all other cases, then return (x, y, z).

Algorithm 3 Compute $P'(x,y,z)$

1: If $x = y$ then
   1. if $P(y) \neq y$ and $S(P(y)) = y$ then
      (a) if $V(P(y)) + 1 < V(y) = z$ then return $(P(y), y, z-1)$,
      (b) if $V(P(y)) + 1 = V(y) = z$ then return $(P(y), P(y), z-1)$,
   2. if $S(P(y)) \neq y$, i.e., x is a source, then
      (a) if $V(y) \geq z > 0$ then return $(y, y, z-1)$.
   3. if $P(S(y)) \neq y$, i.e., x is a sink, then
      (a) if $V(y) < z$ then return $(y, y, z-1)$

2: If $x \neq y$, $S(x) = y$, and $P(y) = x$ then
   1. if $V(x) + 1 < z \leq V(y) - 1$ then return $(x, y, z-1)$,
   2. if $V(x) + 1 = z < V(y)$ then return $(x, x, z-1)$.

3: In all other cases, then return (x, y, z).

LINE by Fearnley, Gordon, Mehta, and Savani [Fea+20]. Here, we add additional processing to make the length of all paths the same.

Finally, for each vertex $(x,y,z) \in \mathscr{U}$, we define

$$V'(u) := \begin{cases} 0 & \text{if } S'(u) = (u) = P'(u), \\ z & \text{otherwise.} \end{cases}$$

It is easy to see that the circuits S', P', and V' are constructed in polynomial time, and they can be polynomial-time computable. Now, we denote the new set of k distinct standard sources by $\Pi' := \{(\pi_i, \pi_i, 0) \in \mathscr{U}; i = 1,2,\ldots,k\}$. We complete the construction of the intermediate instance $\mathscr{I}_M$.

Before completing the first step, we verify that we can extract a solution to $\mathscr{I}$ from a solution to $\mathscr{I}_M$ in polynomial time. First, we show that each valid arc on S' and P' is increased by exactly one.

Claim 4.13. *Let $u = (x,y,z) \in \mathscr{U}$. If $S'(u) \neq u$, then it satisfies that $P'(S'(u)) = u$ and $V'(S'(u)) = V'(u) + 1$. Similarly, if $P'(u) \neq u$, then it satisfies that $S'(P'(u)) = u$ and $V'(u) = V'(P'(u)) + 1$.*

Proof. It is clear from our construction. If a vertex $u = (x,y,z)$ satisfies that $S'(u) \neq u$, then $V'(u) = z$. Therefore, we have $V'(S'(u)) = z + 1$. Similarly, it holds that $V'(u) = V'(P'(u)) + 1$ when $P'(u) \neq u$. □

Next, we show that if a vertex $u = (x,y,z) \in \mathscr{U}$ is a sink or a source, $x = y$ holds. The next claim states the contraposition of this.

Claim 4.14. *For each vertex $u = (x,y,z) \in \mathscr{U}$, if $x \neq y$ then $S'(P'(u)) = u$ and $P'(S'(u)) = u$.*

Proof. We only consider when $S(x) = y$ and $P(y) = x$ hold because x is a self-loop in all other cases. From Algorithm 3, we have

$$P'(u) = \begin{cases} (x,y,z-1) & \text{if } V(x)+1 < z < V(y)-1, \\ (x,x,z-1) & \text{if } V(x)+1 = z < V(y). \end{cases}$$

Hence, $S'(P'(u)) = u$ holds. Similarly, we can see that $P'(S'(u)) = u$ holds if $x \neq y$. Therefore, the claim follows. □

So far, we have shown the properties that the intermediate instance $\mathscr{I}_M$ satisfies. We prove that we can efficiently convert a solution to the original instance $\mathscr{I}$ from a solution to the intermediate instance $\mathscr{I}_M$. First, we show that every sink $u = (x,y,z) \in \mathscr{U}$ on S' and P' includes an original solution, specifically, x is a sink for $\mathscr{I}$. This fact appears in the proof of the next claim. The following claim is useful in the second step; a vertex u is a sink if and only if u has potential $2^h - 1$.

Claim 4.15. *Let $u = (x,y,z) \in \mathscr{U}$. A vertex u is a sink on S' and P', i.e., $P'(S'(u)) \neq u$, if and only if $V'(u) = z = 2^h - 1$.*

Proof. First, we prove that $V'(u) = z = 2^h - 1$ when a vertex u is a sink on S' and P'. From Claim 4.14 and Claim 4.13, it satisfies that $x = y$ and $S'(u) = u \neq P'(u)$. We show that $P(S(x)) \neq x$ holds. For the sake of contradiction, we suppose that

$P(S(x)) = x$. It is clear that $S'(u) = u = P'(u)$ if $S(x) = x$. This is contradiction from $P'(u) \neq u$. Therefore, we have $S(x) \neq x$ and $P(S(x)) = x$. It satisfies that $V(x) \neq z$ since $S'(u) = u$. This implies that $P'(u) = u$, which contradicts. Hence, $P(S(x)) \neq x$ holds, and thus, x is a sink on the original instance. Notice that $P'(u) \neq u$, we have $V(x) < z$, and it satisfies that $z = 2^h - 1$ by $S'(u) = u$.

Conversely, we prove that if $V'(u) = z = 2^h - 1$, then $P'(S'(u)) \neq u$ holds. It is easy to see that at least one of $S'(u) \neq u$ and $P'(u) \neq u$ since $V'(u) = z = 2^h - 1 > 0$. By definition, every $u \in \mathscr{U}$ has $V'(u) < 2^h$. This implies that $S'(u) = u$ and $P'(u) \neq u$ hold. □

Next, we show that we can efficiently take an original solution from each source $u = (x, y, z) \in \mathscr{U}$ on S' and P'. This fact appears in the proof of Claim 4.16. In this claim, we show that if a vertex u is a source, u has potential 0. Conversely, if a vertex u has a potential 0 and it is not self-loop, u is a source. Notice that assuming $u \notin \Pi'$ in Claim 4.16, we obtain a non-standard source on S and P, that is, we get a solution to $\mathscr{I}$ efficiently.

Claim 4.16. *We assume that a vertex $u = (x, y, z) \in \mathscr{U}$ is not a self-loop for S' and P'. The vertex u is a source, i.e., $S'(P'(u)) \neq u$, if and only if $V'(u) = z = 0$.*

Proof. From Claim 4.14, we have $x = y$. Applying Claim 4.13, it satisfies that $P'(u) = u \neq S'(u)$. First, we prove that $S(P(x)) \neq x$ holds. For the contradiction, we suppose that $S(P(x)) = x$. We obviously see that $S'(u) = u$ if $P(x) = x$. This is a contradiction from $S'(u) \neq u$. Therefore, we have $P(x) \neq x$ and $S(P(x)) \neq x$. Since $P'(u) = u$, it satisfies that $V(x) \neq z$. and thus, we have $S'(u) = u$, which contradicts. Hence, $S(P(x)) \neq x$ holds, i.e., x is a source on the original instance. Notice that $S'(u) \neq u$, we have $V(x) > z$, and it satisfies that $z = 0$ by $P'(u) = u$.

Conversely, we prove that if $V'(u) = z = 0$, then $S'(P'(u)) \neq u$. Notice that u is not a self-loop for S' and P', we have at least one of $S'(u) \neq u$ and $P'(u) \neq u$. Furthermore, for every vertex in $\mathscr{U}$, the potential function V' has a non-negative value. It must hold that $P'(u) = u$ from Claim 4.13, and thus we have $S'(u) \neq u$. This implies that the vertex u is a source for S' and P'. □

We complete the first step, construction of the intermediate instance $\mathscr{I}_M$. Note that this instance is also a valid instance of kS-EoPL. From now on, we start the second step. In the second step, we construct a polynomial-time reduction from kS-EoPL to End of Potential Line. Specifically, we transform $\mathscr{I}_M$ to the instance $\mathscr{J}$ of End of Potential Line in polynomial time. Note that we can extract an original solution from a solution to $\mathscr{J}$ if we efficiently extract an intermediate solution from a solution to $\mathscr{J}$ since we can convert an original solution from a solution to the intermediate solution in polynomial time.

First, we define the set $\Gamma_k := \bigcup_{\alpha=0}^{2^h-1} \Gamma_k(\alpha)$, where

$$\Gamma_k(\alpha) := \{\{u_1 \ldots, u_k\}; \forall i \neq j, u_i \neq u_j, \forall i, V'(u_i) = \alpha, \text{ and } u_i \in \mathscr{U}\}.$$

It is easy to see that we can decide whether a given string $v = \{u_1, \ldots, u_k\}$ is in Γ_k in polynomial time. Moreover, the bit-length of $v = \{u_1, \ldots, u_k\}$ in Γ_k is bounded by some polynomial in n. We need $(2n+h)$-bits to represent each u_i, so $k(2n+h)$-bits

are sufficient to represent v. Notice that k is a constant number, this is a polynomial in n.

For each vertex $v = \{u_1, \ldots, u_k\}$ in Γ_k, we define the successor circuit S_k and the predecessor circuit P_k as follows. If there exists an element u_i such that $S'(u_i) = u_i = P'(u_i)$, then $S_k(v) = v$, otherwise $S_k(v) = \{S'(u_1), \ldots, S'(u_k)\}$. Similarly, if there exists an element u_i such that $S'(u_i) = u_i = P'(u_i)$, then $P_k(v) = v$, otherwise $P_k(v) = \{P'(u_1), \ldots, P'(u_k)\}$.

It satisfies that $S_k(v)$ is also in Γ_k for each vertex $v = \{u_1, \ldots, u_k\}$ in Γ_k. It is clear that when there is an element u_i such that $S'(u_i) = u_i = P'(u_i)$. Therefore, we show when there are no such elements. It suffices to prove that $S'(u_i) \neq S'(u_j)$ for each pair of u_i and u_j with $u_i \neq u_j$. If u_i is a sink on S' and P', i.e., $S'(u_i) = u_i \neq P(u_i)$, then $V'(u_i) = 2^h - 1$. From Claim 4.15, it follows that $V'(u_j) = 2^h - 1$ for every j, and thus, u_j is also a sink. We have that $S'(u_i) = u_i \neq u_j = S'(u_j)$. Next, we show when u_i is not a sink on S' and P'. Note that u_j is also not a sink. Suppose for the sake of contradiction that $S'(u_i) = S'(u_j)$. From Claim 4.13, it satisfies that $P'(S'(u_i)) = u_i$. We get the following $u_i = P'(S'(u_i)) = P'(S'(u_j)) = u_j$, getting a contradiction. Finally, we can easily see that $V'(S'(u_i)) = V'(S'(u_j))$ for each pair of u_i and u_j with $u_i \neq u_j$ from Claim 4.13. Similarly, we can see that $P'(v)$ is also in Γ_k for each vertex $v = \{u_1, \ldots, u_k\}$ in Γ_k.

We define the new potential function V_k as follows.

$$V_k(v) = \begin{cases} 0 & \text{if } S_k(v) = v = P_k(v), \\ V(u_1) & \text{otherwise.} \end{cases}$$

By definition, the above function S_k, P_k, and V_k can be constructed in polynomial time, and they are polynomial-time computable. Finally, the new standard source is defined as $\pi^* = \{(\pi_1, \pi_1, 0), (\pi_2, \pi_2, 0), \ldots, (\pi_k, \pi_k, 0)\} \in \Gamma_k$. It is easy to see that $P_k(\pi^*) = \pi^* \neq S_k(\pi^*)$ and $V_k(\pi^*) = 0$ hold. Therefore, the tuple $\mathscr{J} = (k(2n+h), h, S_k, P_k, V_k, \pi^*)$ is a valid instance of End of Potential Line. In the rest of this proof, we show that when we obtain a solution of $\mathscr{J}$, we can convert to an original solution in polynomial time.

First, we show that there are no violating solutions for $\mathscr{J}$. Suppose that $S_k(v) \neq v = \{u_1, \ldots, u_k\}$. Then v is not a self loop with respect to S_k and P_k. Furthermore, if there is an element u_i in v such that $S'(u_i) = u_i \neq P'(u_i)$, then it satisfies that $z_i = 2^h - 1$ from Claim 4.15. This implies that $z_j = 2^h - 1$ for every $j \in [k]$. This is a contradiction from $S_k(v) \neq v$ because it satisfies that $S'(u_j) = (u_j)$ for every $j \in [k]$. Therefore, it holds that $S'(u_i) \neq u_i$ for every $i \in [k]$. Therefore, we obtain that $P'(S'(u_i)) = u_i$ by the definitions of S' and P'. From our normalization assumption for $\mathscr{I}$, it satisfies that $V(S(x)) = V(x) + 1$ unless $S(x) = x$. This implies that $V'(S'(x)) > V'(x)$ holds, and thus, there are no violating solutions for $\mathscr{J}$.

Second, we consider that we obtain a solution $v = \{u_1, \ldots, u_k\}$ satisfying that $P_k(S_k(v)) \neq v$. Then v has no elements which are self-loops. Hence, it holds that

$$\{P'(S'(u_1)), \ldots, P'(S'(u_k))\} \neq \{u_1, \ldots, u_k\}.$$

Therefore, there exists at least one element u_i such that $P'(S'(u_i)) \neq u_i$ holds. From Claim 4.15, it must hold that $V'(u_i) = 2^h - 1$, and thus, $V'(u_1) = \cdots = V'(u_k) = 2^h - 1$ since $v \in \Gamma_k$. For every element $u_j = (x_j, y_j, z_j)$ in v, it holds that $P'(S'(u_j)) \neq u_j$.

This implies that it satisfies that $P(S(x_j)) \neq x_j$ for every x_j. Therefore, we can extract one solution for the original instance in polynomial time.

Finally, we consider that we obtain a solution $v = \{u_1, \ldots, u_k\}$ satisfying that $S_k(P_k(v)) \neq v \neq \pi^*$. Then v has no elements which are self-loops. Hence, it holds that

$$\{S'(P'(u_1)), \ldots, S'(P'(u_k))\} \neq \{u_1, \ldots, u_k\}.$$

Therefore, there exists at least one element u_i such that $S'(P'(u_i)) \neq u_i$ holds. From Claim 4.16, it holds that $V'(u_i) = 0$. Hence, $V'(u_1) = \cdots = V'(u_k) = 0$ since $v \in \Gamma_k$. Note that for every $j \in [k]$, $u_j = (x_j, y_j, z_j)$ is not a self-loop, and thus, this implies that $S'(P'(u_j)) \neq u_j$ holds. Therefore, it satisfies that $S(P(x_j)) \neq x_j$ for every x_j. Since $v \neq \pi^*$, there exists at least one string $x_{j'}$ such that $x_{j'} \notin \Pi$. Such a string $x_{j'}$ corresponds to an original solution. Therefore, we can extract one solution for the original instance in polynomial time.

From the above arguments, we complete the polynomial-time reduction from kS-EoPL to End of Potential Line, and this implies that the problem kS-EoPL is a member of the class EOPL. □

The following theorem follows from Lemma 4.11 and Lemma 4.12.

Theorem 4.17. *kS-EoPL is EOPL-complete.*

Note that the proof of Lemma 4.12 works even if k is not constant. Indeed, our proof works for a multiple-source variant that has at most polynomially numbers of sources. We consider the following problem, and we immediately see that this problem is an EOPL-complete problem.

Definition 4.18. Multiple-Source End of Potential Line (abbreviated MS-EoPL)
Input:

- two Boolean circuits $S, P : \Sigma^n \to \Sigma^n$
- a Boolean circuit $V : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$
- a set of k standard sources $\Pi = \{\pi_1, \ldots, \pi_k\}$ such that $P(\pi) = \pi \neq S(\pi)$ and $V(\pi) = 0$ for each π in Π.

Task: Find a vertex x in Σ^n satisfying at least one of the following:

(R1) $P(S(x)) \neq x$, i.e., x is a sink

(R2) $S(P(x)) \neq x \notin \Pi$, and

(V1) $S(x) \neq x$, $P(S(x)) = x$, and $V(S(x)) - V(x) \leq 0$

Lemma 4.19. MS-EoPL *is EOPL-complete.*

Remark 4.20. Another variant of End of Potential Line worth considering is finding a non-standard source, a sink, or a non-increasing arc when we are given k standard sources and l standard sinks, where $l < k$. In the case of End of Line, it can be easily reduced to $(k-l)$S-EoL in polynomial time [HG18]. We can add

a valid arc from each of l standard sinks to some corresponding known source, and thus, we obtain a valid instance with $(k-l)$ standard sources and no standard sinks. Unfortunately, this simple technique does not work for a variant of END OF POTENTIAL LINE. Recall that every source/sink given by our reduction from kS-EOPL to END OF POTENTIAL LINE contains k sources/sinks. Furthermore, the vertex consisting of k standard sources is a standard source of the new instance. Therefore, we can extract a non-standard source from an obtained non-standard source. Notice that $l < k$, we can extract a non-standard sink from an obtained sink for the new instance. That is, this variant is also EOPL-complete.

4.3.2 Higher Degree Problem: IMBALANCE with Potential

Up to this point, we have only considered implicit graphs where every vertex has in-degree/out-degree at most one. However, the principle that guarantees the totality of END OF POTENTIAL LINE can be generalized to higher degree implicit graphs. If we are given an implicit digraph with potential and an "unbalanced" vertex, i.e., a vertex with in-degree $\neq$ out-degree, then there must exist another unbalanced vertex or a non-increasing arc.

In this section, we consider the new variant of the problem END OF POTENTIAL LINE that corresponds to a higher degree digraph. We define this problem as follows.

Definition 4.21. POTENTIAL IMBALANCE
Input:

- an implicit digraph $G(S,P) = (\Sigma^n, E)$
  - defined by two Boolean circuits $S, P : \Sigma^n \to \Sigma^{dn}$
- a Boolean circuit $V : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$
- an unbalanced vertex $\pi \in \Sigma^n$ such that $\delta^+(\pi) > \delta^-(\pi)$.

Task: Find a vertex $x \in \Sigma^n$ satisfying at least one of the following:

(R1) $\delta^+(x) \neq \delta^-(x)$ and $x \neq \pi$,

(V1) there exists a valid arc $(x,y) \in E$ such that $V(y) \leq V(x)$

By definition, it is not hard to see that the problem POTENTIAL IMBALANCE is EOPL-hard. Surprisingly, this problem also belongs to EOPL, and thus POTENTIAL IMBALANCE is EOPL-complete. To prove this, we show that there is a polynomial-time reduction from POTENTIAL IMBALANCE to MS-EOPL in Lemma 4.22.

Lemma 4.22. POTENTIAL IMBALANCE *is reducible to* MS-EOPL *in polynomial time.*

Proof. Our proof relies on the proof in Hollender and Goldberg [HG18]. The reduction constructed by this proof is the same as their technique, except for defining the potential function. Indeed, our proof is completed simply by adding the construction of a natural potential function to a polynomial-time reduction from IMBALANCE to MS-EOPL constructed by Hollender and Goldberg [HG18].

Let (n,m,d,S,P,V,π) be a valid instance of POTENTIAL IMBALANCE, and let $G(S,P) = (\Sigma^n, E)$ be the implicit digraph produced by S and P, where E is a set of all valid arcs with respect to S and P.

To construct a reduction to MS-EOPL, we apply the "chessplayer algorithm" used to prove that ODD reduces to LEAF by Papadimitriou [Pap94b]. We fix a label function $\lambda_x^+ : E^+(x) \to [\delta^+(x)]$ and $\lambda_x^- : E^-(x) \to [\delta^-(x)]$ for each vertex $x \in \Sigma^n$. The function $\lambda_x^+(y)$ is the index of y in successor list of x, ordered in lexicographically. Similarly, the function $\lambda_x^-(z)$ corresponds to the index of z in predecessor list of x, ordered in lexicographically. Note that both functions λ_x^+ and λ_x^- are bijective and polynomial-time computable. This means that there exists an algorithm computing the label $\lambda_x^+((x,y))$ for given vertex x in Σ^n and valid arc (x,y) in $E^+(x)$ in polynomial time. Similarly, there exists an algorithm computing the label $\lambda_x^-((y,x))$ for a given vertex x in Σ^n and a valid arc (y,x) in $E^-(x)$ in polynomial time.

To show the reduction to MS-EOPL, we consider the following vertex set,

$$\mathscr{K} = \bigcup_{x \in \Sigma^n} \{(i,x); i \in [d(x)]\},$$

where $d(x) = \max\{\delta^+(x), \delta^-(x)\}$. Furthermore, we define a successor S' and a predecessor P' over $\mathscr{K}$ as follows.

For each vertex $(i,x) \in \mathscr{K}$, if $i > \delta^+(x)$, then define $S'(i,x) = (i,x)$, else define $S'(i,x) = (j,y)$, where $y \in \Sigma^n$ satisfies that $\lambda_x^+((x,y)) = i$ and the label $j = \lambda_y^-((x,y))$. Since the label function is a bijection, such a string y and a label j are unique for each vertex (i,x).

Similarly, for each vertex $(i,x) \in \mathscr{K}$, if $i > \delta^-(x)$, then $P'(i,x) = (i,x)$, else define $P'(i,x) = (j,y)$, where $y \in \Sigma^n$ satisfies that $\lambda_x^-((y,x)) = i$ and the label $j = \lambda_y^+((y,x))$. Since the label function is a bijection, such a string y and a label j are unique for each (i,x).

It is easy to see that the functions S' and P' can be constructed in polynomial time. Straightforwardly, following two claims hold.

Claim 4.23. *For each vertex $(i,x) \in \mathscr{K}$, if $S'(i,x) \neq (i,x)$, then $P'(S'(i,x)) = (i,x)$.*

Proof. Since $S'(i,x) \neq (i,x)$, it satisfies that $i \leq \delta^+(x)$. We can uniquely write $S'(i,x) = (j,y)$ since the label function is a bijection. Now, it satisfies that $(\lambda_x^+)^{-1}(i) = (x,y) \in E$ and $j = \lambda_y^-((x,y))$. Therefore, $P'(j,y) = (i,x)$ holds, and thus, it satisfies that $P'(S'(i,x)) = (i,x)$. □

Lemma 4.24. *For each vertex $(i,x) \in \mathscr{K}$, if $P'(i,x) \neq (i,x)$, then $S'(P'(i,x)) = (i,x)$.*

Proof. Since $P'(i,x) \neq (i,x)$, it satisfies that $i \leq \delta^-(x)$. We can uniquely write $P'(i,x) = (j,y)$. Now it satisfies that $(\lambda_x^-)^{-1}(i) = (y,x) \in E$ and $j = \lambda_y^+((y,x))$. Therefore, we obtain that $S'(j,y) = (i,x)$, and thus, this implies that $S'(P'(i,x)) = (i,x)$ holds. □

Furthermore, we define by $\Pi' = \{(i,\pi); \delta^-(\pi) < i \leq \delta^+(\pi)\}$ the set of all standard sources. It is easy to see that every element of Π' is a souse. For every vertex

(i,x), we define

$$V'(i,x) := \begin{cases} 0 & \text{if } (i,x) \in \Pi', \\ V(x) & \text{otherwise.} \end{cases}$$

From the above arguments, the tuple $(m+n, m, S', P', V', \Pi')$ is a valid instance of MS-EOPL, and these constructions can be in polynomial time. In the rest of this proof, we show that when we obtain a solution of $(m+n, m, S', P', V', \Pi')$, we can convert it to an original solution in polynomial time.

First, we consider the case that we obtain a solution $(i,x) \in \mathscr{K}$ satisfying that $P'(S'(i,x)) \neq (i,x)$. Then it holds that $S'(i,x) = (i,x)$ from Claim 4.23. Hence, it satisfies that $\delta^+(x) < i$. Since $S'(i,x) = (i,x)$, it holds that $P'(S'(i,x)) = P'(i,x) \neq (i,x)$. Therefore, it satisfies that $i \leq \delta^-(x)$, and thus, $\delta^+(x) < \delta^-(x)$ holds. This implies that x is a solution for the original instance.

Second, we consider the case that we obtain a solution $(i,x) \in \mathscr{K}$ satisfying that $S'(P'(i,x)) \neq (i,x) \notin \Pi'$. Then it holds that $\delta^-(x) < i$ from the definition of P'. This implies that $x \neq \pi$ since $(i,x) \notin \Pi'$. Furthermore, it holds that $S'(P'(i,x)) = S'(i,x) \neq (i,x)$ since $P'(i,x) = (i,x)$, and thus, $i \leq \delta^+(x)$ holds. Hence, it satisfies that $\delta^-(x) < \delta^+(x)$, and this implies that x is a solution for the original instance.

Finally, we consider the case that we obtain a solution (i,x) satisfying that $S'(i,x) \neq (i,x)$, $P'(S'(i,x)) = (i,x)$, and $V'(S(i,x)) \leq V'(i,x)$. Since $S'(i,x) \neq (i,x)$, there is a valid arc $(x,y) \in E$ such that $S'(i,x) = (j,y)$. Furthermore, it holds that $V(y) = V'(S'(i,x)) \leq V'(i,x) = V(x)$ by the definition of V'. Hence, x is a solution to the original instance.

We complete the construction of the polynomial-time reduction form POTENTIAL IMBALANCE to MS-EOPL. □

Straightforwardly, we obtain the following theorem from the above lemma.

Theorem 4.25. POTENTIAL IMBALANCE *is* EOPL-*complete.*

Remark 4.26. In the definition of the problem POTENTIAL IMBALANCE (see Definition 4.21), we assume that the standard unbalanced vertex π satisfies that $\delta^+(\pi) > \delta^-(\pi)$ on the implicit digraph $G(S,P)$ with the potential function V. In fact, the proof of Lemma 4.22 works even if we assume that the standard vertex π satisfies that $\delta^+(\pi) < \delta^-(\pi)$ on the implicit digraph $G(S,P)$ with the potential function V by making a simple modification. Specifically, we define the new potential function as $(2^m - 1) - V(x)$ for each vertex x in Σ^n, and change the direction of every valid arcs. Therefore, without loss of generality, we assume that the standard unbalanced vertex π given by an instance of POTENTIAL IMBALANCE satisfies that $\delta^+(\pi) \neq \delta^-(\pi)$.

4.3.3 Looking for Multiple Solutions

As mentioned in the Introduction, Hollender and Goldberg [HG18] also proved that the problem, in which given k sources, find k distinct sources or sinks, is also PPAD-complete. Now, we consider a generalized problem of END OF POTENTIAL LINE that is similar to this problem. The principle that guarantees the existence of solutions for END OF POTENTIAL LINE implies that if there are k sources, then there must

exist at least k distinct sinks. Clearly, this problem is EOPL-hard. In this section, we show that this problem also belongs to EOPL. The new variant of the problem END OF POTENTIAL LINE is defined as follows.

Definition 4.27. k-END OF POTENTIAL LINE
Input:

- two Boolean circuits $S, P : \Sigma^n \to \Sigma^{dn}$
- a Boolean circuit $V : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$
- a set of k standard sources $\Pi = \{\pi_1, \ldots, \pi_k\}$ such that $P(\pi) = \pi \neq S(\pi)$ and $V(\pi) = 0$ for each π in Π.

Task: Find a vertex $x \in \Sigma^n$ satisfying at least one of the following:

(R1) $x_i \neq P(S(x_i))$,

(R2) $S(P(x_i)) \neq x_i \notin \Pi$, and

(V1) $S(x_i) \neq x_i$ and $P(S(x_i)) = x_i$, and $V(S(x_i)) - V(x_i) \leq 0$.

By definition, it is easy to see that kS-EOPL is EOPL-hard. For every given instance of END OF POTENTIAL LINE, we create k copies it.

Lemma 4.28. k-EOPL *is* EOPL-*hard.*

Surprisingly, the complexity is the same as in EOPL; this problem is reducible to END OF POTENTIAL LINE in polynomial time. We show this result in Lemma 4.29. Our technique is an extension of the technique by Hollender and Goldberg [HG18] to END OF POTENTIAL LINE. Their technique has a weak point: It possibly can not efficiently compute a potential for each vertex. Our method shown in Lemma 4.12 gets over this weak point, i.e., we can efficiently compute a potential for each vertex.

Lemma 4.29. k-EOPL *is reducible to* END OF POTENTIAL LINE *in polynomial time.*

Proof. Let $\mathscr{I} = (n, m, S, P, V, \Pi)$ be a valid instance k-EOPL. First, we assume that $\mathscr{I}$ is already normalized by applying the method Proposition 4.2. Notice that our method also works for multiple-source variants. Hence, in this proof, we assume that the following properties holds:

- if $S(x) \neq x$ then $P(S(x)) = x$ and $V(S(x)) > V(x)$;
- if $P(x) \neq x$ then $S(P(x)) = x$ and $V(x) > V(P(x))$.

Note that we might be able to extract strictly fewer than k solutions to the original instance when we are given k solutions to the normalized instance. In the normalization method shown in Proposition 4.2, we remove a non-increasing arc, and we make two new solutions; a sink and a non-standard source. If we get these two solutions, we can only extract a single solution to the original instance.

Our reduction from k-EOPL to END OF POTENTIAL LINE constructed in this proof avoids this. A solution to the new instance consists of either k' distinct sinks or k' distinct sources for the normalized instance, where $k' \geq k$. Notice that every

solution of the new instance does not mix the sinks and sources of the normalized instance. Therefore, we always extract the k distinct solutions to the original instance.

Let $\mathscr{U} = \Sigma^n \times \Sigma^n \times \{0,1,\ldots,2^h-1\}$ and $h = 2n+m$. First, we construct three polynomial-time computable and constructive circuits $S', P' : \mathscr{U} \to \mathscr{U}$ and $V' : \mathscr{U} \to \{0,1,\ldots,2^h-1\}$ that are constructed in the proof of Lemma 4.12.

Hereafter, the same construction as Lemma 4.12 is continued. However, only the vertex set is different. For each $l = 0,1,\ldots,k$ and each $\alpha = 0,1,\ldots,2^h-1$, we define $\Gamma_{k+l} := \bigcup_{\alpha=0}^{2^h-1} \Gamma_{k+l}(\alpha)$, where

$$\Gamma_{k+l}(\alpha) := \{\{u_1,\ldots,u_{k+l}\}; \forall i \neq j, u_i \neq u_j \text{ and } \forall i, V'(u_i) = \alpha, u_i \in \mathscr{U}\}.$$

Notice that k is constant. For every given string $v = \{u_1,\ldots,u_{k+l}\}$, we can check whether v is in Γ_{k+l} in polynomial time. Moreover, the bit-length of string $v = \{u_1,\ldots,u_{k+l}\} \in \Gamma_{k+l}$ is bounded by polynomial in n. We need $(2n+h)$-bits to represent each u_i, and thus, $(2k+1)(2n+h)$-bits are sufficient to represent v.

For each $l = 0,1,\ldots,k$, we define a successor S_{k+l} and a predecessor P_{k+l} over Γ_{k+l} as follows.

- If l is even, then for each $v = \{u_1,\ldots,u_{k+l}\}$, we define

$$S_{k+l}(v) := \begin{cases} \{(u_1,\ldots,u_{k+l}\} & \text{if there exists } u_i \text{ that is a self-loop,} \\ \{S'(u_1),\ldots,S'(u_{k+l})\} & \text{otherwise,} \end{cases}$$

  and

$$P_{k+l}(v) := \begin{cases} \{u_1,\ldots,u_{k+l}\} & \text{if there exists } u_i \text{ that is a self-loop,} \\ \{P'(u_1),\ldots,P'(u_{k+l})\} & \text{otherwise.} \end{cases}$$

- If l is odd, then for each $v = \{u_1,\ldots,u_{k+l}\}$, we define

$$S_{k+l}(v) := \begin{cases} \{u_1,\ldots,u_{k+l}\} & \text{if there exists } u_i \text{ that is a self-loop,} \\ \{P'(u_1),\ldots,P'(u_{k+l})\} & \text{otherwise,} \end{cases}$$

  and

$$P_{k+l}(v) := \begin{cases} \{u_1,\ldots,u_{k+l}\} & \text{if there exists } u_i \text{ that is a self-loop,} \\ \{S'(u_1),\ldots,S'(u_{k+l})\} & \text{otherwise.} \end{cases}$$

Note that the direction of the valid arc is different when l is even and odd. Furthermore, we define the potential function V_{k+l} depending on the parity of l as follows. For each vertex $v = \{u_1,\ldots,u_{k+l}\}$ in Γ_{k+l}, if v is a self loop with respect to S' and P', then we define $V_{k+l}(v) = 0$, else we define

$$V_{k+l}(v) := \begin{cases} V'(u_1) & \text{if } l \text{ is even,} \\ (2^h-1) - V'(u_1) & \text{if } l \text{ is odd.} \end{cases}$$

Finally, we define by $\Pi' = \{(\pi_1,\pi_1,0),\ldots,(\pi_k,\pi_k,0)\}$ the standard source, where π_i is in Π for each $i \in [k]$.

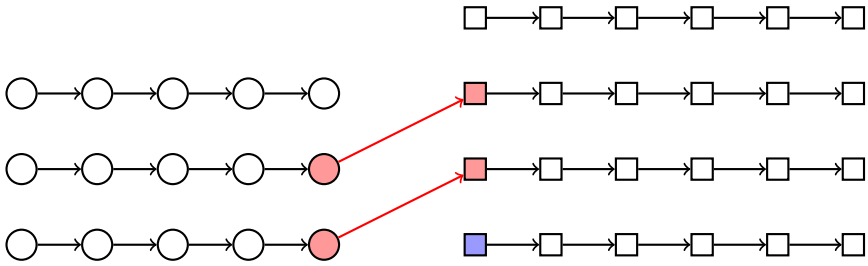

FIGURE 4.3: The paths consisting of rounded vertices is a path where l is odd. On the other hand, the paths consisting of square vertices is a path where l is even. The red vertices are bad solutions. The blue vertex is a standard source. The red arcs represent the new arc introduced by the method of Hollender and Goldberg [HG18].

We denote by $G_{k+l}(S_{k+l}, P_{k+l}) = (\Gamma_{k+l}, E_{k+l})$ the implicit digraph that are derived from S_{k+l}, P_{k+l} for each $l = 0, 1, \ldots, k$. Moreover, the implicit graph $G_{k+l}(S_{k+l}, P_{k+l})$ has the potential function V_{k+l} for each $l = 0, 1, \ldots, k$. To simply the notation, we denote by $G_{k+l}(S_{k+l}, P_{k+l}, V_{k+l})$ this implicit digraph with potential. It is easy to see that $G_{k+l}(S_{k+l}, P_{k+l}, V_{k+l})$ has the following sinks and sources:

- Let $t_1, \ldots, t_{k+l}$ be distinct $k+l$ sinks with respect to S' and P'. Then the vertex $\{t_1, \ldots, t_{k+l}\}$ in Γ_{k+l} is a sink of G_{k+l} if l is even, and $\{t_1, \ldots, t_{k+l}\}$ in Γ_{k+l} is a non-standard source of G_{k+l} if l is odd. Therefore, we can extract at least k original solutions.

- Let $s_1, \ldots, s_{k+l}$ be distinct $k+l$ sources with respect to S' and P'. Then the vertex $\{s_1, \ldots, s_{k+l}\}$ in Γ_{k+l} is a source of G_{k+l} if l is even, and $\{s_1, \ldots, s_{k+l}\}$ in Γ_{k+l} is a sink of G_{k+l} if l is odd. However, it might contain strictly less than k non-standard sources of the original instance, and thus, we can not extract k distinct original solutions. In this case, we call $\{s_1, \ldots, s_{k+l}\}$ a bad solution.

To complete our reduction from k-EOPL to END OF POTENTIAL LINE, we need to remove every bad solution. Hollender and Goldberg [HG18] established the method of removing every bad solution from a sink and a non-standard source of G_{k+l}. They made a one-to-one correspondence between bad solutions that are sinks and bad solutions that are sources and connected two bad solutions. Thereby, their technique removed bad solutions from sinks and non-standard sources. See Figure 4.3. Notice that a bad solution is a sink when l is odd, and a bad solution is a source when l is even, we can efficiently compute the potential without generating violating solutions.

Our reduction heavily relays on the elegant technique of Hollender and Goldberg [HG18]; in the following, we apply their method to construct the new successor circuit $\hat{S}$ and the new predecessor circuit $\hat{P}$.

First, we fix a strict order on the set of standard sources Π. Here, we suppose that an order such $\pi_1 \prec \pi_2 \prec \cdots \prec \pi_k$ is given. Then we consider the set $\hat{\Gamma} = \bigcup_{l=0}^{k} \hat{\Gamma}_{k+l}$, where $\hat{\Gamma}_{k+l} := \{(v, (a_1, \ldots, a_l));\ v \in \Gamma_{k+l}, a_i \in \Pi$ for all $i \in [l]$, and $a_1 \preceq a_2 \preceq \cdots \preceq a_l\}$. In particular, for $l = 0$, the elements of $\hat{\Gamma}_{k+l}$ are the form $(v, ())$, where $()$ denotes the empty tuple. Furthermore, the bit-length of a string $(v, (a_1, \ldots, a_l)) \in \hat{\Gamma}$ is bounded by some polynomial in n. We need at most $(2k+1)(2n+h)$-bits to

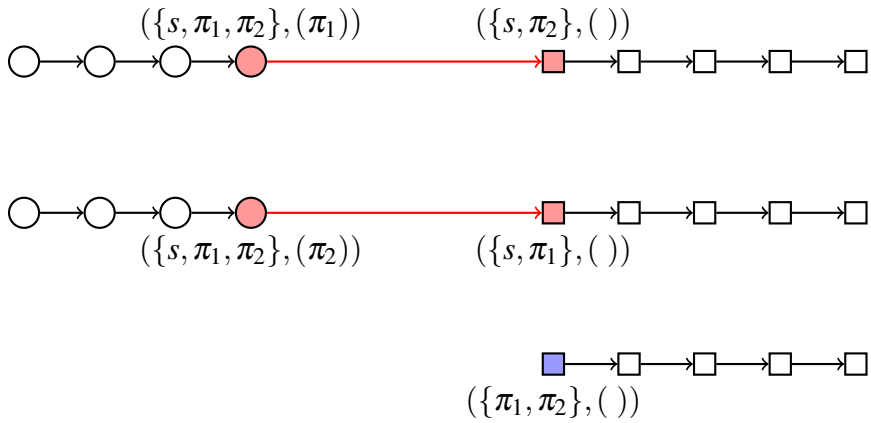

FIGURE 4.4: An example of removing bad solutions when $k=2$. An element s is any source, and π_i is a standard source for each $i=1,2$. The paths consisting of rounded vertices is a path where $l=1$. On the other hand, the paths consisting of square vertices is a path where $l=0$. The red vertices are bad solutions. The blue vertex is a standard source. Note that when $l=2$, every sink and every source are solution that we can extract two original solutions.

represent v, $(2n+h)$-bits to represent each a_l, and $(\log_2(l)+1)$-bits to express the value l. Hence, $((2k+1)(2n+h)+k(2n+h)+\log_2(k)+1)$-bits are sufficient to represent $(v,(a_1,\ldots,a_l))$. This is a polynomial in n since k is a constant number.

We define the new successor circuit $\hat{S}$ and the new predecessor circuit $\hat{P}$ for this vertex set $\hat{\Gamma}$ as follows. First, we consider the case of that l is even. For each vertex $(v,(a_1,\ldots,a_l))$ in $\hat{\Gamma}$, if v is not a source, then we define $\hat{S}(v,(a_1,\ldots,a_l)) := (S_{k+l}(v),(a_1,\ldots,a_l))$ and $\hat{P}(v,(a_1,\ldots,a_l)) := (P_{k+l}(v),(a_1,\ldots,a_l))$. On the other hand, if v is a source, i.e., it satisfies that $S_{k+l}(P_{k+l}(v)) \neq v$, then we can write $v = K \cup U$, where $K \subseteq \Pi'$ and $U \subseteq \Gamma_{k+l} \setminus \Pi'$. Note that $K \cap U = \emptyset$ holds. If $|U| \geq k$, then we can extract k distinct original solutions, and thus, we define $\hat{S}(v,(a_1,\ldots,a_l)) := (S_{k+l}(v),(a_1,\ldots,a_l))$ and $\hat{P}(v,(a_1,\ldots,a_l)) := (P_{k+l}(v),(a_1,\ldots,a_l))$. However, if $|U| < k$, then we can not extract k distinct original solutions. Hence, we introduce the new valid arc to remove the vertex $(v,(a_1,\ldots,a_l))$ from a non-standard source.

Here, let $\bar{K} = \Pi' \setminus K$, and define $\max(X) := \arg\max_{\prec} X$ for every subset $X \subseteq \Pi'$; specifically, let $a_1 \succ \max(\emptyset)$. For vertex $v = K \cup U$, if $a_l \succ \max(\bar{K})$, then we define

$$\hat{S}(v,(a_1,\ldots,a_l)) := (S_{k+l}(v),(a_1,\ldots,a_l))$$

and

$$\hat{P}(v,(a_1,\ldots,a_l)) := ((K \setminus \{a_l\} \cup U),(a_1,\ldots,a_{l-1})).$$

On the other hand, if $a_l \preceq \max(\bar{K}) =: j$ and $l > 0$, then we define

$$\hat{S}(v,(a_1,\ldots,a_l)) := (S_{k+l}(v),(a_1,\ldots,a_l))$$

and

$$\hat{P}(v,(a_1,\ldots,a_l)) := ((K \cup \{j\}) \cup U),(a_1,\ldots,a_l,j)),$$

else if $a_l \preceq \max(\bar{K}) =: j$ and $l = 0$, the define as follows.

$$\hat{S}(v, ()) := (S_{k+l}(v), ())$$

and

$$\hat{P}(v, ()) := \begin{cases} ((K \cup \{j\}) \cup U), (j)) & \text{if } |U| > 0, \\ (v, ()) & \text{if } |U| = 0. \end{cases}$$

Next, we consider the case that l is odd. For each vertex $(v, (a_1, \ldots, a_l)) \in \hat{\Gamma}$, if v is not a sink, then $\hat{S}(v, (a_1, \ldots, a_l)) = (S_{k+l}(v), (a_1, \ldots, a_l))$ and $\hat{P}(v, (a_1, \ldots, a_l)) = (P_{k+l}(v), (a_1, \ldots, a_l))$. On the other hand, if v is a sink, i.e., it satisfies that $P'(S'(v)) \neq v$, then we can write $v = K \cup U$, where $K \subseteq \Pi'$ and $U \subseteq \Gamma_{k+l} \setminus \Pi'$ with $K \cap U = \emptyset$. If $|U| \geq k$, then we can extract k distinct original solutions. Hence we define $\hat{S}(v, (a_1, \ldots, a_l)) = (S_{k+l}(v), (a_1, \ldots, a_l))$ and $\hat{P}(v, (a_1, \ldots, a_l)) = (P_{k+l}(v), (a_1, \ldots, a_l))$. However, if $|U| < k$, then we can not extract k distinct original solutions. Hence, we introduce the new valid arc to remove the vertex $(v, (a_1, \ldots, a_l))$ from a sink.

For vertex $v = K \cup U$, if $a_l \succ \max(\bar{K})$, then we define

$$\hat{S}(v, (a_1, \ldots, a_l)) := ((K \setminus \{a_l\}) \cup U), (a_1, \ldots, a_{l-1}))$$

and

$$\hat{P}(v, (a_1, \ldots, a_l)) := (P_{k+l}(v), (a_1, \ldots, a_l)).$$

On the other hand, if $a_l \preceq \max(\bar{K}) =: j$ and $l > 0$, then we define

$$\hat{S}(v, (a_1, \ldots, a_l)) := ((K \cup \{j\}) \cup U), (a_1, \ldots, a_l, j))$$

and

$$\hat{P}(v, (a_1, \ldots, a_l)) := (P_{k+l}(v), (a_1, \ldots, a_l)).$$

From the above definitions $\hat{S}$ and $\hat{P}$, the every bad solution is removed from a sink and a standard source. This completes the application of the technique by Hollender and Goldberg [HG18]. We can immediately see that their method removes every bad solution. In Figure 4.4, we show an example of their method for $k = 2$. What remains is to define the potential function.

Finally, we define the new potential function $\hat{V}$ to complete the reduction. For each vertex $(v, (a_1, \ldots, a_l))$, we define

$$\hat{V}(v, (a_1, \ldots, a_l)) := 2^h \chi_{even}(l) + V_{k+l}(v),$$

where $\chi_{even}(l)$ is 1 if l is even, and is 0 if l is odd. Exceptionally, we define $\hat{V}(\Pi', ()) = 0$.

It is easy to see that this construction is polynomial-time computable. The tuple $(\hat{S}, \hat{P}, \hat{V}, \Pi')$ is a valid instance of End of Potential Line. Moreover, we can efficiently extract an original solution from a sink and a non-standard source on $\hat{S}$ and $\hat{P}$.

What remains is to verify that each valid arc added to remove a bad solution is an increasing arc. We prove that each bad solution is not a violating solution for $(\hat{S}, \hat{P}, \hat{V}, \Pi')$, and we complete the polynomial-time reduction from k-END OF POTENTIAL LINE to END OF POTENTIAL LINE.

First, if l is even, then every bad solution $(v, (a_1, \ldots, a_l))$ is a non-standard source for S_{k+l} and P_{k+l}. In particular, it satisfies that $V_{k+l}(v) = 0$. Therefore, the vertex $(v, (a_1, \ldots, a_l))$ is assigned the potential $\hat{V}(v, (a_1, \ldots, a_l)) = 2^h$. Now, let $(w, \tau) = \hat{P}(v, (a_1, \ldots, a_l))$. By the definition of $\hat{V}$, it holds that $\hat{V}(w, \tau) = 2^h - 1 < 2^h = \hat{V}(v, (a_1, \ldots, a_l))$. Moreover, it satisfies that $(w, \tau) = \hat{P}(v, (a_1, \ldots, a_l))$ and $\hat{S}(w, \tau) = (v, (a_1, \ldots, a_l))$. That is, the vertex $(v, (a_1, \ldots, a_l))$ is not a solution.

Next if l is odd, the every bad solution $(v, (a_1, \ldots, a_l))$ is a sink for S_{k+l} and P_{k+l}. In particular, it satisfies that $V_{k+l}(v) = 2^h - 1$. Thus, it holds that $\hat{V}(v, (a_1, \ldots, a_l)) = 2^h - 1$. Now, let $(w, \tau) = \hat{S}(v, (a_1, \ldots, a_l))$. By the definition of $\hat{V}$, it holds that $\hat{V}(w, \tau) = 2^h > 2^h - 1 = \hat{V}(v, (a_1, \ldots, a_l))$. Moreover, it satisfies that $(w, \tau) = \hat{S}(v, (a_1, \ldots, a_l))$ and $\hat{P}(w, \tau) = (v, (a_1, \ldots, a_l))$. Hence, the vertex $(v, (a_1, \ldots, a_l))$ is not a solution. □

Immediately, we get EOPL-completeness of k-END OF POTENTIAL LINE by combining the above lemmas.

Theorem 4.30. *k-EOPL is EOPL-complete.*

4.4 The Hardness of Parity Argument with Potential

In this section, we study the complexity of the parity argument with potential. As mentioned in Section 3.3.6, the class EOPL is characterized by END OF POTENTIAL LINE. Up to this point, we show that the classification by this problem is robust. Recall the definition of END OF POTENTIAL LINE. This problem can be viewed as the problem that each instance of END OF LINE relaxed by a potential condition, in which every valid arc is an increasing arc. Incidentally, every undirected graph with potential can be introduced to a natural orientation by its potential. The natural question arises. How hard is a PPA-complete problem which is given a potential condition, e.g., the problem POTENTIAL LEAF (see Definition 4.31)? Also, is the classification based on such a problem robust? In Section 4.4.1, we show that the problem, in which every instance of LEAF is given a potential condition, is EOPL-complete. However, the problem, in which every instance of ODD is given a potential condition, is much harder than END OF POTENTIAL LINE, specifically, this problem is PPA∩PLS-complete (see Section 4.4.2 for details).

4.4.1 The Problem: POTENTIAL LEAF

In this section, we introduce a new problem called POTENTIAL LEAF. This problem is the most simple generalization of END OF POTENTIAL LINE. Informally speaking, this problem is defined as: given an implicit undirected graph that every vertex has degree at most two, a potential function, and a known leaf, find at least one of another leaf and a local optimum solution. Intuitively, this problem is EOPL-hard. We can construct a polynomial-time reduction by replacing every valid arc on an instance of END OF POTENTIAL LINE by an undirected edge. Not surprisingly, the

problem POTENTIAL LEAF is polynomially equivalent to the problem END OF POTENTIAL LINE; we can introduce a natural direction to each edge on an instance of POTENTIAL LEAF.

Definition 4.31. POTENTIAL LEAF
Input:

- an implicit graph with potential $G(N,V) = (\Sigma^n, E)$ defined by
  - a Boolean circuit $N : \Sigma^n \to \Sigma^{2n}$ and
  - a Boolean circuit $V : \Sigma^n \to \{0,1,\ldots,2^m-1\}$
- a vertex $\pi \in \Sigma^n$ such that $\deg_G(\pi) = 1$ and $V(\pi) = 0$.

Task: Find a non-isolated vertex satisfying at least one of the following:

(R1) $x \neq \pi$ and $\deg_G(x) = 1$,

(R2) $x \neq \pi$ and $V(x) \leq V(y)$ for every valid edge $\{x,y\}$ in E, and

(R3) $V(x) \geq V(y)$ for every valid edge $\{x,y\}$ in E

Theorem 4.32. POTENTIAL LEAF *is an* EOPL*-complete problem.*

Proof. By the definition of POTENTIAL LEAF, it is straightforward to see that this problem is EOPL-hard. Therefore, it suffices to prove that POTENTIAL LEAF is polynomial-time reducible to END OF POTENTIAL LINE.

Let (n,m,N,V,π) be a valid instance of POTENTIAL LEAF. Without loss of generality, we assume that the neighborhood function N always computes valid edges, i.e., it satisfies that for each pair of vertices x and y in Σ^n, $y \in N(x)$ if and only if $x \in N(y)$.

We construct the successor circuit S and the predecessor circuit P as follows. Let $N(\pi) = \{\rho\}$, where $\rho \neq \pi$. We first define $S(\pi) := \rho$ and $P(\pi) = \pi$. For every vertex $x \in \Sigma^n$ with $x \neq \pi$, we define $S(x) := \arg\max\{y \in N(x); V(y) > V(x)\}$ and $P(x) := \arg\max\{y \in N(x); V(y) < V(x)\}$. However, select exactly one that is lexicographically small if there are two vertices that have the same potential. Furthermore, if $\arg\max\{y \in N(x); V(y) > V(x)\}$ is empty, then the successor circuit S outputs x, similarly if $\arg\max\{y \in N(x); V(y) < V(x)\}$ is empty, then the successor circuit P outputs x.

From the above definitions, it holds that $P(\pi) = \pi \neq S(\pi)$ and $V(\pi) = 0$. Therefore, the tuple (n,m,S,P,V,π) is a valid instance of END OF POTENTIAL LINE. In the rest of this proof, we show that when we obtain a solution of (n,m,S,P,V,π), we can convert to an original solution in polynomial time.

First, we consider when we obtain a solution $x \in \Sigma^n$ satisfying that $P(S(x)) \neq x$. If $S(x) = x$, then for every $y \in N(x)$, it holds that $V(y) \leq V(x)$ by definition. Thus, x is a local maximum solution. Otherwise, i.e., $S(x) \neq x$, then it satisfies that $V(S(x)) > V(x)$. Since $P(S(x)) \neq x$, there is a vertex $z \in N(S(x))$ such that $V(z) \leq V(x) < V(S(x))$ and $z \neq x$. Note that $x \in N(S(x))$, and thus, $N(S(x)) = \{x,z\}$. Hence, $S(x)$ is a local maximum solution since $V(S(x)) > V(x)$ and $V(S(x)) > V(z)$.

Second, we consider when we obtain a solution x in Σ^n satisfying that $S(P(x)) \neq x \neq \pi$. If $P(x) = x$, then for every $y \in N(x)$, it holds that $V(y) \geq V(x)$ by definition.

This implies that x is a local minimum solution. Otherwise, i.e., $P(x) \neq x$, then it satisfies that $V(P(x)) < V(x)$. Furthermore, there exists a vertex $z \in N(P(x))$ such that $V(P(x)) < V(x) \leq V(z)$ and $z \neq x$ since $S(P(x)) \neq x$. Note that $x \in N(P(x))$, and thus, $N(P(x)) = \{x, z\}$. Hence, $P(x)$ is a local minimum solution since $V(P(x)) < V(x)$ and $V(P(x)) < V(z)$.

Finally, it is easy to see that there are no violating solutions for the instance (n, m, S, P, V, π). From the above arguments, we complete the polynomial-time reduction from POTENTIAL LEAF to END OF POTENTIAL LINE. Therefore, the problem POTENTIAL LEAF is an EOPL-complete problem. □

4.4.2 The Problem: POTENTIAL ODD

In Section 4.4.1, we have only considered graphs where every vertex has degree at most two. Now, we generalize the problem POTENTIAL LEAF to a new search problem on higher degree graphs, called POTENTIAL ODD. This problem is defined as follows.

Definition 4.33. POTENTIAL ODD
Input:

- an implicit graph with potential $G(N, V) = (\Sigma^n, E)$ defined by
  - a Boolean circuit $N : \Sigma^n \to \Sigma^{dn}$ and
  - a Boolean circuit $V : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$
- an odd-degree vertex $\pi \in \Sigma^n$ on $G(N, V)$.

Task: Find a non-isolated vertex $x \in \Sigma^n$ satisfying at least one of the following:

(R1) $x \neq \pi$ and $\deg_G(x) = 2k + 1$ for some non-negative integer k,

(R2) $V(x) \geq V(y)$ for every $y \in N(x)$ with $\{x, y\} \in E$, and

(R3) $V(x) \leq V(y)$ for every $y \in N(x)$ with $\{x, y\} \in E$.

It is clear to see that POTENTIAL ODD belongs to the class PPA ∩ PLS. Recall that the parity argument guarantees that every finite graph has an even number of odd-degree vertices, that is, at least one unknown odd-degree vertex exists when we have a known odd-degree vertex. Naturally, this principle works for every finite graph with potential. Moreover, we can find a local minimum or maximum vertex by applying a local search method.

Therefore, in the rest of this thesis, we focus on the hardness of this problem. First, we show that POTENTIAL ODD is, generally, PPA ∩ PLS-hard, and thus, this problem is a PPA ∩ PLS-complete problem.

Theorem 4.34. POTENTIAL ODD *is a* PPA ∩ PLS*-complete problem.*

Proof. We first show that POTENTIAL ODD belongs to PPA ∩ PLS. Let $\mathscr{I} = (n, m, d, N, V, \pi)$ be a valid instance of POTENTIAL ODD. Then, the tuple (n, d, N, π) is a valid instance of ODD. Furthermore, every solution of (n, d, N, π) is also a solution to $\mathscr{I}$. Hence, POTENTIAL ODD is in PPA.

We prove that POTENTIAL ODD also belongs to PLS. To prove this, we construct a polynomial-time reduction from POTENTIAL ODD to LOCALOPT. Without loss of generality, we assume that the known odd-degree vertex π has positive potential, i.e., $V(\pi) > 0$. If $V(\pi) = 0$, then π is obviously a solution to $\mathscr{I}$. We consider the implicit graph with potential $G(N,V) = (\Sigma^n, E)$ produced by N and V, where E is a set of all valid edges for N. Then, the function $f : \Sigma^n \to \Sigma^n$ is defined as follows. For every vertex $x \in \Sigma^n$,

$$f(x) = \arg\max\{V(y); y \in N(x) \text{ and } \{x,y\} \in E\}.$$

Specifically, select exactly one lexicographically small if there are some vertices with the same potential. If a vertex $x \in \Sigma^n$ is an isolated vertex, we define $f(x) = \pi$ and replace its potential by 0, i.e., $V(x) = 0$. Note that an isolated vertex x is not a local maximum for f from our assumption that $V(\pi) > 0$. The tuple (f,V) is a valid instance of LOCALOPT. Every solution $x \in \Sigma^n$ to (f,V) satisfies that $V(f(x)) \leq V(x)$. We have that $V(y) \leq V(x)$ for each $y \in N(x)$ with $\{x,y\} \in E$, and thus, x is also a solution to $\mathscr{I}$. Therefore, POTENTIAL ODD belongs to PLS.

From the above arguments, POTENTIAL ODD is in PPA$\cap$PLS.

To prove that POTENTIAL ODD is PPA$\cap$PLS-hard, we show that EITHER SOLUTION(ODD, FLIP) is reducible to POTENTIAL ODD in polynomial time. Let $\mathscr{I} = (n_1, d, N, \pi, n_2, m_1, C)$ be a valid instance of EITHER SOLUTION(ODD, FLIP). Then, the tuple (n_1, d, N, π) is a valid instance of ODD. Here, we denote by $G(N) = (\Sigma^{n_1}, E)$ the implicit graph produced by N, and let k_0 be a non-negative integer such that $\deg_G(\pi) = 2k_0 + 1$. The tuple (n_2, m_1, C) is a valid instance of FLIP, where the given Boolean circuit C computes a polynomial-time computable function $f : \Sigma^{n_2} \to \{0, 1, \ldots, 2^{m_1} - 1\}$. Without loss of generality, we assume that n_2 is even.

From now on, we construct a valid instance $\mathscr{J} = (n_1 + n_2, m_1, n_2 + d, N', V', \pi')$ of POTENTIAL ODD. We first denote by $U = \Sigma^{n_1} \times \Sigma^{n_2}$ the new vertex set. For each $x = (x_1, x_2) \in U$, we define

$$N'(x) := \{(x_1, y_2) \in U; \|x_2 - y_2\| = 1\} \cup \{(y_1, x_2) \in U; x_2 = 0^{n_2} \text{ and } \{x_1, y_1\} \in E\}.$$

Note that $|N'(x)| \leq n_2 + \deg_G(x) \leq n_2 + d$ for every vertex $x \in U$, and thus, it is bounded by some polynomial in n_1 and n_2. This implies that the function N' is polynomial-time computable. Furthermore, we define the potential function as $V'(x_1, x_2) = C(x_2)$ for every $(x_1, x_2) \in U$, and let $\pi' = (\pi, 0^{n_2})$. We denote by $H(N', V') = (U, F)$ the implicit graph with potential induced by N' and V', where F is a set of all valid edges defined by N'. Then, it satisfies that $\deg_H(\pi') = n_2 + 2k_0 + 1$. Since n_2 is even and k_0 is a non-negative integer, π' is an odd-degree vertex on the graph H. Therefore, the tuple $\mathscr{J} = (n_1 + n_2, m_1, n_2 + d, N', V', \pi')$ is a valid instance of POTENTIAL ODD. What remains is to prove that we can extract a solution to $\mathscr{I}$ in polynomial time when we obtain a solution to $\mathscr{J}$.

We first consider when we obtain a vertex $x = (x_1, x_2) \in U \setminus \{\pi'\}$ satisfying that $\deg_H(x) = 2k + 1$, where k is a non-negative integer. For every vertex $z = (z_1, z_2) \in U$ satisfying that $z_2 \neq 0^{n_2}$, it holds that $\deg_H(z) = n_2$ since $N'(x) = \{(z_1, y_2) \in U; \|z_2 - y_2\| = 1\}$. Assuming that n_2 is even, z is an even-degree vertex. Therefore, it must hold that $x_2 = 0^{n_2}$. Then, it satisfies that $\deg_H(x) = n_2 + \deg_G(x_1)$, and thus, $\deg_G(x_1)$

is odd since n_2 is even and $\deg_H(x)$ is odd. This implies that the vertex x_1 has odd degree. Furthermore, it holds that $x_1 \neq \pi$ since $x = (x_1, 0^{n_2}) \neq \pi' = (\pi, 0^{n_2})$. Hence, the vertex x_1 is a solution to $\mathscr{I}$.

Next, we consider when we obtain a vertex $x = (x_1, x_2) \in U$ satisfying that $V'(x) \leq V'(y)$ for every vertex $y = (y_1, y_2) \in N'(x)$. By the definition of V' and N', it holds that $C(x_2) \leq C(y_2)$ for every y_2 with $\|x_2 - y_2\| = 1$. Hence, x_2 is a solution for FLIP, and it is a solution to $\mathscr{I}$. Similarly, when we obtain a vertex $x = (x_1, x_2) \in U$ satisfying that $V'(x) \geq V'(y)$ for every vertex $y = (y_1, y_2) \in N'(x)$, the string x_2 is a solution to $\mathscr{I}$.

From the above arguments, we complete the polynomial-time reduction from EITHER SOLUTION(ODD, FLIP) to POTENTIAL ODD. Therefore, POTENTIAL ODD is PPA $\cap$ PLS-hard. □

4.4.3 Variants of ODD with Potential

In this section, we scrutinize the complexity classification of these problems in more detail. In the previous sections, we show that POTENTIAL LEAF, which is associated with graphs whose every vertex has degree at most two, is an EOPL-complete problem. However, POTENTIAL ODD, which is associated with higher degree graphs, is a PPA $\cap$ PLS-complete problem.

The most natural question is where the boundaries of the complexity of POTENTIAL ODD lie. We focus on the maximum degree of a given implicit graph. Now, we consider the constant degree variant of POTENTIAL ODD. In this problem, the maximum degree of the implicit graph is bounded by some constant. We prove that even if the maximum degree 4, then POTENTIAL ODD is PPA $\cap$ PLS-complete, but if the maximum degree is at most 3, then POTENTIAL ODD is EOPL-complete.

Definition 4.35. DEGREE-d POTENTIAL ODD
Input:

- a Boolean circuit $N : \Sigma^n \to \Sigma^{dn}$
- a Boolean circuit $V : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$
- an odd-degree vertex $\pi \in \Sigma^n$ on $G(N, V)$.

Task: Find a non-isolated vertex $x \in \Sigma^n$ satisfying at least one of the following:

(R1) $x \neq \pi$ and $\deg_G(x) = 2k + 1$ for some non-negative integer k,

(R2) $V(x) \geq V(y)$ for every $y \in N(x)$ with $\{x, y\} \in E$, and

(R3) $V(x) \leq V(y)$ for every $y \in N(x)$ with $\{x, y\} \in E$.

In order to simplify the following arguments, we prove the following proposition.

Proposition 4.36. *Without loss of generality, we assume that for every instance of* POTENTIAL ODD*, each vertex has a different potential.*

Proof. To prove this, we show that when we are given a valid instance $\mathscr{I} = (n, m, d, N, V, \pi)$ of POTENTIAL ODD, we can construct the another instance $\mathscr{I}' = (n, m + n, N, V', \pi)$ that satisfies that $V'(x) \neq V'(y)$ for each pair of vertices $x, y \in \Sigma^n$ with $x \neq y$. Moreover, we can convert from a solution from $\mathscr{I}'$ to a solution of $\mathscr{I}$ in polynomial time.

For every string x in Σ^n, we define

$$V'(x) := 2^n V(x) + \sum_{i=1}^{n} 2^{i-1} x_i,$$

where let $x = x_1 x_2 \ldots x_n$. Clearly, it satisfies that $V'(x) \neq V'(y)$ for each pair of vertices $x, y \in \Sigma^n$ with $x \neq y$. Furthermore, for each pair of $x, y \in \Sigma^n$, if $V'(x) < V'(y)$ then $V(x) \leq V(y)$. Therefore, if a vertex x in Σ^n satisfying that $V'(x) \geq V'(y)$ for every $y \in N(x)$ with $\{x, y\} \in E$, then x is a solution of $\mathscr{I}$. Similarly, if a vertex x in Σ^n satisfying that $V'(x) \leq V'(y)$ for every $y \in N(x)$ with $\{x, y\} \in E$, then x is a solution of $\mathscr{I}$. Also, a vertex x is an odd-degree vertex for $\mathscr{I}'$ if and only if x is an odd-degree vertex for $\mathscr{I}$. □

Notice that our proof of Proposition 4.36 also works for DEGREE-d POTENTIAL ODD. Without loss of generality, we assume that for every instance of DEGREE-d POTENTIAL ODD, every vertex on the given implicit graph has a different potential.

First, we show that DEGREE-4 POTENTIAL ODD is a PPA∩PLS-complete problem.

Theorem 4.37. DEGREE-4 POTENTIAL ODD *is* PPA∩PLS*-complete problem.*

Proof. It is easy to see that DEGREE-4 POTENTIAL ODD belongs to PPA∩PLS. Therefore, it suffices to prove that DEGREE-4 POTENTIAL ODD is PPA∩PLS-hard. To prove this, we show that there exists a polynomial-time reduction from POTENTIAL ODD to DEGREE-4 POTENTIAL ODD.

The fundamental idea of our proof is that we simulate every vertex with degree > 4 by some vertices with degree ≤ 4. Before beginning our proof, we illustrate the main idea. See Figure 4.5 for an illustration of the construction. In this figure, the vertex v has degree > 4 on the left graph, and we assume that the original potential function V satisfies that $V(x_1) < V(x_2) < V(x_4) < V(x_5)$. We construct the right graph from the left one. The red vertices on the right graph simulate the vertex v. All adjacent vertices of v are placed adjacent to v_0, v_1, and v_2 in order of increasing potential. We define the new potential function V' such that $V'(v_0) < V'(v_1) < V'(v_2)$ and $V'(v_0) < V'(v_*) < V'(v_2)$. Furthermore, V' follows that $V'(v_j) > V'(x_i)$ if and only if $V(v) > V(x_i)$ for all i and j. From our construction, it is easy to see that the following three properties hold:

- v is a local minimum if and only if v_0 is a local minimum,
- v is a local maximum if and only if v_2 is a local maximum, and
- v has an odd degree if and only if v_2 has an odd degree.

We do this for all vertices on the original graph. Notice that every red vertex has degree at most 4. This construction allows us to simulate a vertex with degree > 4

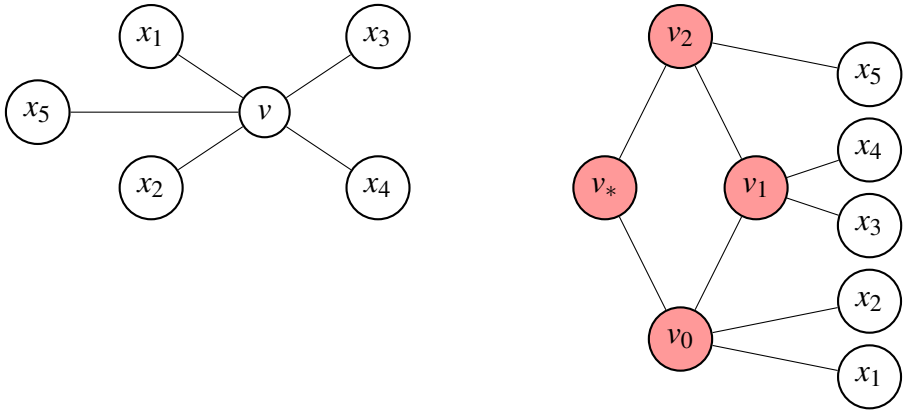

FIGURE 4.5: Illustration of construction

by some vertices with degree ≤ 4. In the rest of this proof, we show that the above construction is polynomial-time computable.

Let $\mathscr{I} = (n, m, d, N, V, \pi)$ be a valid instance of POTENTIAL ODD. Without loss of generality, we assume that every vertex of the instance $\mathscr{I}$ has a different potential, i.e., $V(x) \neq V(y)$ for each pair of vertices $x, y \in \Sigma^n$ with $x \neq y$. We denote by $G(N,V) = (\Sigma^n, E)$ the implicit graph with potential produced by N and V, where E is a set of all valid edges defined by N.

We consider the following vertex set

$$U := \hat{U} \cup \{(*, x); x \in \Sigma^n \text{ and } \deg_G(x) \geq 3\},$$

where $*$ is a special symbol, and the set $\hat{U}$ is defined as follows:

$$\hat{U} := \{(i, x); x \in \Sigma^n \text{ and } 1 \leq i \leq \left\lceil \frac{\deg_G(x)}{2} \right\rceil\}.$$

We construct the new neighborhood function $N' : U \to U^{4d}$ and the new potential function $V' : U \to \{0, 1, \ldots, 2^{2m+d}\}$. Before we construct these two functions, we define the following three functions. First, for each vertex x in Σ^n and each valid edge $\{x, y\}$ in E, we define

$$rank_x(\{x, y\}) := 1 + |\{\{x, z\} \in E; V(z) < V(y)\}|.$$

Note that this function is bijective. Second, for each vertex x in Σ^n and each vertex y in Σ^n with $\{x, y\}$ in E, we define

$$\lambda_x(y) = \left\lceil \frac{rank_x(\{x, y\})}{2} \right\rceil.$$

Finally, for each vertex (i, x) in $\hat{U}$, we define

$$\hat{N}(i, x) := \{(j_1, y_1), (j_2, y_2)\},$$

where it satisfies that $\lambda_x(y_1) = \lambda_x(y_2) = i$ and $j_k = \lambda_{y_k}(x)$ for each $k = 1, 2$. If there is only one vertex y in Σ^n such that $\lambda_x(y) = i$, then we define $\hat{N}(i, x) := \{(j, y)\}$, where

it satisfies that $j = \lambda_y(x)$. If there are no vertices y in Σ^n such that $\lambda_x(y) = i$, we define that $\hat{N}(i,x)$ is empty. However, for each $(*,x)$ in U, we define $\hat{N}(*,x) := \emptyset$. By using the function $\hat{N}$, for each vertex (i,x) in U, the neighborhood function N' is defined as

$$N'(i,x) = \hat{N}(i,x) \cup \{(i+1,x); i < M_x\} \cup \{(i-1,x); i > 0\} \\ \cup \{(*,x); i \in \{1, M_x\}\} \cup \{(1,x), (M_x,x); M_x \neq 1, i = *\},$$

where $M_x = \left\lceil \frac{\deg_G(x)}{2} \right\rceil$. Next, for each (i,x) in U, we define

$$V'(i,x) := \begin{cases} 2^{2m}V(x) + 2(i-1) & \text{if } i \in [M_x], \\ 2^{2m}V(x) + 1 & \text{if } i = *. \end{cases}$$

It is easy to see that given a string (i,x) in $[\deg(G)] \times \Sigma^n$ then we can check whether (i,x) is in U in polynomial time. Hence, the neighborhood function N' and the potential function V' is polynomial-time computable and constructed. Furthermore, it is not hard to see that every vertex (i,x) in U has degree at most 4, i.e., it satisfies that $|N'(i,x)| \leq 4$.

From the construction of N', the following claim holds.

Lemma 4.38. *A vertex (i,x) in U is an odd-degree vertex on H if and only if it satisfies that $i = M_x$ and x is an odd-degree vertex on G.*

Proof. We first show that if a vertex (i,x) in U is an odd-degree vertex on the graph H, then x is an odd-degree vertex on G by using a contradiction. Assume that x has even degree on G, i.e., there is a non-negative integer k such that $\deg_G(x) = 2k$. Then for each $i' \in [k]$, there exist two distinct vertices z_1 and z_2 such that $rank_x(\{x,z_1\}) = 2i'-1$ and $rank_x(\{x,z_2\}) = 2i'$. Hence, it holds that $N'(i',x) = \{(j_1,z_1),(j_2,z_2)\}$, where $\lambda_{z_l}(x) = j_l$ for each $l = 1,2$. This is a contradiction from (i,x) has odd degree on H.

Therefore, x is an odd-degree vertex on G, and thus, there is a non-negative integer k' such that $\deg_G(x) = 2k'+1$. Then it satisfies that $M_x = k'+1$. Assuming that $i < M_x$, there exist two distinct vertices z_1 and z_2 such that $rank_x(\{x,z_1\}) = 2i-1$ and $rank_x(\{x,z_2\}) = 2i$. This implies that (i,x) has even degree on H. This is a contradiction. Hence, it satisfies that $i = M_x$. Then there is only one vertex z such that $rank_x(\{x,z\}) = 2k'+1$.

Next, we show that if $i = M_x$ and a vertex x in Σ^n is an odd-degree vertex on G, then a vertex (M_x,x) in U is an odd-degree vertex on H. Since x is an odd-degree vertex on G, there is a non-negative integer k'' such that $\deg_G(x) = 2k''-1$. Therefore, it holds that

$$\left\lceil \frac{(2k''-1)-1}{2} \right\rceil < \left\lceil \frac{2k''-1}{2} \right\rceil = k'' = M_x.$$

Therefore, there is only one vertex y in Σ^n such that $\lambda_x(y) = k''$. This implies that the vertex (M_x,x) is an odd-degree vertex on H. □

Immediately, it is easy to see that $\pi' := (M_\pi, \pi)$ has odd degree on the graph H from the above claim. Hence, the tuple $\mathscr{J} = (n+d+1, 2m+d, N', V', \pi')$ is a valid

instance of DEGREE-4 POTENTIAL ODD. In the rest of this proof, we show that when we obtain a solution of $\mathscr{J}$, we can extract an original solution.

We first consider when we obtain an odd-degree vertex $(i,x) \neq \pi'$ on the graph H. From the above claim, it is easy to see that $i = M_x$, $x \neq \pi$, and x is an odd-degree vertex on the graph G. Therefore, x is a solution for the original instance $\mathscr{J}$.

Next, we consider when we obtain a vertex (i,x) that is a local minimum solution on H, i.e., it satisfies that $V'(i,x) \leq V'(j,y)$ for every vertex $(j,y) \in U$ with $\{(i,x),(j,y)\} \in F$. Note that $i \neq *$ since it satisfies that $V'(1,x) < V'(*,x) < V'(M_x,x)$ and $N'(*,x) = \{(1,x),(M_x,x)\}$. Similarly, we can see that $i = 1$. We denote by y_{min} the vertex in U that has the minimum potential among vertices satisfying that $\lambda_x(y) = 1$. Since the vertex (i,x) is a local minimum solution on H, it holds that $V'(i,x) \leq V'(j_{min}, y_{min})$. Thus, $V(x) \leq V(y_{min})$ holds. Therefore, it holds that $V(x) \leq V(y_{min}) \leq V(y)$ for every vertex y in Σ^n with $\{x,y\} \in E$ since y_{min} has the minimum potential among neighbors of x on the graph G. This implies that the vertex x is a solution of the original instance.

From the similar argument, we show that if a vertex (i,x) is a local maximum solution on H, i.e., it satisfies that $V'(i,x) \geq V'(j,y)$ for every vertex $(j,y) \in U$ with $\{(i,x),(j,y)\} \in F$, then the vertex x is a local maximum solution on G, and thus, x is a solution of the original instance.

Therefore, we complete the polynomial-time reduction from POTENTIAL ODD to DEGREE-4 POTENTIAL ODD. Since POTENTIAL ODD is a PPA∩PLS-complete problem, this implies that DEGREE-4 POTENTIAL ODD is PPA∩PLS-hard. Hence, this problem is also PPA∩PLS-complete. □

Next, we prove that DEGREE-3 POTENTIAL ODD is EOPL-complete. We firstly show the EOPL-hardness of this problem. Our proof gives a polynomial-time reduction from POTENTIAL LEAF to DEGREE-3 POTENTIAL ODD. Although every instance of POTENTIAL LEAF is also an instance of DEGREE-3 POTENTIAL ODD, the known leaf π given an instance of POTENTIAL LEAF is a local minimum vertex, i.e., π is a solution for DEGREE-3 POTENTIAL ODD. Therefore, it is necessary to avoid the known leaf becoming a solution for DEGREE-3 POTENTIAL ODD. For this, we make three-copies of the original instance, and we add a vertex with degree three as a known odd-degree vertex.

Lemma 4.39. POTENTIAL LEAF *is reducible to* DEGREE-3 POTENTIAL ODD *in polynomial time.*

Proof. To prove EOPL-hardness, we construct a polynomial-time reduction from POTENTIAL LEAF to DEGREE-3 POTENTIAL ODD. Our reduction is simple. Let $\mathscr{I} = (n,m,N,V,\pi)$ be a valid instance of POTENTIAL LEAF. Without loss of generality, we assume that the neighborhood function N always computes valid edges, i.e., it satisfies that for each pair of vertices x and y in Σ^n, $y \in N(x)$ if and only if $x \in N(y)$.

We make three copies of the instance $\mathscr{I}$; $\mathscr{I}_1$, $\mathscr{I}_2$, and $\mathscr{I}_3$, respectively. We denote by π_j the known leaf of $\mathscr{I}_j$, where $j = 1,2,3$. We invert the potential of all vertices on $\mathscr{I}_1$, i.e., we define it as $(2^m - 1) - V(x)$ for every vertex x. In the other two instances, we redefine the potential as $2^{m+2} \cdot V(x)$ for every vertex x. Moreover, we add a vertex π^*. The potential of this vertex is 2^{m+1}. The vertex π^* is adjacent to the known leaves of each copied instance π_1, π_2, and π_3. That is, π^* is a vertex with degree three. Note that the known leaf on the original instance is not a solution.

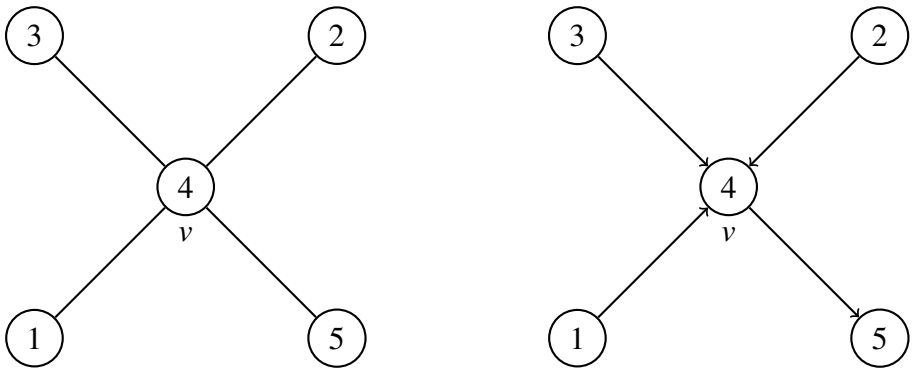

FIGURE 4.6: Left: An original undirected graph. Right: A directed graph which naturally induced from the left one. This figure shows that x is an even-degree vertex on the left graph but x is an unbalanced vertex on the right graph.

Furthermore, π^* is a known odd-degree vertex on the new instance, and it is not a local optimum. It is easy to see that we can extract an original solution from a solution to the new instance. □

It is easy to see that DEGREE-3 POTENTIAL ODD belongs to EOPL; we can apply the same technique used in the proof of Theorem 4.32. We naturally define every valid arc depending on the potential function. Note that, in the case of degree three, each unbalanced vertex is an odd-degree vertex, and thus, our simple technique works well. Unfortunately, this simple technique does not work to show that DEGREE-4 POTENTIAL ODD belongs to EOPL. See Figure 4.6. The digraph in the right of this figure is naturally induced from the left one. In the right graph, the vertex v is an unbalanced vertex, that is, v is a solution for POTENTIAL IMBALANCE. However, v has an even degree on the left one, that is, v is not a solution for POTENTIAL ODD. Therefore, new ideas seem necessary to prove that DEGREE-4 POTENTIAL ODD belongs to EOPL.

In the rest of this thesis, we consider the more general potential condition and its complexity. In other words, we provide a potential condition for POTENTIAL ODD belonging to EOPL. This condition gives us a limitation that the natural orientation works well for proving that the problem on undirected graphs with potential belongs to EOPL. Informally speaking, the potential condition introduced in the following means that we can well-pair the neighbors of every even-degree vertex that is not local optimum.

Let $G(N,V) = (\Sigma^n, E)$ be an implicit graph with potential produced by a neighborhood function N and a potential function V. We define $E_V^+(x) := \{y \in U; \{x,y\} \in E \text{ and } V(y) > V(x)\}$ and $E_V^-(x) := \{y \in U; \{x,y\} \in E \text{ and } V(y) < V(x)\}$ for each vertex x in Σ^n. A vertex x in Σ^n is said to be well balanced if x satisfies that

$$|E_V^+(x)| = |E_V^-(x)|.$$

We say that the graph $G(N,V)$ is almost balanced if every even-degree vertex that is not local optimum on $G(N,V)$ is well balanced. In Figure 4.6, the vertex v is not well balanced. We consider the problem involving such vertices as violations. Hence,

the problem called ALMOST BALANCED ODD requires that the implicit graph with potential given by instance is almost balanced. This problem is defined as follows.

Definition 4.40. ALMOST BALANCED ODD
Input:

- an implicit graph with potential $G(N,V) = (\Sigma^n, E)$ defined by
  - a Boolean circuit $N : \Sigma^n \to \Sigma^{dn}$ and
  - a Boolean circuit $V : \Sigma^n \to \{0,1,\ldots,2^m-1\}$
- an odd-degree vertex $\pi \in \Sigma^n$ on $G(N,V)$.

Task: Find a non-isolated vertex $x \in \Sigma^n$ satisfying at least one of the following:

(R1) $x \neq \pi$ and $\deg_G(x) = 2k+1$ for some non-negative integer k,

(R2) $V(x) \geq V(y)$ for every $y \in N(x)$ with $\{x,y\} \in E$, and

(R3) $V(x) \leq V(y)$ for every $y \in N(x)$ with $\{x,y\} \in E$.

(V1) $|E_V^+(x)| \neq 0 \neq |E_V^-(x)| \neq |E_V^+(x)|$, and $\deg_G(x) = 2k$ for some integer k.

We shall prove that ALMOST BALANCED ODD is EOPL-complete. First, in Lemma 4.41, we show that a polynomial-time reduction from DEGREE-3 POTENTIAL ODD to this problem, that is, we show EOPL-hardness of ALMOST BALANCED ODD. After that, in Lemma 4.42, we show that ALMOST BALANCED ODD belongs to EOPL by constructing a polynomial-time reduction to POTENTIAL IMBALANCE. Of course, our proof implies that DEGREE-3 POTENTIAL ODD is also EOPL-complete.

Lemma 4.41. DEGREE-3 POTENTIAL ODD *is reducible to* ALMOST BALANCED ODD *in polynomial time.*

Proof. Each valid instance $\mathscr{I} = (n,m,N,V,\pi)$ of DEGREE-3 POTENTIAL ODD is also a valid instance of ALMOST BALANCED ODD. We denote by $G(N,V) = (\Sigma^n, E)$ the implicit graph with potential produced by N and V. We can easily see that every odd-degree vertex and every local optimum vertex on $G(N,V)$ are a solution to $\mathscr{I}$. Therefore, it is sufficient to show that $\mathscr{I}$ has no violating solutions. We consider an even-degree vertex x that is not local optimum on $G(N,V)$. It satisfies that $|E_V^+(x)| \neq 0 \neq |E_V^-(x)|$. Notice that x has even degree and $\mathscr{I}$ is an instance of DEGREE-3 POTENTIAL ODD, we have $|E_V^+(x)| + |E_V^-(x)| = 2$, and thus, $|E_V^+(x)| = 1 = |E_V^-(x)|$. Hence, the vertex x is a well-balanced vertex. □

Lemma 4.42. ALMOST BALANCED ODD *is reducible to* POTENTIAL IMBALANCE *in polynomial time.*

Proof. Assume that we are given an instance $\mathscr{I} = (n,m,d,N,V,\pi)$ of ALMOST BALANCED ODD. We denote by $G(N,V) = (\Sigma^n, E)$ the implicit graph with potential that is produced by N and V. For every vertex x in Σ^n, we define $S(x) := E_V^+(x)$ and $P(x) := E_V^-(x)$. Furthermore, we denote $\delta^+(x) = |E_V^+(x)|$ and $\delta^-(x) = |E_V^-(x)|$ for each vertex x in Σ^n. Then the vertex π satisfies that $\delta^+(\pi) \neq \delta^-(\pi)$ since

π has odd degree on the implicit graph with potential $G(N,V)$. Thus, the tuple $\mathscr{J} = (n,m,d,S,P,V,\pi)$ is a valid instance of POTENTIAL IMBALANCE.

By definition, there are no violating solutions to $\mathscr{J}$. Therefore, we only obtain a solution x in Σ^n satisfying that $\delta^+(x) \neq \delta^-(x)$ and $x \neq \pi$. If $\delta^+(x) = 0$, then it holds that $V(x) \geq V(y)$ for every vertex valid edge $\{x,y\} \in E$. This implies that the vertex x is a local maximum solution of the original instance $\mathscr{I}$. Similarly, if $\delta^-(x) = 0$, then the vertex x is a local minimum solution of $\mathscr{I}$.

In the following, we consider the case where $\delta^+(x) \neq 0 \neq \delta^-(x)$. If x is an odd-degree vertex on the original instance, it is obviously a solution to $\mathscr{I}$. Otherwise, we obtain a violating solution to $\mathscr{I}$ because $|E_V^+(x)| \neq 0 \neq |E_V^-(x)| \neq |E_V^+(x)|$ and x is an even-degree vertex. □

Immediately, the following two theorems follow.

Theorem 4.43. DEGREE-3 POTENTIAL ODD *is* EOPL-*complete.*

Theorem 4.44. ALMOST BALANCED ODD *is* EOPL-*complete.*

Finally, we show that ALMOST BALANCED ODD can be normalized.

Lemma 4.45. ALMOST BALANCED ODD *is normalizable.*

Proof. Let $\mathscr{I} = (n,m,d,N,V,\pi)$ is a valid instance of ALMOST BALANCED ODD. For each vertex $x \in \Sigma^n$, we add d additional vertices (x,i) where $i = 1,2,\ldots,d$.

First, we define the new neighborhood function N' as follows. If a vertex $x \in \Sigma^n$ is a violating solution, $N'(x) = N(x) \cup \{(x,i); i = 1,2,\ldots,\big||E_V^+(x)| - |E_V^+(x)|\big|\}$. Otherwise, we define $N'(x) = N(x)$. Furthermore, for each additional vertex (x,i), we define $N'(x,i) = \{x\}$ if $i \leq \big||E_V^+(x)| - |E_V^+(x)|\big|$, otherwise $N'(x,i)$ is empty. Next, we define the new potential function V' as follow. For each original vertex $x \in \Sigma^n$, we define $V'(x) = V(x)$. For each additional vertex (x,i), we define

$$V'(x,i) = \begin{cases} V(x)+1 & \text{if } |E_V^+(x)| < |E_V^-|, \\ V(x)-1 & \text{if } |E_V^+(x)| > |E_V^-|, \\ V(x) & \text{if } |E_V^+(x)| = |E_V^-|. \end{cases}$$

Every violating solution x is adjacent to $\big||E_V^+(x)| - |E_V^+(x)|\big|$ additional vertices. Furthermore, the vertex x is well balanced from the definition of the new potential function V'. Therefore, the new instance has no violating solutions. Every additional vertex adjacent to x is an odd-degree vertex, and thus, we can extract a solution to the original instance efficiently. Moreover, each regular solution for the original instance is also a regular solution for the new instance. Hence, we get a normalization of ALMOST BALANCED ODD. □

Remark 4.46. Recall the definition of ALMOST BALANCED ODD. We require that every even-degree vertex which is not local optimum is well balanced. From this requirement, we can remove the non-local optimum condition; that is, we require that every even degree vertex is well balanced. Notice that the solution set of the new search problem is the same as the original problem. Therefore, the complexity of this problem is also EOPL-complete. Note that every regular solution to this new variant is an odd-degree vertex.

4.5 Conclusions and Open Problems

We have studied the complexity of several variants of END OF POTENTIAL LINE based on previous exciting work by Hollender and Goldberg [HG18]. Their technique can be extended to END OF POTENTIAL LINE. We have shown that several variants of END OF POTENTIAL LINE are also EOPL-complete. Our results imply that the classification of search problems based on END OF POTENTIAL LINE is robust.

We have extended this argument to a similar problem on an undirected graph with potential. We have proved that the undirected variant of END OF POTENTIAL LINE is generally not EOPL-complete. Specifically, DEGREE-3 POTENTIAL ODD is EOPL-complete, but DEGREE-4 POTENTIAL ODD is $\mathtt{PPA} \cap \mathtt{PLS}$-complete. These facts leave an intriguing issue about the relationship between EOPL and $\mathtt{PPA} \cap \mathtt{PLS}$. Are EOPL and $\mathtt{PPA} \cap \mathtt{PLS}$ separated? We conjecture that is true.

There are some open problems left in this thesis. The question worth considering is equivalence between $\mathtt{PPA} \cap \mathtt{PLS}$ and EOPL. Specifically, if $\mathtt{PPA} \cap \mathtt{PLS} = \mathtt{EOPL}$, then $\mathtt{EOPL} = \mathtt{CLS}$. Hence, that would resolve an important open question presented by Fearnley, Gordon, Mehta, and Savani [Fea+20]. Furthermore, we should also consider the relationship between $\mathtt{PPA} \cap \mathtt{PLS}$ and PPAD. Naturally, if $\mathtt{PPAD} \subseteq \mathtt{PPA} \cap \mathtt{PLS}$, then PPAD is a subset of PLS; this is an important open question that has been unsolved for a quarter-century. On the other hand, the containment of $\mathtt{PPA} \cap \mathtt{PLS}$ in PPAD implies that $\mathtt{PPA} \cap \mathtt{PLS} = \mathtt{PPAD} \cap \mathtt{PLS}$.

Another question worth considering is to verify whether UNIQUE END OF POTENTIAL LINE can be normalized. The problem UNIQUE END OF POTENTIAL LINE is introduced by Fearnley, Gordon, Mehta, and Savani [Fea+20]. This problem considers an instance of END OF POTENTIAL LINE that contains a "*single*" line that starts at the standard source.

Part III

Fixed Point Theory

Chapter 5

The Complexity of Fixed Point Computation

We have discussed the complexity class TFNP in Part II. Total search problems compose interesting computational complexity classes. However, the class TFNP is a semantic class; it is unlikely to have complete problems. As mentioned in Chapter 3, Papadimitriou [Pap94b] introduced some syntactical subclasses of TFNP based on Combinatorics Theory. These classes provide a way to characterize the complexity of discrete problems whose solutions are guaranteed by a combinatorial proof method.

This chapter aims to make known another approach to formulate syntactical subclasses of TFNP, which capture real-valued search problems. Let us focus on Fixed Point Theory. Consider the problem of computing a fixed point, which is guaranteed by a fixed point theorem, and its complexity. A fixed point theorem is one of the mathematical lemmata that guarantee the existence of a solution to a search problem. It is known that some fixed point theorem characterizes some complexity classes. The best well-known example is Brouwer's fixed point theorem. In fact, the problem of finding a Brouwer's fixed point is PPAD-complete [Pap94b; CD09].

Recently, in addition to Brouwer's fixed point theorem, a few fixed point theorems have been studied from the viewpoint of computational complexity. Daskalakis, Tzamos, and Zampetakis [DTZ18] have shown that the problem of finding a Banach's fixed point is complete for the class CLS (see Definition 5.4). Etessami, Papadimitriou, Rubinstein, and Yannakakis [Ete+20] have proven the problem of finding a Tarski's fixed point belongs to $\mathtt{PPAD} \cap \mathtt{PLS}$. This problem is in CLS because Fearnley, Goldberg, Hollender, and Savani [Fea+21] have shown that $\mathtt{CLS} = \mathtt{PPAD} \cap \mathtt{PLS}$. Although a fixed point theorem that characterizes PLS had been previously unknown, the author has shown that Caristi's fixed point theorem does.

In the remaining chapter, we define an arithmetic circuit, which needs to formulate continuous search problems in Section 5.1. After that, we survey the complexity of computing a fixed point in Section 5.2.

5.1 Arithmetic Circuits

An arithmetic circuit representing a function $f : \mathbb{R}^n \to \mathbb{R}^m$ consists of an acyclic digraph having the following propeties: (i) There are n input gates that are labeled with a variable; (ii) there are some input gates that are labeled with a rational number; (iii) there are m output gates; and (iv) every internal node is a binary gate and labeled with one of the operators $+$, $-$, $\times$, max, min, and $>$. Here, the operator $>$ takes a

pair of real numbers (x, y) as an input and outputs 1 if $x > y$, and otherwise outputs 0. A binary gate is a gate with fan-in two. An input and output gates are a gate with fan-in and fan-out zero, respectively.

We restrict the inputs and outputs on an arithmetic circuit into the interval $[0, 1]$. When the output returns negative, the arithmetic circuit redefines it to 0, and when the output returns a value greater than 1, the arithmetic circuit redefines it to 1. For an arithmetic circuit C, we denote by $\text{size}(C)$ the size of C, i.e., the bit-length needed to describe the arithmetic circuit C, including the rational constant used in C.

The above definition of an arithmetic circuit is based on [DP11]. It seems to be very natural, but this original model causes a subtle issue that was overlooked in [DP11].

Recently, Daskalakis and Papadimitriou have noticed this issue (see [DP20]) and proposed a way to correct it. Note that their original definition allows that an arithmetic circuit uses multiplication gates where both inputs are non-constant nodes. By exploiting this property, a circuit can function repeated squaring and produce an exponentially large number for the size of the circuit and its input. Unfortunately, it is unknown that such a circuit lies on FNP. Therefore, we must restrict the ability of an arithmetic circuit to ensure that it belongs to FNP.

While Daskalakis and Papadimitriou [DP20] proposed a restriction to achieve this, we adopt an alternative restriction introduced by Fearnley, Goldberg, Hollender, and Savani [Fea+21] to resolve the above issue in this thesis. They restricted arithmetic circuits as an instance of an FNP continuous search problem to *well-behaved* arithmetic circuits.

An arithmetic circuit C is *well-behaved* if, on any directed path that leads to an output, there are at most $\log(\text{size}(C))$ *true* multiplication gates; let a *true* multiplication gate be one where both inputs are non-constant nodes in the arithmetic circuit. As mentioned in [Fea+21], this model has two advantages over the modification by Daskalakis and Papadimitriou [DP20]: (a) There is no need to add the additional input and the extra violation solution; and (b) we can efficiently check whether a given arithmetic circuit is well-behaved.

There are some known TFNP classes defined by arithmetic circuits. Primarily, Daskalakis and Papadimitriou [DP11] presented an alternative definition of PLS. Moreover, they introduced the complexity class CLS. In the following, we describe these definitions and related works.

5.1.1 Alternative Definition of PLS

Recall that a total search problem with polynomial balance and polynomial-time verifiable restrictions is a problem in which every instance has a solution and the correctness of a solution can be verified in polynomial time in the length of the input instance (see Chapter 3). Usually, the class consisting of total search problems is called TFNP. We are interested in whether there is an efficient method to seek a solution to a problem in TFNP since such a problem has the nice property of always having a solution. Over the past decades, many total search problems have been classified by a method of finding a solution. The complexity class PLS, which Johnson, Papadimitriou, and Yannakakis [JPY88] introduced, consists of search problems that can be solved by using a local search method.

Remember that we have exhibited the formal definition of PLS based on Boolean circuits in Section 3.3.1. Now, we describe the alternative formulation of PLS based on arithmetic circuits. Consider the following computational problem, called REAL LOCALOPT:

Definition 5.1. REAL LOCALOPT
Input:

- two parameters $\varepsilon, \lambda > 0$,
- two well-behaved arithmetic circuits computing $f : [0,1]^3 \to [0,1]^3$ and $p : [0,1]^3 \to [0,1]^2$

Task: Find one of the following:

- a point $x \in [0,1]^3$ such that $p(x) \leq p(f(x)) + \varepsilon$;
- two points $x, y \in [0,1]^3$ such that $|p(x) - p(y)|_\infty > \lambda \|x - y\|_\infty$.

Daskalakis and Papadimitriou [DP11] have shown that REAL LOCALOPT is polynomially equivalent to LOCALOPT, i.e., REAL LOCALOPT is PLS-complete.

Theorem 5.2 (Daskalakis and Papadimitriou [DP11])**.** *The problem* REAL LOCALOPT *is a* PLS*-complete problem.*

5.1.2 Complexity Class CLS

Now, we explain the complexity class CLS. This class was introduced by Daskalakis and Papadimitriou [DP11]; they pointed out that many natural problems belong to both PPAD and PLS. For instance, computing a fixed point of a contraction map, called CONTRACTION MAP (see Definition 5.6), solving the linear complementarity problem for P-matrices, called P-LCP, finding a stationary point of a polynomial, and a computing a mixed Nash equilibrium on a network coordination game, which see also Chapter 7.

Daskalakis and Papadimitriou [DP11] defined the complexity class CLS for "*Continuous Local Search*" in order to understand the complexity of search problems belonging to both PPAD and PLS. As mentioned in Chapter 3, several typical complexity classes, such as PPAD and PLS, were defined by using Boolean circuits. On the other hand, CLS is defined by using arithmetic circuits. The formal definition of CLS is a formulation based on the problem CONTINUOUS LOCALOPT, which is a total search problem; and hence, the class CLS is a set of total search problems that are polynomial-time reducible to CONTINUOUS LOCALOPT.

Definition 5.3. CONTINUOUS LOCALOPT
Input:

- two parameters $\varepsilon, \lambda > 0$,
- two well-behaved arithmetic circuits computing $f : [0,1]^3 \to [0,1]^3$ and $p : [0,1]^3 \to [0,1]^2$

Task: Find one of the following:

- a point $x \in [0,1]^3$ such that $p(x) \leq p(f(x)) + \varepsilon$;
- two points $x, y \in [0,1]^3$ such that $\|f(x) - f(y)\|_\infty > \lambda \|x - y\|_\infty$.
- two points $x, y \in [0,1]^3$ such that $\|p(x) - p(y)\|_\infty > \lambda \|x - y\|_\infty$.

Definition 5.4. The complexity class CLS is the set of all search problems that are reducible to CONTINUOUS LOCALOPT in polynomial time.

Of course, we are able to generalize CONTINUOUS LOCALOPT to the same problems with other dimensions.

Recently, Fearnley, Goldberg, Hollender, and Savani [Fea+21] have shown that $\mathtt{CLS} = \mathtt{PPAD} \cap \mathtt{PLS}$. They proved that the problem of finding a KKT point is CLS-complete. Furthermore, Göös, Hollender, Jain, Mayster, Pires, Robere, and Tao [Göö+22] have proven that the class EOPL contains CLS. The definition based on CONTINUOUS LOCALOPT is an alternative formulation of the class EOPL.

5.2 Complexity of Computing a Fixed Point

The fixed point computation, which is a problem whose task is to find a fixed point guaranteed by some fixed point theorem, is a fascinating problem among computational problems based on arithmetic circuits. Basically, a fixed point computation is formulated as follows: We are given an arithmetic circuit that computes a transition function $f : [0,1]^3 \to [0,1]^3$ and an approimate parameter $\varepsilon > 0$ as an input, and seek a point $x \in [0,1]^3$ such that $\|f(x) - x\|_\infty \leq \varepsilon$. Normally, we attach some "violations" as solutions for guaranteeing the totality. For example, the set of solutions to the problem BROUWER (see Definition 5.5) contains a pair of two points that reveals the transition function is not Lipschitz continuous.

In this section, we introduce the research stream of the complexity of fixed-point computation.

5.2.1 Brouwer's Fixed Point Theorem

The problem of finding a Brouwer's fixed point is the most famous fixed point computation problem. This problem aims to compute a fixed point whose existence is guaranteed by Brouwer's fixed point theorem [Bro11]. The formal definition is as follows:

Definition 5.5. BROUWER
Input:

- two paramters $\varepsilon, \lambda > 0$;
- a well-behaved arithmetic circuit computing $f : [0,1]^3 \to [0,1]^3$.

Task: Find one of the following:

- a point $x \in [0,1]^3$ such that $\|f(x) - x\|_\infty \leq \varepsilon$;
- two points $x, y \in [0,1]^3$ such that $\|f(x) - f(y)\|_\infty > \lambda \|x - y\|_\infty$.

Note that we employ the Lipschitz continuous to undertake the continuity of a transition function. This is a condition to meet the totality of BROUWER. We can straightforwardly see that BROUWER is a total search problem from Brouwer's fixed point theorem and the second type of solution, which reports the violation of the Lipschitz continuity.

Papadimitriou [Pap94b] proved that BROUWER is PPAD-complete. It is easy to see that we have a polynomial-time algorithm solving BROUWER in the one-dimensional case. Unfortunately, this problem is PPAD-complete even if the dimension is two [CD09].

The complexity of BROUWER has been applied to a wide range of fields to prove the PPAD-completeness of various computational problems. For instance, the fact that it is PPAD-complete to compute a Nash equilibrium is based on the PPAD-completeness of BROUWER [CD09; Pap94b].

5.2.2 Banach's Fixed Point Theorem

Next, we will introduce the computational problem, which is related to Banach's fixed point theorem [Ban22], and its complexity. Informally speaking, the problem BANACH is a restricted case of BROUWER; we require that the Lipschitz constant λ holds that $0 < \lambda < 1$. The formal definition is as follows:

Definition 5.6. BANACH (also known as CONTRACTION MAP)
Input:

- two paramters $\varepsilon > 0$
- a parameter λ with $0 < \lambda < 1$;
- a well-behaved arithmetic circuit computing $f : [0,1]^3 \to [0,1]^3$.

Task: Find one of the following:

- a point $x \in [0,1]^3$ such that $\|f(x) - x\|_\infty \leq \varepsilon$;
- two points $x, y \in [0,1]^3$ such that $\|f(x) - f(y)\|_\infty > \lambda \|x - y\|_\infty$.

Daskalakis and Papadimitriou [DP11] have proven that the problem BANACH belongs to the complexity class PPAD$\cap$PLS. However, it is still open that BANACH is a complete problem for PPAD$\cap$PLS. More precisely, it is an unsolved problem whether BANACH is PPAD$\cap$PLS-hard for ℓ_p-norm [Fea+21]. The reason why we mention the ℓ_p-norm is that it has proven that the variant of BANACH that has an additional well-behaved arithmetic circuit representing a distance as an input and an extra solution noticing the violation of the metric notion is PPAD$\cap$PLS-complete [DTZ18; Fea+20].

5.2.3 Caristi's Fixed Point Theorem

It is mathematically known that Caristi's fixed point theorem is a generalization of Banach's fixed point theorem. The fixed-point computation related to this theorem is formulated as follows:

Definition 5.7. CARISTI
Input:

- three parameters $\varepsilon, \eta, \lambda > 0$;
- two well-behaved arithemetic circuits computing
  - $f : [0,1]^3 \to [0,1]^3$ and
  - $\varphi : [0,1]^3 \to [0,1]$.

Task: Find one of the following:

- a point $x \in [0,1]^3$ such that $\|f(x) - x\|_\infty \leq \varepsilon$;
- a point $x \in [0,1]^3$ such that $\eta \|x - f(x)\|_\infty > \varphi(x) - \varphi(f(x))$;
- two points $x, y \in [N]^3$ such that $|\varphi(x) - \varphi(y)| > \lambda \|x - y\|_\infty$.

The computational problem CARISTI takes a potential function $\varphi : [0,1]^3 \to [0,1]$ in addition to a transition function $f : [0,1]^3 \to [0,1]^3$. Note that it requires the potantial function is Lipschitz continuous while the transition fucntion does not require. It is easy to see that CARISTI is a total search problem from Caristi's fixed point theorem.

We prove that the problem CARISTI is PLS-complete in Chapter 6.

In the previous work, Chang and Lyuu [CL10] considered the query complexity of Banach's and Caristi's fixed point theorems. They have shown the query lower bounds for the problems BANACH and CARISTI. In the black-box oracle model, we can query a finite metric space (M, c) and a given function $f : M \to M$. Chang and Lyuu [CL10] have proven that every randomized algorithm for finding a Banach's fixed point makes an expected $\Omega(\sqrt{|M|})$ oracle queries. Furthermore, they proved that every randomized algorithm for finding a Caristi's fixed point makes an expected $\Omega(|M|)$ oracle queries.

5.2.4 Brøndsted's Fixed Point Theorem

Brøndsted's fixed point theorem is a variant of Caristi's fixed point theorem; we can also regard that theorem as a generalization of Banach's fixed point theorem. Remember that the statement of Caristi's fixed point theorem requires that a potential function is continuous, but a transition function is not necessarily continuous. On the other hand, Brøndsted's fixed point theorem requires that a transition function is continuous, but a potential function is not necessarily continuous.

The computational problem BRØNDESTED is formulated as follows:

Definition 5.8. BRØNDESTED
Input:

- three parameters $\varepsilon, \eta, \lambda > 0$;
- two well-behaved arithemetic circuits computing
  - $f : [0,1]^3 \to [0,1]^3$

– $\varphi : [0,1]^3 \to [0,1]$.

Task: Find one of the following:

- a point $x \in [0,1]^3$ such that $\|f(x) - x\|_\infty \le \varepsilon$;
- a point $x \in [0,1]^3$ such that $\eta \|x - f(x)\|_\infty > \varphi(x) - \varphi(f(x))$;
- two points $x, y \in [N]^3$ such that $|f(x) - f(y)| > \lambda \|x - y\|_\infty$.

As we will discuss in Chapter 6, It is trivial that BRØNDESTED belongs to the class PPAD. Furthermore, we can straightforwardly see that this computational problem is CLS-hard. However, whether BRØNDESTED is PPAD-complete or CLS-complete is still open.

5.2.5 Tarski's Fixed Point Theorem

Finally, we will introduce the previous works related to Tarski's fixed point theorem. Etessami, Papadimitriou, Rubinstein, and Yannakakis [Ete+20] formulated the fixed-point computation problem based on Tarski's fixed point theorem, and they considered the complexity of such a problem. Tarski's fixed point theorem [Tar55] is also an order-theoretic fixed point theorem. While we explained fixed-point computation problems defined by arithmetic circuits from the previous subsection, the computational problem TARSKI can be formulated by Boolean circuits.

Definition 5.9. TARSKI
Input:

- a Boolean circuit computing $f : [N]^d \to [NN]^d$.

Task: Find one of the following:

- a point $x \in [N]^d$ such that $f(x) = x$;
- two points $x, y \in [N]^d$ such that $x \preceq y$ and $f(x) \not\preceq f(y)$.

Here, $x \preceq y$ implies that for every $i \in [d]$, $x_i \le y_i$ holds; usually, this property is called by the component-wise order.

Etessami, Papadimitriou, Rubinstein, and Yannakakis [Ete+20] have shown that TARSKI is in PPAD$\cap$PLS. However, it is still open whether TARSKI is PPAD$\cap$PLS-hard. They also considered the query complexity of finding a Tarkis's fixed point.

In the black-box oracle model of TARSKI, we can query the function $f : [N]^d \to [N]^d$. Dang, Qi, and Ye [DQY11] established the $O(\log^d N)$ upper bound for this problem. Etessami, Papadimitriou, Rubinstein, and Yannakakis [Ete+20] have shown a lower bound for the query model of TARSKI on a two-dimension Euclidian grid space; we need $\Omega(\log^2 N)$ to find a Tarski's fixed point. Their results have given us the tight bound for the two-dimensional TARSKI problem. Recent studies [FS21; FPS20; CL22] have been improving the upper bound for the query complexity of TARSKI. Fearnley and Savani [FS21] showed an $O(\log^{d-1} N)$-queries algorithm. Fearnley, Pálvögyi, and Savani [FPS20] provided an $O(\log^{2\lceil d/3 \rceil} N)$-queries algorithm. The most recent result by Chen and Li [CL22] has shown an $O(\log^{\lceil (d+1)/2 \rceil} N)$-queries algorithm for TARSKI. Combining the lower bound for TARSKI by Etessami,

Papadimitriou, Rubinstein, and Yannakakis [Ete+20], we have the tight bound for TARSKI when the dimension $d \leq 3$. However, it is still unknown the tight bound for the TARSKI instance with four or more dimensions.

5.3 On the Complexity of Strong Approximation

Up to present, we have considered the notion of weak solution. An ε-approximate fixed point of a function $f : [0,1]^3 \to [0,1]^3$ is a point $x \in [0,1]^3$ such that $\|f(x) - x\|_\infty \leq \varepsilon$. Nevertheless, such a point may be far from actual exact fixed point.

In this section, we consider the complexity of computing an exact fixed point.

5.3.1 Complexity Class FIXP

The complexity class FIXP captures the complexity of computing an exact fixed point of a function mapping the unit cube into itself that is computed by an arithmetic circuit; not this arithmetic circuit is not necessarily well-behaved.

Consider the following problems: Given an algebraic circuit (straight-line program) over basis $\{+, \times, -, /, \max, \min\}$ with rational constants, having n input variables and n outputs, such that the circuit represents a continuous function $f : [0,1]^n \to [0,1]^n$, find a fixed point of f or a strong ε-approximate fixed point of f.

Etessami and Yannakakis [EY10] have shown that the problem of computing an exact or strong approximate mixed Nash equilibrium of a three-player strategic-form game is FIXP-complete. Thus, computing a strong approximate Nash equilibrium for a three-player game is as hard as computing a strong approximate Brouwer's fixed point given by an arithmetic circuit. Remark that Chen, Deng, and Teng [CDT09] have proven that every two-player strategic-form game always has a rational Nash equilibrium, which implies that the problem of computing one is PPAD-complete.

Goldberg and Hollender [GH19] have shown that the Hairy Ball problem is FIXP-hard. However, it is still open whether this problem is FIXP-complete. They mentioned that we should try to reduce Hairy Ball to Borsuk-Ulam as a first step to obtain the completeness result, even though no such mathematical seems to be known.

5.3.2 Complexity Class BU

The complexity class BU was introduced by Deligkas, Fearnley, Melissourgos, and Spirakis [Del+21]. This class is formulated as the set of search problems that are the problem of computing an exact or a strong approximation solution to the Borsuk-Ulam problem. By definition, the complexity class BU captures an exact variant of PPA-complete problems, and thus, BU contains the class FIXP.

Deligkas, Fearnley, Melissourgos, and Spirakis [Del+21] have proven that the problem of computing an exact solution to the CONSENSUS HALVING problem is FIXP-hard and the LinearBU has the same complexity with PPA.

Chapter 6

On the Complexity of Caristi's Fixed Points

As we introduced in the previous chapter, fixed-point computation problems comprise a large and important class of total search problems, i.e., some fixed point theorems make up exotic subclasses of TFNP. It is well-known that some problems of computing a fixed point capture the complexity features of a few fascinating subclasses of TFNP, such as PPAD and CLS.

This chapter focuses on Caristi's fixed point theorem and proves that this fixed point theorem characterizes the class PLS, which is a class of search problems that can be solved by a local search method. Specifically, we show that the problem of finding a Caristi's fixed point is PLS-complete. Caristi's fixed point theorem is an order-theoretic fixed point theorem, and this theorem is often said to be a generalization of Banach's fixed point theorem [GD03]. In fact, we can find Banach's fixed point theorem by using Caristi's fixed point theorem.

Furthermore, we consider a variant of computing a Caristi's fixed point. The existence of a solution to this problem is guaranteed by Brøndsted's fixed point theorem. We provide, in this chapter, an upper bound and a lower bound for the problem of finding a Brøndsted's fixed point. More specifically, we show that this problem is CLS-hard and belongs to PPAD.

Here, we discuss the complexity of computing a Caristi's point and computing a Brøndsted's fixed point. Let (M,d) be a complete metric space and $\eta > 0$ be some positive value. To formulate the basic problem, we consider a potential function $\varphi : M \to \mathbb{R}_{\geq 0}$ and a function $f : M \to M$ satisfying that $\eta d(x, f(x)) \leq \varphi(x) - \varphi(f(x))$ for each $x \in M$; when M is discrete, we assume that the functions are presented by Boolean circuits; when M is continuous, we assume that the functions are presented by arithmetic circuits.

6.1 Computing a Caristi's Fixed Point

Let (M,d) be a complete metric space. We assume that $\varphi : M \to \mathbb{R}_{\geq 0}$ is lower semi-continuous and η is some positive real number. We say that a function $f : M \to M$ satisfies Caristi's condition if it holds that $\eta d(x, f(x)) \leq \varphi(x) - \varphi(f(x))$ for every point $x \in M$. Caristi's fixed point theorem states that every function $f : M \to M$ satisfying Caristi's condition always has a fixed point [Car76]. In this section, we consider the complexity of computing a fixed point of f. We shall formally define

this as a total search problem, using a standard construction to avoid the "*promise*" that ϕ is lower semicontinuous and f satisfies Caristi's condition.

Firstly, we fix the metric notion used throughout this chapter. When we consider the discrete domain, the complete metric space (G_n^k, d_1) is defined as for each pair of x and y in G_n^k,

$$d_1(x,y) = \sum_{i\in[k]} |x_i - y_i|,$$

where n and k are some positive integers. On the other hand, when we consider the continuous domain, the complete metric space $([0,1]^3, d_\infty)$ is defined as for each pair of x and y in $[0,1]^3$,

$$d_\infty(x,y) = \max\{|x_1 - y_1|, |x_2 - y_2|, |x_3 - y_3|, \}.$$

In this chapter, we discuss separately when the domain M is discrete and continuous. In the discrete domain, we consider the finite discrete Euclidian metric space (G_n^k, d_1). Note that, in this case, any function $\varphi : G_n^k \to \mathbb{Z}_{\geq 0}$ is always lower semicontinuous. Furthermore, (G_n^k, d_1) is a complete metric space. Section 6.1.1 shows that the problem of comuputing a Caristi's fixed point on (G_n^k, d_1) is PLS-complete. On the other hand, in the continuous domain, we consider the unit cube $([0,1]^3, d_\infty)$. In order to avoid semicontinuous, we require that φ is λ-Lipschitz continuous. Section 6.1.2 shows that the problem of finding a Caristi's fixed point on $([0,1]^3, d_\infty)$ is also PLS-complete.

6.1.1 Discrete Domain

In this subsection, we consider the complexity of computing a Caristi's fixed point on a finite discrete Euclidian grid space (G_n^k, d_1). Thus, we consider the complexity the problem such as: Given a positive integer η and two functions $\varphi : G_n^k \to \mathbb{Z}_{\geq 0}$ and $f : G_n^k \to G_n^k$ satisfying that $\eta d(x, f(x)) \leq \varphi(x) - \varphi(f(x))$ for each $x \in G_n^k$, find a fixed point of f. Notice that it is unlikely that there will be a polynomial-time algorithm verifying whether the given functions satisfy Caristi's condition. Now, we formally define the problem of finding a Caristi' fixed point as a discrete total search problem by adding a witness showing that f violates Caristi's condition.

We are interested in the complexity of the following total search problem. In the rest of this section, we prove that this problem is PLS-complete.

Definition 6.1. DISCRETE CARISTI
Input:

- a positive integer η and
- two Boolean circuits computing $f : G_n^k \to G_n^k$ and $\varphi : G_n^k \to G_m$.

Task: Find on of the following:

- a point $x \in [N]^d$ such that $f(x) = x$;
- a point $x \in G_k^k$ such that $\eta d_1(x, f(x)) > \varphi(x) - \varphi(f(x))$.

Note that DISCRETE CARISTI is a total search problem: If for every point $x \in G_n^k$, the given function f satisfies Caristi's condition, then there exists a fixed point, and otherwise, we obtain a point which violates Caristi's condition.

Theorem 6.2. DISCRETE CARISTI *is* PLS-*complete.*

Proof. First, we show that DISCRETE CARISTI is in PLS. To prove this, we construct a polynomial-time reduction from DISCRETE CARISTI to LOCALOPT. Our construction is very easy. When we are given an instance of (f, φ, η) of DISCRETE CARISTI, we define $p(x) = \varphi(x)$ for each $x \in G_n^k$ and construct the instance (f, p) of LOCALOPT.

Assume that we obtain a point $x \in G_n^k$ satisfying that $p(x) \leq p(f(x))$. If $f(x) = x$, then it is a fixed point of f. On the other hand, if $f(x) \neq x$, then it holds that $p(x) - p(f(x)) \leq 0 < 1 \leq \eta d_1(x, f(x))$. Therefore, x is a solution to the original instance.

Hence, DISCRETE CARISTI belongs to PLS.

Next, we prove that DISCRETE CARISTI is PLS-hard. To prove this, we construct a polynomial-time reduction from LOCALOPT to DISCRETE CARISTI. Given an instance (f, p) of LOCALOPT, we define $\pi = k \cdot 2^n$ and the function $\varphi(x) = \pi p(x)$ for each $x \in G_n^k$. Moreover, we define $\eta := 1$. It is easy to see that every fixed point of f is a solution to LOCALOPT. What remains is to prove that a point $x \in G_n^k$ satisfying that $d_1(x, f(x)) > \varphi(x) - \varphi(f(x))$ is a solution to LOCALOPT. Suppose that we obtain such a point x. Then it satisfies that $d_1(x, f(x)) > \varphi(x) - \varphi(f(x)) = \pi(p(x) - p(f(x)))$. Notice that $d_1(x, f(x)) = \sum_{i \in [k]} |x_i - y_i| \leq \sum_{i \in [k]} (2^n - 1) < k \cdot 2^n = \pi$, and hence, it hollows that

$$p(x) - p(f(x)) < \frac{1}{\pi} d_1(x, f(x)) < \frac{1}{\pi} \cdot \pi = 1.$$

Rcall that p returns a non-negative integer for each point. This implies that $p(x) \leq p(f(x))$. Thus, we obtain a solution to LOCALOPT. Therefore, DISCRETE CARISTI is PLS-hard. □

6.1.2 Continuous Domain

In this section, We consider the complexity of computing a Caristi's fxied point on a unit cubit $([0,1]^3, d_\infty)$. Thus, we consider the problem such as: Given two positive numbers η and ε and two functions $\varphi : [0,1]^3 \to [0,1]$ which is λ-Lipschitz continuous and $f : [0,1]^3 \to [0,1]^3$ satisfying Caristi's condition, find an ε-approximate fixed point of f, i.e., a point $x \in [0,1]^3$ such that $d_\infty(f(x), x) \leq \varepsilon$. However, as in the discrete domain, there seem to be no polynomial-time algorithms that decide whether, for every pair of points, φ is λ-Lipschitz continuous and satisfies Caristi's condition. Therefore, we formally define the problem of finding an approximate Caristi's fixed point as a continuous total search problem by adding a witness showing that f violates Caristi's condition or φ is not λ-Lipschitz continuous.

Now, we are interested in the complexity of the following total search problem. In the rest of this subsection, we prove that this problem is also PLS-complete.

Definition 6.3. CONTINUOUS CARISTI
Input:

- three positive parameters ε, η, and λ
- two well-behaved arithmetic circuits computing $f : [0,1]^3 \to [0,1]^3$ and $\varphi : [0,1]^3 \to [0,1]$.

Task: Find on of the following:

- a point $x \in [0,1]^3$ such that $d_\infty(x, f(x)) \leq \varepsilon$;
- a point $x \in [0,1]^3$ such that $\eta d_\infty(x, f(x)) > \varphi(x) - \varphi(f(x))$;
- two points $x, y \in [0,1]^3$ such that $|\varphi(x) - \varphi(y)| > \lambda d_\infty(x, y)$.

Note that CONTINUOUS CARISTI is a total search problem: If the given function φ is λ-Lipschitz continuous and f satisfies Caristi's condition, then there exists at least one fixed point by Caristi's fixed point theorem, and otherwise, we obtain a witness which shows that φ is not λ-Lipschitz continuous or f does not satisfy Caristi's condition.

Theorem 6.4. CONTINUOUS CARISTI *is* PLS-*complete.*

Proof. First, we prove that CONTINUOUS CARISTI is PLS-hard. Thus, we construct a polynomial-time reduction from REAL LOCALOPT to CONTINUOUS CARISTI. We assume that we are given an instance $(f, p, \varepsilon, \lambda)$ of REAL LOCALOPT. Then, we define $\varphi(x) := p(x)$ for every $x \in [0,1]^3$, $\varepsilon' := \varepsilon/(1+\lambda)$, and $\eta := \varepsilon$. Thus, we consider the instance $(f, \varphi, \varepsilon', \eta, \lambda)$ of CONTINUOUS CARISTI. It is easy to see that a two points x, y satisfying that $|\varphi(x) - \varphi(y)| > \lambda d_\infty(x, y)$ are a solution of the original instance $(f, p, \varepsilon, \lambda)$. Henceforth, we suppose that each obtained solution $x \in [0,1]^3$ of $(f, \varphi, \varepsilon', \eta, \lambda)$ satisfies that $|\varphi(x) - \varphi(f(x))| \leq \lambda d_\infty(x, y)$. Next, we assume that we obtain a point $x \in [0,1]^3$ satisfying that $d_\infty(x, f(x)) \leq \varepsilon'$. By definition, it follows that

$$\begin{aligned}
|p(x) - p(f(x))| &= |\varphi(x) - \varphi(f(x))| \\
&\leq \lambda d_\infty(x, f(x)) \\
&\leq \lambda \cdot \varepsilon' \\
&\leq \lambda \cdot \frac{\varepsilon}{1+\lambda} \\
&< \varepsilon.
\end{aligned}$$

This implies that x is a solution to REAL LOCALOPT since $p(x) \leq p(f(x)) + \varepsilon$. Finally, we suppose that we obtain a point $x \in [0,1]^3$ satisfying that $\eta d_\infty(x, f(x)) > \varphi(x) - \varphi(f(x))$. By definition, it is not difficult to see that

$$\begin{aligned}
\varepsilon = \eta \cdot 1 &\geq \eta d_\infty(x, f(x)) \\
&> \varphi(x) - \varphi(f(x)) \\
&= p(x) - p(f(x)).
\end{aligned}$$

This implies that x is a solution to REAL LOCALOPT. Therefore, CONTINUOUS CARISTI is PLS-hard.

Next, we prove that CONTINUOUS CARISTI belongs to PLS. To prove this, we construct a polynomial-time reduction from CONTINUOUS CARISTI to REAL LOCALOPT. We assume that we are given an instance $(f, \varphi, \varepsilon, \eta, \lambda)$ of CONTINUOUS CARISTI. Then we construct the tuple $(f, \varepsilon, \eta \cdot \varepsilon, \lambda)$; this is the instance of REAL LOCALOPT. It is easy to see that this instance can be constructed in polynomial time. What remains is to show that we can convert from every solution of $(f, \varepsilon, \eta \cdot \varepsilon, \lambda)$ to a solution of $(f, \varepsilon, \varepsilon, \eta, \lambda)$ in polynomial time. It is not difficult to see that every pair of points which violates λ-Lipschitz continuously of φ is an original solution. Now, we assume that we obtain a point x such that $\varphi(x) \leq \varphi(f(x)) + \eta \cdot \varepsilon$. If x satisfies that $d_\infty(x, f(x)) \leq \varepsilon$, then x is an ε-approximate fixed point of f, and otherwise, it follows that

$$\eta d_\infty(x, f(x)) > \eta \cdot \varepsilon \geq \varphi(x) - \varphi(f(x))$$

This implies that x is a solution to CONTINUOUS CARISTI.

Therefore, CONTINUOUS CARISTI belongs to PLS. □

6.2 Computing a Brøndested's Fixed Point

In this section, we consider the complexity of a continuous total search problem that the existence of a solution is guaranteed by Brøndsted's fixed point theorem. This is a variant of CONTINUOUS CARISTI.

Let (M, d) be a complete metric space. We assume that $\varphi : M \to \mathbb{R}_{\geq 0}$ is any (not necessarily lower semicontinuous function and η is some positive number. Brøndsted's fixed point theorem states that every continuous function $f : M \to M$ satisfying that $\eta d(x, f(x)) \leq \varphi(x) - \varphi(f(x))$ for each $x \in M$, that is, f satisfies Caristi's condition, has at least one fixed point [Brø74]. From now on, we consider the complexity of the problem of finding a Brøndsted's fixed point. As same as the problem of finding a Caristi's fixed point, we shall formally define this problem as a total search problem, using a standard construction to avoid the "*promise*" that f is continuous and satisfies Caristi's condition.

Notice that this problem on a finite discrete Euclidian metric space is the same problem as DISCRETE CARISTI because a function $g : G_n^k \to G_n^k$ is continuous on (G_n^k, d_1). Hence, we consider the problem on the continuous unit cube $([0,1]^3, d_\infty)$. Thus, we are interested in the complexity of the following total search problem.

Definition 6.5. BRØNDESTED
Input:

- three positive rational numbers ε, η, and λ
- two well-behaved arithmetic circuits computing $f : [0,1]^3 \to [0,1]^3$ and $\varphi : [0,1]^3 \to [0,1]$.

Task: Find on of the following:

- a point $x \in [0,1]^3$ such that $d_\infty(x, f(x)) \leq \varepsilon$,
- a point $x \in [0,1]^3$ such that $\eta d_\infty(x, f(x)) > \varphi(x) - \varphi(f(x))$,

- two points $x, y \in [0,1]^3$ such that $d_\infty(f(x), f(y)) > \lambda d_\infty(x, y)$.

Note that BRØNDESTED on $([0,1]^3, d_\infty)$ is a total search problem: If the given function f is λ-Lipschitz continuous and satisfies Caristi's condition, then there exists at least one fixed point, and otherwise, we obtain two points which exhibit that f is not λ-Lipschitz continuous or a point which exhibit that f does not satisfy Caristi's condition.

In the rest of this section, we provide the upper bound and lower bound for BRØNDESTED. Section 6.2.1 shows that this problem belongs to PPAD. In Section 6.2.2, we prove that BRØNDESTED is CLS-hard, that is, every search problem belonging to CLS is polynomial-time reducible to BRØNDESTED.

6.2.1 Comupting a Brøndested's Fixed Point is in PPAD

Recall that the complexity class PPAD is defined as the set of all search problems that are polynomial-time reducible to the problem of finding a Brouwer's fixed point: Given a λ-Lipschitz continuous function $f : [0,1]^3 \to [0,1]^3$ and a positive number $\varepsilon > 0$, find an ε-approximate fixed point of f. To avoid the "*promise*" that the given function f is λ-Lipschitz continuous, the problem BRØNDESTED is defined as follows.

For each instance $(f, \varphi, \varepsilon, \eta, \lambda)$ of BRØNDESTED, the tuple $(f, \varepsilon, \lambda)$ is an instance of BROUWER from the definition. Then it is very easy to see that every obtained solution to BROUWER is a solution to BRØNDESTED. This implies that there exists a polynomial-time reduction from BRØNDESTED to BROUWER. Therefore, the following statement immediately follows.

Theorem 6.6. BRØNDESTED *belongs to* PPAD.

6.2.2 Computing a Brøndested's Fixed Point is CLS-hard

The complexity class CLS, introduced by Daskalakis and Papadimitriou [DP11], is the first important subclass of PPAD $\cap$ PLS. Recall that CLS is defined by the following search problem called CONTINUOUS LOCALOPT (see Definition 5.3).

In the rest of this subsection, we show the CLS-hardness of BRØNDESTED. Thus, we construct a polynomial-time reduction from CONTINUOUS LOCALOPT to BRØNDESTED. The construction of our reduction is basically the same as the reduction from REAL LOCALOPT to CONTINUOUS CARISTI shown in Theorem 6.4.

Theorem 6.7. BRØNDESTED *is* CLS*-hard.*

Proof. Now, we assume that we are given an instance $\mathscr{I} = (f, p, \varepsilon, \lambda)$ of CONTINUOUS LOCALOPT. Then we define the instance $\mathscr{J} = (f', \varphi, \varepsilon', \eta, \lambda)$ of BRØNDESTED as $\varphi(x) := p(x)$ for every $x \in [0,1]^3$, $\varepsilon' := \varepsilon/(1+\lambda)$, and $\eta := \varepsilon$. It is easy to see that this instance can be constructed in polynomial time.

From now on, we prove that we can convert a solution of $\mathscr{I}$ from an obtained solution of $\mathscr{J}$. By definition, it is not difficult to see that two points $x, y \in [0,1]^3$ such that $d_\infty(f(x), f(y)) > \lambda d_\infty(x, y)$ are immediately a solution of $\mathscr{I}$. In the rest of this proof, we suppose that every obtained point $x \in [0,1]^3$ satisfies that $d_\infty(f(x), f(f(x)))$ $\leq \lambda d_\infty(x, f(x))$.

We assume that we obtain a point $x \in [0,1]^3$ such that $d_\infty(x, f(x)) \leq \varepsilon'$. If $|p(x) - p(f(x))| > \lambda d_\infty(x, f(x))$, then two points x and $f(x)$ imply that p is not λ-Lipschitz continuous, that is, we obtain a solution to CONTINUOUS LOCALOPT, and otherwise, it follows that

$$\begin{aligned} |p(x) - p(f(x))| &\leq \lambda d_\infty(x, f(x)) \\ &\leq \lambda \cdot \varepsilon' \\ &= \lambda \cdot \frac{\varepsilon}{1+\lambda} \\ &\leq \varepsilon. \end{aligned}$$

This implies that $p(x) \leq p(f(x)) + \varepsilon$. Hence, x is a solution of $\mathscr{I}$. Finally, we asume that we obtain a point $x \in [0,1]^3$ such that $\eta d_\infty(x, f(x)) > \varphi(x) - \varphi(f(x))$. It holds that

$$\begin{aligned} \varepsilon = \eta \cdot 1 &\geq \eta \cdot d_\infty(x, f(x)) \\ &> \varphi(x) - \varphi(f(x)) = p(x) - p(f(x)). \end{aligned}$$

Thus, x is a solution to CONTINUOUS LOCALOPT.

Therefore, BRØNDESTED on $([0,1]^3, d_\infty)$ is CLS-hard. □

6.3 Conclusions

We have studied the complexity of finding a Caristi's fixed point on a discrete domain and a continuous domain, and we have proved that this problem is a PLS-complete in both domains. We have also provided the upper bound and the lower bound for the problem of finding a Bøndsted's fixed point. Specifically, we have proved that this problem is in PPAD and is a CLS-hard problem. We illustrate, in Figure 6.1, the relationship of the complexity around the problems of finding a fixed point obtained in our and the previous results.

There are open problems worth considering left by our work. It would be fascinating to clarify whether BRØNDESTED is a complete problem for PPAD or CLS. In particular, to show the PPAD-completeness of BRØNDESTED, we need to establish a method proving Brouwer's fixed point theorem on $([0,1]^3, d_\infty)$ by using Brøndsted's fixed point theorem. To show the CLS-completeness of BRØNDESTED, it is sufficient to prove that this problem is in PLS since CLS = PPAD $\cap$ PLS [Fea+21], and BRØNDESTED is CLS-hard by Theorem 6.7. Brøndsted's fixed point theorem is an order-theoretic fixed point theorem, and hence, we can find a Brøndsted's fixed point by using a local search method. It seems that BRØNDESTED also belongs to PLS. Recall that a potential function, p, given by an instance of REAL LOCALOPT, is required λ-Lipschitz continuous for some positive number λ. Furthermore, Daskalakis and Papadimitriou [DP11] have proven that REAL LOCALOPT is PLS-complete by relying on λ-Lipschitzness of the given potential function. On the other hand, a potential function, φ, given by an instance of BRØNDESTED, is not necessarily continuous. Therefore, we need more ideas to prove that BRØNDESTED belongs to PLS.

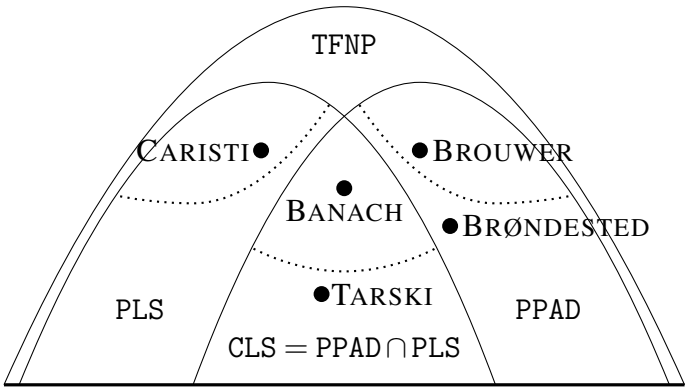

FIGURE 6.1: Overview of the complexity of problems of finding a fixed point belonging to TFNP from our and previous studies [Pap94b; CD09; DTZ18; Ete+20; Fea+21]. Problems which are known to be complete for some class are drawn above a dotted line.

Another interesting open question is the complexity of CONTINUOUS CARISTI with $\eta = 1$: Is CONTINUOUS CARISTI PLS-complete even if $\eta = 1$? The original statement of Caristi's fixed point theorem does not use the positive number η [Car76]. The claims are mathematically equivalent even if $\eta > 0$ is added. Naturally, we can define the problem of finding a Caristi's fixed point without η, but our to prove the PLS-hardness of CONTINUOUS CARISTI in the proof of Theorem 6.4. In contrast, we can straightforwardly see that DISCRETE CARISTI with $\eta = 1$ is PLS-complete.

Part IV

Algorithmic Game Theory

Chapter 7

Nash Equilibrium Computation

So far, we have reviewed the complexity of total search problems from mathematical perspectives. Now, we discuss such topics from the outlook on applied realms. This thesis focuses on the computational aspects of Nash equilibria, which are one of the most fascinated mathematical notions for the author.

The research interaction between Game Theory and Theoretical Computer Science has helped to study the computational issues underlying fundamental game-theoretic notions deeply. A seminal topic of these research streams is to clarify the complexity of computing Nash equilibria in multi-player non-cooperative games. A Nash equilibrium is an intuitive and essential concept of rationality in which no player could unilaterally deviate from her selected strategy to improve her reward.

7.1 Essence of Game Theory

We first define basic notations. A finite game $\mathscr{G} = ([N], \{S_i\}, \{p_i\})$ consists of a finite set of players $[N]$, a finite set of pure strategies S_i for each player $i \in [N]$, and a payoff function $p_i : \times_{i\in[N]} S_i \to \mathbb{R}_{\geq 0}$ for each player $i \in [N]$. In the following, we denote S as $\times_{i\in[N]} S_i$ and S_{-i} as $\times_{j\in[N]\setminus\{i\}} S_j$ for simplicity.

A list of pure strategies $(s_1, s_2, \ldots, s_N) \in S$ is called a pure strategy profile. When each player $i \in [N]$ plays a pure strategy $s_i \in S_i$, we interpret it as a pure strategy profile $s = (s_1, s_2, \ldots, s_N)$ are selected and each player $i \in [N]$ obtains a payoff $p_i(s)$.

Consider the setting when every player tries to maximize their payoff. A pure strategy profile $s = (s_1, s_2, \ldots, s_N) \in S$ is a pure Nash equilibrium if for each player $i \in [N]$ and each strategy $t_i \in S_i$, it satisfies that $p_i(s_i, s_{-i}) \geq p_i(t_i, s_{-i})$. When players select a pure strategy profile $s = (s_1, s_2, \ldots, s_N) \in S$, the best response for a player $i \in [N]$ is $\arg\max_{t_i\in S_i} p(t_i, s_{-i})$. Here, we denote by s_{-i} by the list of pure strategies of all players except i.

Next, we describe mixed strategies. A mixed strategy is a probability distribution on a set of pure strategies. Thus, the set of mixed strategies Δ_i is defined as $\{x_i \in \mathbb{R}^S_{\geq 0} \; ; \; \sum_{s_i\in S_i} x_i(s_i) = 1\}$ for a player $i \in [N]$. When a player $i \in [N]$ plays a mixed strategy $x_i \in \Delta_i$, we interpret it as the player i plays a strategy $s_i \in S_i$ with probability $x_i(s_i)$. We call a list of mixed strategies $x = (x_1, x_2, \ldots, x_N) \in \Delta$ a mixed strategy profile, where $\Delta := \times_{i\in[N]} \Delta_i$. For a mixed strategy profile $x = (x_1, x_2, \ldots, x_N) \in \Delta$, an expected payoff for a player $i \in [N]$ is defined as $P(x) = \sum_{s\in S} \prod_{j\in[N]} p_i(s) x_j(s_j)$. A mixed strategy profile $x = (x_1, x_2, \ldots, x_N) \in \Delta$ is a mixed Nash equilibrium if for each player $i \in [N]$ and every mixed strategy $\xi_i \in \Delta_i$, it holds that $P(x_i, x_{-i}) \geq P(\xi_i, x_{-i})$. Here, we denote by x_{-i} by the list of mixed strategies of all players except i.

A pure/mixed Nash equilibrium is an intuitive notion of rationality or stability. Every player has no incentive to change their current pure/mixed strategy in a pure/mixed Nash equilibrium. Nash [NJ50] have proven that every finite game always has a mixed Nash equilibrium. His statement implies that any finite game can reach a quiescent situation. Note that Nash's theorem does not guarantee the existence of a pure Nash equilibrium. Actually, many finite games have no pure Nash equilibria; the Rock-paper-scissors game is an example.

The following natural questions worth considering arise: (1) Can such a situation reach practically? (2) Can we efficiently compute a mixed Nash equilibrium? (3) Can we efficiently decide whether a finite game has a pure Nash equilibrium? These questions have been widely explored, and there are many negative and some positive results. The next section will survey the research stream of computing pure/mixed Nash equilibria.

7.2 On the Complexity of Equilibrium Computation

The results of the last two decades are unraveling the complexity of computing mixed Nash equilibria [CD06; DGP09]. It is the most important result that the intractability for computing Nash equilibria in two-player games even if both players gain only a unit payoff at most [AKV05]. Furthermore, it is well known that the best algorithm has an exponential worst-case running time even in a two-player setting [SS04].

Motived by the above negative facts, some studies have concentrated on the complexity of computing specified classes of equilibria, such as pure, mixed, or correlated equilibria [FPT04; Pap05]. This thesis focuses on the uniform Nash equilibria, i.e., Nash equilibria consisting of strategies in which strategies are played according to a uniform distribution. A uniform Nash equilibrium is a kind of mixed equilibrium. However, unlike mixed Nash equilibria, uniform Nash equilibria do not always exist. Bonifaci, Iorio, and Laura [BIL08] showed that it is NP-complete to decide whether a given two-player game has a uniform Nash equilibria even if every element of payoff matrices is either 0 or 1. On the other hand, Addario-Berry, Kane, and Variant [AKV05] proved that there is a polynomial-time algorithm to find a uniform Nash equilibrium in such a two-player game if the graph defined by the game is planar.

7.2.1 Succinct Representation of Games

Any computational problem has inputs; the problem of computing a Nash equilibria takes a description of a finite game $\mathscr{G}$ for which we want to find a pure/mixed Nash equilibrium as an input. We need to concern about how long is such a description.

Consider a finite game $\mathscr{G} = ([N], \{S_i\}, \{p_i\})$. Here, we denote by m and k the maximum and minimum numbers of pure strategies, respectively: $m = \max_{i \in [N]} |S_i|$ and $k = \min_{i \in [N]} |S_i|$. Recall that a payoff function p_i depends on all players' selected pure strategies for each player $i \in [N]$. That is, we need at least k^N entries to describe a payoff function p_i. Thus, we need $N \cdot k^N$ elements to explain the game $\mathscr{G}$. It is obviously a huge size input.

Of course, the study of computational complexity aspects of seeking a pure/mixed Nash equilibrium on a finite strategic-form game is interested in multi-player settings. Therefore, we reflect on the way to succinctly represent a game. As mentioned

in the previous paragraph, it suffices to equip the description of a payoff function for every player. There are three natural ways to explain them briefly:

1. Gives a Boolean or arithmetic circuit computing a payoff function;
2. regards the number of players as a constant; and
3. for each player, restrict the number of players who affect its payoff function.

Certainly, the first approach is a concise implementation of a payoff function. However, it is inappropriate as our motive. It is unknown whether we have an efficient algorithm for deciding whether a given pure/mixed strategy profile is a pure/mixed Nash equilibrium. We can easily see that such a problem is NP-hard. Thus, the Nash equilibrium computation problem described in the first natural way may be outside of FNP.

Our research purpose is to sharpen the tractability-intractability boundary of the problem of computing pure/mixed Nash equilibria. Especially, we are interested in the problem belonging to between P and NP. Hence, this thesis only focuses on the remaining approaches. The last two methods have been widely studied.

Bimatrix Games

The second technique is simple. A fascinating situation is the two-player setting; we call such a game a bimatrix game. A bimatrix game is a two-player strategic-form game where players have a finite set of pure strategies. We call the first (resp. second) player the *row* (resp. *column*) player. Here, we denote by R and C by the sets of pure strategies for the row and column players, respectively. A bimatrix game is specified by non-negative real matrices $M_R, M_C \in \mathbb{R}_{\geq 0}^{R \times C}$; the rows and columns of both matrices are indexed by the pure strategies of the players.

A bimatrix game lies on the tractability-intractability boundary because the problem of computing a mixed Nash equilibrium on a bimatrix game is generally PPAD-complete [CDT09]. However, it is efficiently solvable if a bimatrix game is zero-sum [Cai+16]. Note that a three-player zero-sum game is PPAD-complete; we can easily see this fact from the PPAD-completeness of a bimatrix game.

Surprisingly, it is known that the problem of finding a mixed Nash equilibrium on a bimatrix game is PPAD-complete even if both players obtain at most a unit reward. Abbott, Kane, Valiant [AKV05] have provided an efficient procedure to transform a two-player game into a win-lose bimatrix game, where a win-lose bimatrix game is a two-player game in which every component of payoff matrices is either zero or one. Combining their result and [CDT09], we can see that it is PPAD-complete to compute a mixed Nash equilibrium on a win-lose bimatrix game.

From the above negative results, some studies related to the computational aspects of bimatrix games have focused on the specified classes of equilibria, such as pure, mixed, correlated etc. [FPT04; Pap05]. Bonifaci, Iorio, and Laura [BIL08] focused on uniform Nash equilibria. A Nash equilibrium (x_R, x_C) is said to be uniform if for each $r_i \in \mathrm{supp}(x_R)$ and each $c_j \in \mathrm{supp}(x_C)$, $x_R(r_i) = 1/|\mathrm{supp}(x_R)|$ and $x_C(c_j) = 1/|\mathrm{supp}(x_C)|$, where $\mathrm{supp}(x) = \{i \in [n] \ ; \ x(i) > 0\}$ for a vector $x \in \mathbb{R}_{\geq 0}^n$. Although a uniform Nash equilibrium is a mixed Nash equilibrium, the existence of uniform Nash equilibria does not guarantee. Bonifaci, Iorio, and Laura [BIL08] and

Codenotti and Štefancič [CS05] showed the NP-completeness of checking whether a given bimatrix game has a uniform Nash equilibrium.

Their proofs were based on a graph-theoretical approach. There is a one-to-one correspondence between a win-lose bimatrix game and a bipartite digraph. Let (M_R, M_C) be a win-lose bimatrix game. Addario-Berry, Oliver, and Vetta [AOV07] and Bonifaci, Iorio, and Laura [BIL08] have constructed a bipartite digraph $G = (V, E)$ from (M_R, M_C) as follows: The set of vertices V is equal to $R \cup C$; and we have an arc $(r_i, c_j) \in E$ if $M_R(r_i, c_j) = 1$ and an arc $(c_j, r_i) \in E$ if $M_C(r_i, c_j) = 1$. We can effortlessly see that a similar technique can also be done in the opposite direction, i.e., construction from a bipartite digraph to a bimatrix game.

Fix arbitrary non-empty subset $S \subseteq V$, let $G[S]$ be the subgraph induced by S. We denote by $\Gamma^+(v)$ the set of out-neighbors of vertex $x \in V$, and by Γ^+_S the set of out-neighbors of a vertex $v \in V$ that are in $S \subseteq V$.

Let $G = (R \cup C, E)$ be a bipartite digraph, where $R \cap S = \emptyset$. For a non-empty subset $S \subseteq R \cup C$ with $S \cap R \neq \emptyset \neq S \cap C$, the induced subgraph $G[S]$ is (α, β) out-regular if the following conditions holds: (i) For every $r_i \in S \cap R$, $|\Gamma^+_S(r_i)| = \alpha$; and (ii) for every $c_j \in S \cap C$, $|\Gamma^+_S(c_j)| = \beta$. We say that G has an out-regular subgraph if there exists a triple (α, β, S) such that $G[S]$ is (α, β) out-regular, and $\alpha, \beta \geq 1$.

A vertex $v \in (R \cup C) \setminus S$ dominates an an (α, β) out-regular subgraph $G[S]$ is it satisfies that either $|\Gamma^+_S(v)| > \alpha$ if $v \in R$; or $|\Gamma^+_S(v)| > \beta$ if $v \in C$. We say that an (α, β) out-regular subgraph $G[S]$ is undominated if there are no vertices that dominate $G[S]$. We say that G has an undominated out-regular subgraph if there is a triple (α, β, S) such that $G[S]$ is an undominated (α, β) out-regular subgraph.

Bonifaci, Iorio, and Laura [BIL08] and Addario-Berry, Oliver, and Vetta [AOV07] have proven the following important characterization:

Theorem 7.1 (Bonifaci, Iorio, and Laura [BIL08] and Addario-Berry, Oliver, and Vetta [AOV07])**.** *A win-lose bimatrix game has a uniform Nash equilibrium if and only if the corresponding bipartite digraph has an undominated out-regular subgraph.*

Bonifaci, Iorio, and Laura [BIL08] have proven the NP-hardness of verifying the existence of uniform Nash equilibria on a given win-lose bimatrix game by using Theorem 7.1; they reduced 3-SAT to the problem of deciding whether a bipartite digraph has an undominated out-regular subgraph.

Addario-Berry, Oliver, and Vetta [AOV07] focused on the special case of win-lose bimatrix games which induce a planar digraph. They have shown that such a win-lose bimatrix game always has a uniform Nash equilibrium, and we can find it in polynomial time. Their proof also relied on Theorem 7.1; they proved that a planar bipartite digraph has an undominated out-regular subgraph, and it can be found in polynomial time. Datta and Krishnamurthy [DK11] have improved the results of [AOV07]; they have shown that there is a nondeterministic logarithmic-space algorithm for finding an undominated out-regular subgraph on a planar bipartite digraph.

We generalize these results to non-win-lose bimatrix games in Chapter 8. We provide a one-to-one correspondence between a bimatrix game and an edge-weighted bipartite digraph. Furthermore, we prove that the problem of deciding whether a bimatrix game has a uniform Nash equilibrium is NP-complete even if the game corresponds to a planar digraph.

Finally, we mention the recent results related to the computational aspects of win-lose bimatrix games. Recently, Bilò and Mavronicolas [BM21] have shown the computational complexity of variants of win-lose bimatrix games and left a few open questions, for example, the complexity of deciding whether a given symmetric win-lose bimatrix game has a Pareto-optimal Nash equilibrium.

Graphical Games

The second method is usually called a graphical game, introduced by Kearn, Littman, and Singh [KLS01]. A graphical game consists of

- an undirected graph $G = (V, E)$, where V is a finite set of players and every edge in E represent the interaction between its endpoints,
- for each player $i \in V$, a finite set of strategies S_i,
- a payoff function $p_i : \times_{i \in N(i) \cup \{i\}} S_i \to \mathbb{R}_{\geq 0}$, where $N(i)$ is the set of neighbors of player i, that is, $N(i) = \{j \in V \ ; \ \{i, j\} \in E\}$.

Consider a graphical game whose graph has n vertices and degree is d that every player has m pure strategies. We need only $n \cdot m^{d+1}$ elements to explain this game; this is modest compared to $N \cdot k^N$.

Generally, it is intractable to compute a pure Nash equilibrium on a graphical game. Gottlob, Greco, and Scarcello [GGS05] have proven that deciding whether a given graphical game has a pure Nash equilibrium is NP-hard. In particular, we can straightforwardly see that the negative results presented in **Bimatrix Games** immediately follow if we note that every two-player game is a graphical game.

Recent studies motived by these negative facts have concentrated on the subclasses of graphical games; these subclasses are usually introduced by restricting (i) the graph structure, (ii) the number of pure strategies, or (iii) the properties of payoff functions.

Daskalakis and Papadimitriou [DP06] have shown that the existence of pure Nash equilibria on a graphical game is efficiently checkable, and we have a polynomial-time algorithm for resolving that problem if the graph has $O(\log n)$-treewidth, where n is the number of players. Elkind, Goldberg, and Goldberg [EGG06] have proven that a mixed Nash equilibrium can be found in polynomial time if a graph is a path, i.e., every vertex has at most two neighbors, and every player has two pure strategies.

Polymatrix Games A polymatrix game is the best well-known subclass of graphical games. Each edge on the graph in a polymatrix game is assigned a two-player game, a bimatrix game, between endpoints. Thus, a polymatrix game consists of

- an undirected graph $G = (V, E)$,
- for each player $i \in S$, a finite set of strategies S_i,
- for each edge $\{i, j\} \in E$, a bimatrix game $\langle M_{i,j}, M_{ji} \rangle$, where $M_{i,j}$ and $M_{j,i}$ imply the payoff matrix for the player i and j, respectively.

The total payoff for each player $i \in V$ is the sum of the payoffs from the bimatrix game engaged; $p_i(s) = \sum_{j \in N_i} M_{i,j}(s_i, s_j)$ when a pure strategy profile $s = (s_v)_{v \in V}$ are selected.

Cai, Condogan, Daskalakis, and Papadimitriou [Cai+16] have proven that we have a polynomial-time algorithm to compute a mixed Nash equilibrium if a polymatrix game is zero-sum. Their proof showed that a mixed Nash equilibrium could be found by linear programming. Unfortunately, it is also generally hard to compute a mixed Nash equilibrium on a polymatrix game. Recently, Deligkas, Fearnley, and Savani [DFS20] have shown that it is PPAD-complete to find a mixed Nash equilibrium on a tree polymatrix game even if each player has twenty pure strategies.

Network Coordination Games A network coordination game is a class obtained by further restricting polymatrix games. In a network coordination game, both players receive the same reward from the bimatrix game, i.e., $M_{i,j} = M_{j,i}$. Interestingly, every network coordination game always has a pure Nash equilibrium. However, it is hard to find a pure Nash equilibrium for them; Cai and Daskalakis [CD11] have shown that the problem of computing a pure Nash equilibrium on a network coordination game is PLS-complete even if the graph has degree five and every player has two pure strategies. Their PLS-hardness results depended on the hardness of MaxCut; Elsässer and Tscheuschner [Tsc10; ET11] proved that MaxCut is PLS even on graphs with the maximum degree five. Remark that MaxCut is a class of network coordination games; this is obvious. There is a positive result about computing a pure Nash equilibrium on a network coordination game: Poljak [Pol95] has proven that we have a polynomial-time algorithm for solving MaxCut with on a cubic graph.

Naturally, it seems easy to compute a mixed Nash equilibrium on a network coordination game. Cai and Daskalakis [CD11] have pointed out such a problem belongs to PPAD$\cap$PLS. However, it is now only still unknown whether we have a polynomial-time algorithm for finding a mixed Nash equilibrium on a network coordination game, but also it is still unknown that such a problem is PPAD$\cap$PLS-complete.

Other Succinct Games

An (implicit) congestion game is a succinctly represented game. The set of pure strategies of n players is a set of subsets of edges, which indicates resources. Each edge e has a delay function $d_e : [n] \to \mathbb{Z}_{>0}$. When all players select a subset each from its pure strategy set $(P_1, \ldots, P_n)$, we calculate the congestion $c(e)$ of each edge e, and the (negative) payoff player $i \in [n]$ is $-\sum_{e \in P_i} d(c(e))$. Fabrikant, Papadimitriou, and Talwar [FPT04] have proven that the problem of computing a pure Nash equilibrium on a congestion game is PLS-complete.

An *implicit* congestion game is a succinct variant of congestion games. In the implicit case, E is the set of edges of an actual network, every player is associated with two nodes in the network, and the set of strategies of a player $i \in [n]$ is the set of all paths between the two points corresponding to the player i. Fabrikant, Papadimitriou, and Talwar [FPT04] also have shown the complexity of finding a pure Nash equilibrium on an implicit congestion game is a PLS-complete problem.

Simple Stochastic Games are also well-known succinct games. Such games were formulated by Shapley [Sha53] and Condon [Con92]. It is known that the problem of solving a simple stochastic game belongs to UNIQUE END OF POTENTIAL LINE [Fea+20]. However, the hardness of the problem related to simple stochastic games is still open.

Unfortunately, it seems to be hard to compute a mixed Nash equilibrium on an implicit congestion game. Babichenko and Rubinstein [BR21] have shown the problem of finding a mixed Nash equilibrium on an implicit congestion game is PPAD∩PLS-complete.

Furthermore, the *anonymous games* also have been widely studied. This is a generalization of symmetric games: Each player is different but cannot distinguish each other. Their utility depends on the partition into their selected strategies. Chen, Durfee, and Orfanou [CDO15] have proven that computing a Nash equilibrium on an anonymous game is PPAD-complete.

7.2.2 Other Computational Aspects of Game Theory

Stable Matching

The stable matching model introduced by Gale and Shapley [GS13] is one of the most important mathematical models for matching problems. An instance of a matching problem consists of an undirected graph and a preference list. Thus, this model is naturally generalized to hypergraphs, called a hypergraphic preference system. It is not difficult to see that there exists an instance of the stable matching problem in the hypergraphic preference system that has no stable matching. Therefore, we consider in this chapter the following relaxation concept called a fractional matching. In the ordinary stable matching problem, the value 0 or 1 is assigned to each edge. On the other hand, in a fractional matching, a real number between zero and one is assigned to each edge. Fortunately, Aharoni and Fleiner [AF03] showed that every hypergraphic preference system always has at least stable fractional matching.

In this chapter, we consider the problem of finding a stable fractional matching in a hypergraphic preference system. Kintali, Poplawski, Rajaraman, Sundraman, and Teng [Kin+13] proved that the problem of finding a stable fractional matching in a hypergraphic preference system is PPAD-complete. We introduce in this chapter the problem of finding a stable fractional matching in a hypergraphic preference system whose maximum degree is bounded by some constant. The proof by Kintali, Poplawski, Rajaraman, Sundaram, and Teng [Kin+13] implies the PPAD-completeness of the problem of finding a stable fractional in a hypergraphic preference system whose maximum degree is five. We first prove that the problem of finding a stable fractional matching in a hypergraphic preference system whose maximum degree is at most two can be solved with stable fractional matching. We second prove that the problem of finding a stable fractional matching in a hypergraphic preference system is PPAD-complete even if the maximum degree is three. Finally, we prove that the problem of finding an approximate stable fractional matching in a hypergraphic preference system is also PPAD-complete.

Recently, Csáji [Csá21] has proven that the problem of computing a stable fractional matching is PPAD-complete even if every hyperedge contains at most three vertices and every vertex joins at most three hyperedges.

Chapter 8

Uniform Nash on Planar Bimatrix Games

8.1 Basics

The research interaction between Game Theory and Theoretical Computer Science has helped to study the computational issues underlying fundamental game-theoretic notions deeply. A seminal topic of these research streams is to clarify the complexity of computing Nash equilibria in multi-player non-cooperative games [Pap94b]. A Nash equilibrium is an intuitive and essential concept of rationality in which no player could unilaterally deviate from her selected strategy to improve her reward.

The results of the last two decades are unraveling the complexity of computing Nash equilibria [CD06; DGP09]. The most important result is the intractability for computing Nash equilibria in two-player games even if both players gain only a unit reward at most [AKV05]. Furthermore, it is well known that the best algorithm has an exponential worst-case running time in a two-player setting [SS04].

Motived from the above negative facts, some studies have concentrated on the complexity of computing specified classes of equilibria, such as pure, mixed, or correlated equilibria [FPT04; Pap94b]. This chapter focuses on the uniform Nash equilibria, i.e., Nash equilibria consisting of strategies in which strategies are played according to a uniform distribution. A uniform Nash equilibria do not always exist. Bonifaci, Iorio, and Laura [BIL08] showed that it is NP-complete to decide whether a given two-player game has uniform Nash equilibria even if every element of payoff matrices is either 0 or 1. On the other hand, Addario-Berry, Olver, and Vetta [AOV07] proved that there is a polynomial-time algorithm to find a uniform Nash equilibrium in such a two-player game if the graph defined by the game is planar.

We prove, in this chapter, that the problem of deciding whether a two-player game that induces a planar graph has uniform Nash equilibria is NP-complete. Furthermore, we state that if both players' payoff matrices consist of only three types of non-zero components, then such a problem is also NP-complete.

8.1.1 Our Results

This thesis sharpens the tractability-intractability boundary for the problem of computing uniform Nash equilibria on bimatrix games. We prove that deciding whether a bimatrix game has a uniform Nash equilibrium is intractable even if the game is planar. Our proof shown in Section 8.3 is inspired by the proof techniques used in

[AKV05; BIL08; CS05]. The key technique in proving the main result is the construction of the clause-variable gadget, shown in Figure 8.2.

We also discuss the number of types of non-zero elements that appear in the payoff matrices in Section 8.4.

8.2 Preliminaries

8.2.1 Bimatrix Games and Uniform Nash Equilibria

A bimatrix game is a two-player strategic form game where both payers have a finite set of strategies. We call the first (resp. second) player the *row* (resp. *column*) player. Here, we denote by R and C the set of strategies for the row and column player, respectively. A bimatrix game is specified by non-negative real matrices $M_R, M_C \in \mathbb{R}_{\geq 0}^{R \times C}$; the rows and columns of both matrics are indexed by the pure strategies of the players.

A mixed strategy is a probability distribution over pure strategies, i.e., a vector $x_R \in \mathbb{R}_{\geq 0}^{R}$ such that $\sum_{r_i \in R} x_R(r_i) = 1$ is a mixed state of the rwo player. Similarly, a vector $x_C \in \mathbb{R}_{\geq 0}^{C}$ such that $\sum_{c_j \in C} x_C(c_j) = 1$ is a mixed strategy of the column player. For a mixed strategy x, the support $\mathrm{supp}(x)$ of x is the set of pure strategies i such that $x(i) > 0$. A mixed strategy x is said to be *uniform* if for every $i \in \mathrm{supp}(x)$, $x(i) = 1/|\mathrm{supp}(x)|$.

When the row player plays a mixed strategy x_R and the column player plays a mixed strategy x_C, the expected payoffs for the row player and the column player is $x_R^\top M_R x_C$ and $x_R^\top M_C x_C$, respectively. A Nash equilibrium of the bimatrix game (M_R, M_C) is a pair of mixed strategies (x_R^*, x_C^*) satisfying for all mixed strategies $x_R \in \mathbb{R}_{\geq 0}^{R}$ of the row player, $(x_R^*)^\top M_R x_C^* \geq x_R^\top M_R x_C^*$, and for all mixed strategies $x_C \in \mathbb{R}_{\geq 0}^{C}$ of the column player, $(x_R^*)^\top M_C x_C^* \geq (x_R^*)^\top M_C x_C$. We call a Nash equilibrium (x_R^*, x_C^*) a uniform Nash equilibrium if both mixed strategies are uniform.

Relationship between Edge-Weighted Digraphs and Bimatrix Games We now describe how to construct an edge-weighted bipartite digraph from a bimatrix game. Our notation and definition introduced below are based on [AOV07; BIL08]. Note that they only considered unweighted digraphs. We generalize their notion to edge-weighted digraphs.

Let (M_R, M_C) be a bimatrix game, where the sets of strategies for the row players and the column players are R and C, respectively. Now, we define the edge-weighted bipartite digraph $G = (V, E, w)$ induced by the bimatrix game (M_R, M_C). Here, $w : E \to \mathbb{R}_{\geq 0}$ represents weightes on edges. The set V of vertices is $R \cup C$. We have an arc (r_i, c_f) if $M_R(r_i, c_v) > 0$, and the weight on the arc (r_i, c_j) is equal to $M_R(r_i, c_j)$. Similarly, we have an arc (c_j, r_i) if $M_C(r_i, c_j) > 0$, and the weight on the arc (c_j, r_i) is equal to $M_C(r_i, c_j)$.

The bimatrix game (M_R, M_C) is a planar bimatrix game if the edge-weighted digraph G is induced by (M_R, M_C) is a planar graph.

Remark that it is possible that the reverse transformation. Thus, we can construct a bimatrix game from an edge-weighted bipartite digraph.

Let $G = (V, E, w)$ be an edge-weighted bipartite digraph. Since G is bipartite, we can naturally separate the set of vertices V into two disjoint subsets R and C. We regard R and C as the sets of pure strategies for the row and column players, respectively. The payoff matrix of the row player M_R is defined as follows: For each $r_i \in R$, $M_R(r_i, c_j) = w(r_i, c_j)$ if $(r_i, c_j \in E$; otherwise $M_R(r_i, c_j) = 0$. The definition of the payoff matrix of the column player M_C is similar, i.e., for each $c_j \in C$, $M_C(r_i, c_j) = w(c_j, r_i)$ if $(c_j, r_i) \in E$; otherwise $M_C(r_i, c_j) = 0$.

From the above observation, we obtain the following statement.

Lemma 8.1. *For every bimatrix game, there is a corresponding edge-weighted bipartite digraph; vice versa.*

Undominated Out-Regular Subgraphs For a finite subset $S \subseteq V$, let $G[S]$ be the subgraph induced by S. We denote by $\Gamma^+(v)$ the set of out-neighbors of a vertex $v \in V$, and by $\Gamma^+_S(v)$ the set of out-neighbors of a vertex $v \in V$ that are in $S \subseteq V$.

Fix arbitrary non-empty subset of strategies $S = S_R \cup S_C \subseteq V$, where $\emptyset \neq S_R \subseteq R$ and $\emptyset \neq S_C \subseteq S$. The induced subgraph $G[S]$ is (α, β) out-regular if the following two conditions: (1) for every $r_i \in S_R$, $\sum_{c_j \in \Gamma^+_S(r_i)} w(r_i, c_j) = \alpha$; (2) for every $c_j \in S_C$, $\sum_{r_i \in \Gamma^+_S(c_j)} w(c_j, r_i) = \beta$. We say that G has an out-regular subgraph if there is a triple is a triple (α, β, S) such taht $G[S]$ is (α, β) out-regular.

A vertex $v \in V \setminus S$ dominates an (α, β) out-regular subgraph $G[S]$ if it satifies that eigther $\sum_{c_j \in \Gamma^+_S(v)} w(v, c_j) > \alpha$ if $v \in R$; or $\sum_{r_i \in \Gamma^+_S(v)} w(v, r_i) > \beta$ if $v \in C$. We say that an (α, β) out-regular subgraph $G[S]$ is undominated if there are no vertices that dominate $G[S]$. We say that G has an undominated out-regular subgraph if there is a triple (α, β, S) such that $G[S]$ is an undominated (α, β) out-regular subgraph.

8.2.2 Problem Formulation

We present a list of computational problems that will be used in this chapter.

Definition 8.2. BIMATRIXGAME
Input:

- a bimatrix game (M_R, M_C).

Task: Decide whether

- there is a uniform Nash equilibrium (x_R, x_C).

Definition 8.3. PLANAR BIMATRIX GAME
Input:

- a planar bimatrix game (M_R, M_C)

Task: Decide whether

- there is a uniform Nash equilibrium (x_R, x_C).

Definition 8.4. UNDOMINATED OUTREGULAR
Input:

- an edge-weighted bipartite digraph $G = (V, E, \{w(e)\}_{e \in E})$.

Task: Decide whether

- there is G has an undominated out-regular subgraph.

Definition 8.5. UNDOMINATED OUTREGULAR ON PLANAR
Input:

- an edge-weighted bipartite planar digraph $G = (V, E, \{w(e)\}_{e \in E})$.

Task: Decide whether

- there is G has an undominated out-regular subgraph.

8.3 On the Complexity of Planar Bimatrix Games

The purpose of this section is to prove the NP-hardness for the problem of deciding whether a given bimatrix game (M_R, M_C) has uniform Nash equilibrium even if the given game is planar. So, we prove the next theorem.

Theorem 8.6. *The problem* PLANAR BIMATRIX GAME *is* NP*-complete.*

It is not hard to see that PLANAR BIMATRIX GAME belongs to NP. Hence, it suffices to show the NP-hardness of this problem. To prove Theorem 8.6, we show the following three lemmata. The proofs of these lemmata are given in Sections 8.3.2, 8.3.2, and 8.3.3, respectively.

Lemma 8.7. *Two problems* BIMATRIXGAME *and* UNDOMINATED OUTREGULAR *are polynomially equivalent.*

Lemma 8.8. 3-SAT *is polynomial-time reducible to* UNDOMINATED OUTREGULAR

Lemma 8.9. *We can efficiently modify the graph given by the proof of Lemma 8.8 to be planar.*

Theorem 8.6 immediately follows from the above lemmata. The last two lemmata show that the problem UNDOMINATED OUTREGULAR ON PLANAR is NP-hard. Since this problem also belongs to NP, we obtain the NP-completeness of UNDOMINATED OUTREGULAR ON PLANAR. From Lemma 8.7, we see that PLANAR BIMATRIX GAME is NP-complete. Note that our proof shown in Section 8.3.1 shows that the two problems PLANAR BIMATRIX GAME and UNDOMINATED OUTREGULAR ON PLANAR are polynomially equivalent.

8.3.1 Proof of Lemma 8.7

Lemma 8.7 is a generalization of Lemma 5 of Bonifaci, Iorio, and Laura [BIL08]. They only focused on the relationship between a win-lose bimatrxi game and an unweighted bipartite digraph. Here, we extend their results to the correspondence between a bimatrix game and an edge-weighted bipartite digraph.

The reduction is natural. As mentioned in Lemma 8.1, we have a one-to-one correspondence between a bimatrix game and an edge-weighted bipartite digraph. Furthermore, our constructions shown in Section 8.2 can be done in polynomial time. Hence, to prove Lemma 8.7, it suffices to show the following theorem.

Theorem 8.10. *Fix a subset of pure strategies $S = S_R \cup S_C \subseteq V$, where $\emptyset \neq S_R \subseteq R$ and $\emptyset \neq S_C \subseteq C$. Then, $G[S]$ is an undominated (α,β) out-regular subgraph if and only if the pair of uniform strategies (x_R, x_C) such that* $\mathrm{supp}(x_R) = S_R$ *and* $\mathrm{supp}(x_C) = S_C$ *is a Nash equilibrium.*

Proof. First, we assume that $G[S]$ is an undominated (α,β) out-regular subgraph. Then, we show that the pair of uniform strategies (x_R, x_C) defined as $\mathrm{supp}(x_R) = S_R$ and $\mathrm{supp}(x_C) = S_C$ is a Nash equilibrium. When the column player playes x_C, the row player gains the expected payoff $\alpha/|S_C|$ by playing $r_i \in S_R$, but gains the expected payoff at most $\alpha/|S_C|$ by playing $r_{i'} \in R \setminus S_R$. Hence, the uniform strategy x_R is a best response to x_C for the rwo player. Similarly, we can see that the uniform strategy x_C is a best response to x_R for the column player. Thus, (x_R, x_C) is a uniform Nash equilibrium.

Next, we assume that the pair of uniform strategies (x_R, x_C) such that $\mathrm{supp}(x_R) = S_R$ and $\mathrm{supp}(x_C) = S_C$ is a Nash equilibrium. For the sake of contradiction, we suppose that there are strategies $r_i, r_k \in R$ such that $\sum_{c_j \in \Gamma_S^+(r_i)} w(r_i, c_j) > \sum_{c_\ell \in \Gamma_S^+(r_k)} w(r_k, c_\ell)$ when the column player plays x_C. In this case, the rwo player gains the expected payoff $(1/|S_C|)\sum_{c_j \in \Gamma_S^+(r_i)} w(r_i, c_j)$ by playing r_i. On the other hand, she gains the expected payoff $(1/|S_C|)\sum_{c_\ell \in \Gamma_S^+(r_k)} w(r_k, c_\ell)$ by playing r_k. From our supposition, the row player will get a lager expected payoff for playing r_i than for playing r_k, which contradicts that (x_R, x_C) is a Nash equilibrium. Therefore, $G[S]$ is an (α,β) out-regular subgraph. Using a similar technique, it is easy to see that $G[S]$. □

8.3.2 Proof of Lemma 8.8

Let $\phi : \{0,1\}^n \to \{0,1\}$ be a 3-CNF formula. Without loss of generality, we assume that ϕ has m clauses, and each clause has exactly three literals.

Now, we construct an edge-weighted bipartite digraph G_ϕ such that every undominated out-regular subgraph leads to a satisfying assignment of ϕ. The set of vertices of G_ϕ consists of variable vertices $x_i, \bar{x}_i$ for each $i \in [n]$, clauses vertices C_j for each $j \in [m]$, and the additional vertex a, variable coordinating vertices $y_i, \bar{y}, z_i$ for each $i \in [n]$, and clause coordinating vertices $v_{1,j}, v_{2,j}, u_{1,j}, u_{2,j}$ for each $j \in [m]$.

We create an arc (C_j, ℓ_i) when the clause C_j contains the literal ℓ_i. Notice that we are distinguishing between negation and non-negation. Furthermore, we define the weight on (C_j, ℓ_i) as 1. We add an arc (a, C_j) for every clause $j \in [m]$, and define the weight on (a, C_j) as 1. For each clause $j \in [m]$, we create the clause gadget such as the left of Figure 8.1. For each $i \in [n]$, we create the variable gadget such as the right of Figure 8.1.

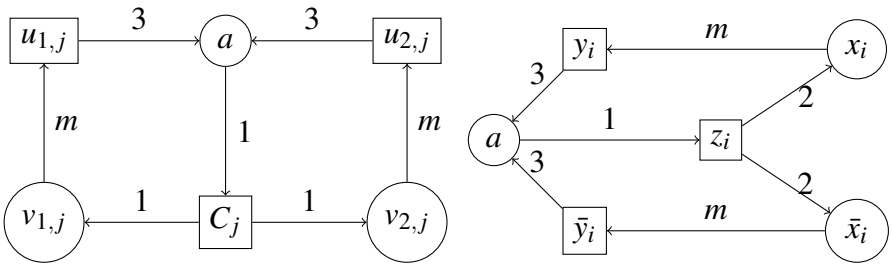

FIGURE 8.1: The left is the clause gadget; the right is the variable gadget.

From the above construction, the digraph G_ϕ has $5n+5m+1$ vertices and is a bipartite digraph. Hence, this construction can be done in polynomial time. It is sufficient to prove the following theorem to complete the proof of Lemma 8.8.

Theorem 8.11. *A Boolean formula ϕ has a satisfying assignment if and only if the graph G_ϕ has an undominated out-regular subgraph.*

Proof. We assume that ϕ has a satisfying assignment ξ. Then, we define the subset S of vertices $\{C_j\,;\, j\in[m]\}\cup\{x_i,y_i\,;\,\xi(i)=1\}\cup\{\bar{x}_i,\bar{y}_i\,;\,\xi(i)=0\}\cup\{a\}$. Also, we add some $v_{1,j},v_{2,j}$ into S so that $|\Gamma^+_S(C_j)|=3$ for every clause $j\in[m]$, and furthermore, we add $u_{k,j}$ into S when the vertex $v_{k,j}$ is in S.

We show that $G_\phi[S]$ is an undominated out-regular subgraph. It is easy to see that $G_\phi[S]$ is $(3,m)$ out-regular. By construction, there are no dominating vertices. Therefore, $G[S]$ is an undominated out-regular subgraph.

Next, we assume that the graph G_ϕ has an undominated out-regular subgraph. More specifically, we suppose that the subset S of vertices induces an undominated out-regular subgraph $G_\phi[S]$.

Note that the subgraph G_ϕ is strongly connected, and every directed edge has a weight of at least one. This implies that $\sum_{u\in\Gamma^+_S(v)} w(v,u)>0$ for every vertex $v\in S$. From this fact, we can see that there is at least one vertex $u\in S$ such that the arc (u,v) is in G_ϕ.

Lemma 8.12. *For every $v\in S$, it satisfies that $\sum_{u\in\Gamma^+_S(v)} w(v,u)>0$.*

Proof. For the sake of contradiction, we assume that there is a vertex $v\in S$ such that $\sum_{u\in\Gamma^+_S(v)} w(v,u)=0$. We take a vertex $u\in S$ that belongs to the different part from v. Since the graph G_ϕ is strongly connected, there is an arc $(v',u)\in E(G_\phi)$. If $v'\in S$, it contradicts that S induces an out-regular subgraph. On the other hand, if $v'\in V(G_\phi)\setminus S$, it contradicts that $G_\phi[S]$ has no dominating vertices. □

From Lemma 8.12, we can see that the additional vertex a is contained in S. The graph given by removing the additional vertex a from the graph G_ϕ is an acyclic digraph. Note that any induced subgraph $H(S)$ on an acyclic digraph H has a vertex v such that $\Gamma^+_S(v)=\emptyset$, which implies that $\sum_{u\in\Gamma^+_S(v)} w(v,u)=0$.

The subset S contains at least one clause vertex C_j. To prove this, it suffices to prove that any variable coordinating vertex z_i is not contained in S. Recall that the out-neighbors of the additional vertex a are only clause vertices C_j or variable coordinating vertices z_i. For the sake of contradiction, we assume that there is a

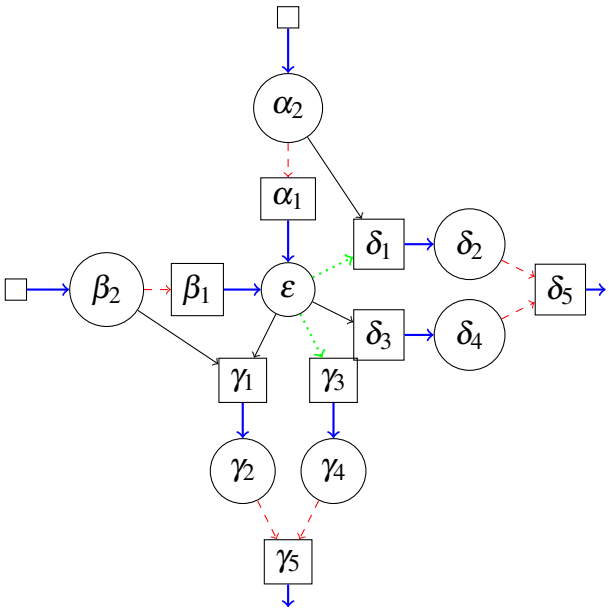

FIGURE 8.2: The construction of the Clause-Variable Gadget. Here, weights on fine, dashed, thick, and dotted are 1, m, 3, and $m-1$, respectively.

vertex z_i belonging to S. This implies that at least one of x_i and $\bar{x}_i$ is contained in S. If $x_i \in S$, then $y_i \in S$. Hence, we have $\sum_{u \in \Gamma_S^+(y_i)} w(y_i, u) \neq \sum_{u \in \Gamma_S^+(z_i)} w(z_i, u)$, which contradicts that S induces an out-regular subgraph. Therefore, $z_i \notin S$ holds.

Since $z_i \notin S$ for any $i \in [n]$, there is a clause vertex C_j contained in S. Then, S contains at least one of the variable vertices in $\Gamma_S^+(C_j)$. Assuming that any such vertices do not join in S, it holds that $0 < \sum_{u \in \Gamma_S^+(C_j)} w(C_j, u) \leq 2$. We take a clause coordinating vertex $v_{k,j} \in S$. In this case, we also have $u_{k,j} \in S$ by Lemma 8.12 and the construction of G_ϕ. Since $w(u_{k,j}, a) = 3$, a contradiction arises. Therefore, S contains at least one of the variable vertices $\Gamma_S^+(C_j)$. In the following, we write x_i such a variable vertex. From Lemma 8.12, the variable coordinating vertex y_i is also contained in S if $x \in S$. Note that y_i is the outneighbor of x_i. Furthermore, it is easy to see that $\bar{x}_i$ is not contained in S if $x_i \in S$; the vertex z_i dominates $G[S]$ when S contains both x_i and $\bar{x}_i$.

Finally, we show that every clause vertices C_j are contained in S. Assuming that there exists a clause vertex $C_{j'}$ such that $C_{j'} \notin S$, the sum of weights on arcs outgoing from the additional vertex a is strictly less than m. On the other hand, the sum of weights on arcs outgoing from the variable vertex contained in S is equal to m. This is a contradiction.

We can construct a satisfying assignment ξ of the Boolean formula ϕ the subset S. For each $i \in [n]$, we define $\xi(i) = 1$ if $x_i \in S$ and $\xi(i) = 0$ if $\bar{x}_i \in S$. Here, for any $i \in [n]$ such that neither x_i nor $\bar{x}_i$ is contained in S, we assign arbitary value to $\xi(i)$ (without loss of genrality, $\xi(i) = 0$). From the construction of the graph G_ϕ, it satisfies that $\phi(\xi) = 1$. $\square$

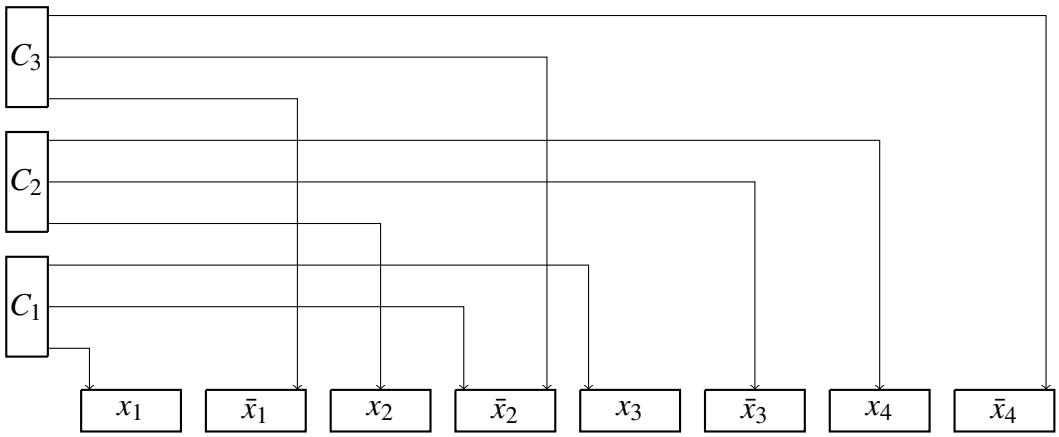

FIGURE 8.3: The digraph constructed from a given 3-SAT instance $\phi(x) = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_4)$, where $C_1 = x_1 \vee \bar{x}_2 \vee x_3$, $C_2 = x_2 \vee \bar{x}_3 \vee x_4$, and $C_3 = \bar{x}_1 \vee x_3 \vee \bar{x}_4$.

8.3.3 Proof of Lemma 8.9

In this section, we embed the graph G_ϕ constructed in Section 8.3.2 into a plane. Thus, we eliminate crossing points. From simple observation, the only directed edges that may not be planar are those between the variable vertices and the clause vertices. To eliminate every crossing point, we construct a clause-variable gadget shown in Figure 8.2 and insert this gadget into each crossing point.

First, we embed the graph G_ϕ into the $3m \times 2mn$ grid plane. We arrange all clause vertices to the left and all variable vertices to the bottom. Each clause vertex has three points to connect variable vertices, and each variable vertex has m points to connect clause vertices. At the variable vertices, we label each point to connect adjacent edges with $1, 2, \ldots, m$. If the variable vertex x_i is contained in the clause vertex C_j, then the corresponding arc extends horizontally from any point of C_j to just above the j-th connection point of x_i. The edge is then lowered vertically to the j-th point of x_i. We repeat this procedure to connect every directed edge between clause and variable vertices. We lexicographically chose a connecting point of a clause vertex without loss of generality. We illustrate an example of our construction in Figure 8.3. In the remainder of this section, we will use the clause-variable gadget to remove the crossing points of directed edges that occur in the above construction.

Before inserting clause-variable gadgets, we take the following steps. For each arc (C_j, ℓ_i), we add two vertices $\ell_i^{C_j}$ and $C_j^{\ell_i}$, and add three directed edges $(C_j, \ell_i^{C_j})$, $(\ell_i^{C_j}, C_j^{\ell_i})$, and $(C_j^{\ell_i}, \ell_i)$. We define the weights on these edges as follows: $w(C_j, \ell_i^{C_j}) = 1$, $w(\ell_i^{C_j}, C_j^{\ell_i}) = m$, and $w(C_j^{\ell_i}, \ell_i) = 3$. By this process, every edge between clause and variable vertices has weights with three.

We insert the clause-variable gadget into each crossing point. Here, we regard a crossing point as the vertex ε. We illustrate an example for inserting the clause-variable gadget in Figure 8.4.

By the construction, it is obvious that the clause-variable gadget is planar. We can convert the graph G_ϕ to the planar graph by inserting the clause-variable gadget to each crossing point. Of course, our reconstruction can be done in polynomial time

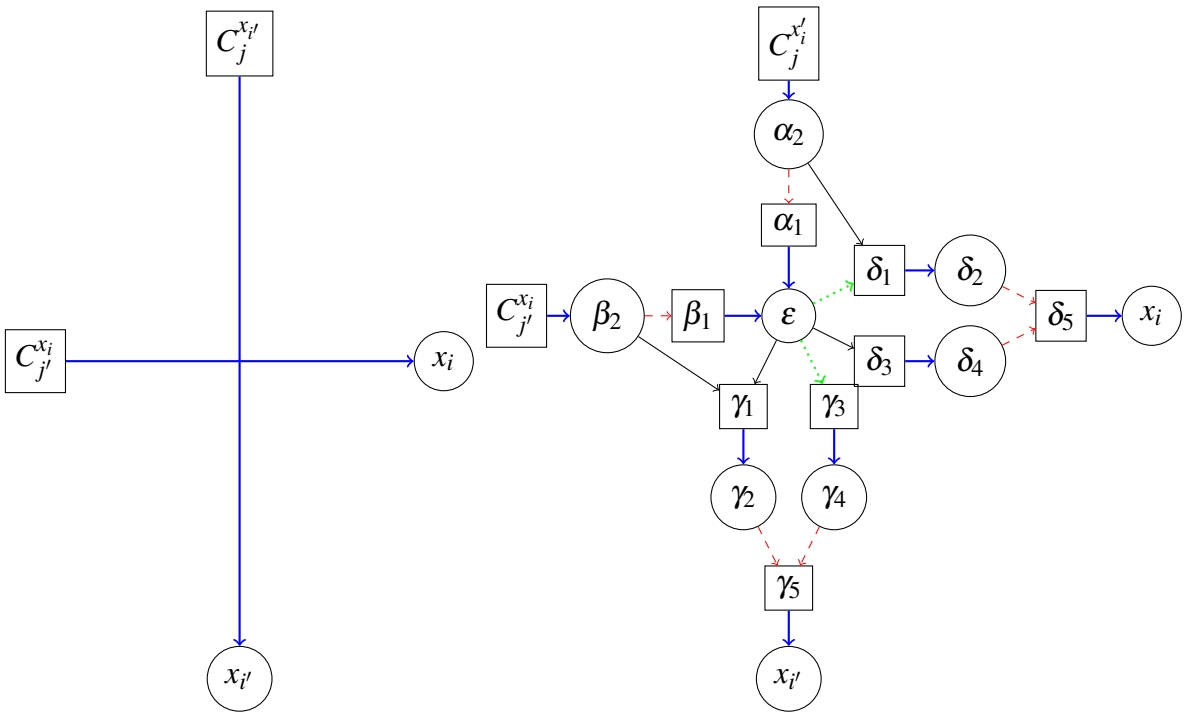

FIGURE 8.4: An example of the elimination of a crossing point (the left figure: an example of a possible crossing point) with a clause-variable gadget (the right figure: an example of the Clause-Variable Gadget). Here, weights on fine, dashed, thick, and dotted edges are 1, m, 3, and $m-1$, respectively.

since the clause-variable gadget has at most a constant number of vertices, and some polynomial bounds the number of possible crossing points.

We denote by H_ϕ the planar digraph constructed from G_ϕ by the above procedure. What remains is to prove that H_ϕ has an undominated out-regular subgraph if and only if the Boolean formula ϕ has a satisfying assignment. To prove this, it suffices to show the next lemma.

Lemma 8.13. *The graph G_ϕ has an undominated out-regular subgraph if and only if the graph H_ϕ has an undominated out-regular subgraph.*

Proof. First, we assume that G_ϕ has an undominated out-regular subgraph $G_\phi[S]$, i.e., the subset of vertices $S \subseteq V(G_\phi)$ induces an undominated out-regular subgraph. Then, we show how to construct a subset T of vertices on H_ϕ that induces the undominated out-regular subgraph $H_\phi[T]$. We define the subset T by performing the procedure shown in Algorithm 4.

Algorithm 4 A procedure to construct the subset T

We initialize T with S.
for all pair of a clause vertex C_j and a variable vertex ℓ_i such that $(C_j, \ell_i) \in E(G_\phi[S])$ **do**
    we add the vertices $\ell_i^{C_j}$ and $C_j^{\ell_i}$ to T.
end for
if there are some vertices on the path from $C_j^{\ell_i}$ to ℓ_i **then**
    we add some vertices as follows:
    1. When the path passes through the clause-variable gadget horizontally, we add the vertices β_2, β_1, ε, δ_3, δ_4 , and δ_5 to T.
    2. When the path passes through the clause-variable gadget vertically, we add the vertices α_2, α_1, ε, γ_3, γ_4, and γ_5 to T.
end if
for all $\varepsilon \in T$ satisfying that $\sum_{u \in \Gamma_T^+(\varepsilon)} w(\varepsilon, u) < m$ **do**
    we add out-neighbors of ε as appropriate. More precisely, we add the vertex γ_1 to T if $\beta_1 \notin T$, else add the vertex δ_1 to T if $\alpha_1 \notin T$.
end for

What remains is to prove that the graph $H_\phi[T]$ is an undominated out-regular subgraph. By construction of H_ϕ, the vertices that can dominate $H_\phi[T]$ are the variable coordinating vertices z_i for $i \in [n]$ or α_2 and β_2 that are on the clause-variable gadget. As mentioned in the proof of Theorem 8.10, any variable coordinating vertex z_i does not dominate the graph $H_\phi[T]$ since at most one of the variable vertices x_i and $\bar{x}_i$ is contained in S. The vertex α_2 does not dominate $H_\phi[T]$ since at most one of the vertices α_1 and δ_1 is contained in T by the above procedure. Similarly, we see that the vertex β_2 is not a dominating vertex. Hence, $H_\phi[T]$ is an undominated subgraph. It is straightforward to see that the total weight on out-neighbors equals either 3 or m for every vertex in T. Therefore, $H_\phi[T]$ is out-regular. From the above argument, H_ϕ has an undominated out-regular subgraph.

Next, we assume that the graph H_ϕ has an undominated out-regular subgraph $H_\phi[T]$, i.e., the subset T induces an undominated out-regular subgraph. Then, we

show how to construct the subset S of vertices on G_ϕ that induces an undominated out-regular subgraph. We construct the subgraph S by performing the procedure shown in Algorithm 5.

Algorithm 5 A procedure to construct the subset S

```
We initialize S with {a} ∪ {C_j ; j ∈ [m]}.
for all i ∈ [n] do
    if x_i ∈ T then
        we add the vertices x_i and y_i to S
    end if
    if x̄_i ∈ T then
        we add the vertices x̄_i and ȳ_i to S
    end if
end for
for all j ∈ [m] satisfying that Σ_{u∈Γ_S^+(C_j)} w(C_j, u) = 1 do
    we add the vertices v_{1,j}, u_{1,j}, v_{2,j}, and u_{2,j} to S
end for
for all j ∈ [m] satisfying that Σ_{u∈Γ_S^+(C_j)} w(C_j, u) = 2 do
    we add the vertices v_{1,j} and u_{1,j} to S.
end for
```

What remains is to prove that the graph $G_\phi[S]$ is an undominated out-regular subgraph. It is easy to see that $G_\phi[S]$ is out-regular. Note that exactly one of variable vertices x_i and $\bar{x}_i$ is contained in T for each $i \in [n]$. If not, then the variable coordinating vertex z_i dominates the graph $G_\phi[S]$. This fact implies that the graph $G_\phi[S]$ is an undominated subgraph. From the above argument, the subset S induces an undominated out-regular subgraph. □

8.4 Types of Non-Zero Elements

This section focuses on the number of types of non-zero elements that appear in the payoff matrices.

We call a bimatrix game a win-lose bimatrix game if every element of payoff matrices is either 0 or 1. By definition, both players on a win-lose bimatrix game have exactly one type of non-zero element. Addario-Berry, Olver, and Betta [AOV07] have proven that a planar bimatrix game always has unique Nash equilibria. They also have shown that we have a polynomial-time algorithm for finding a unique Nash equilibrium of a planar Nash equilibrium. This fact can be rephrased as follows: There exists a polynomial-time algorithm to decide whether a given planar bimatrix game has a uniform Nash equilibrium if both players' payoff matrices have only one type of non-zero element.

In the proof of Theorem 8.6, we construct the reduction that both players' payoff matrices consist of three types of non-zero elements; each element of one player's payoff matrix is either m, $m-1$, 1, or 0, and each element of the other player's payoff matrix is either 3, 2, 1, or 0. Hence, we obtain the following corollary.

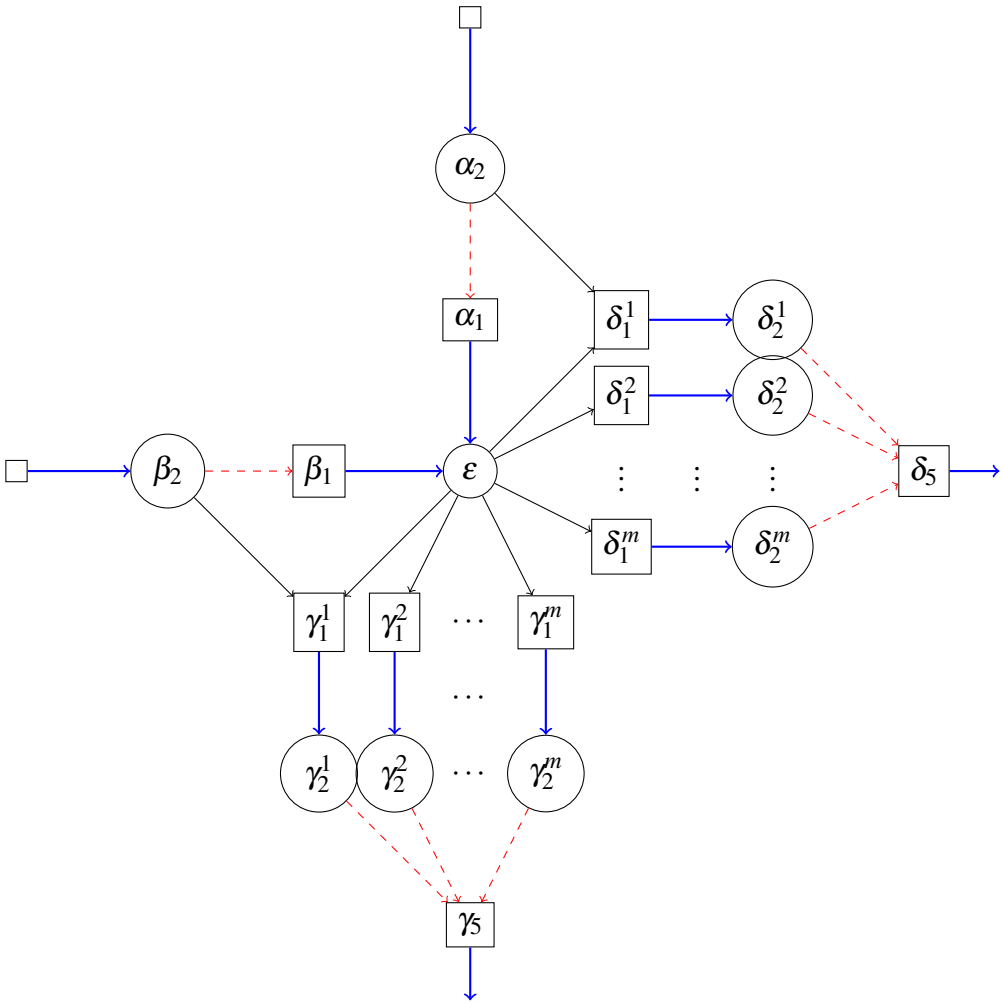

FIGURE 8.5: Modification of Clause-Variable Gadget. Here, weights on fine, dashed, and thick edges are 1, m, and 3, respectively.

Corollary 8.14. *It is* NP*-complete to decide whether a given planar bimatrix game has a uniform Nash equilibrium even if both players' payoff matrices consist of three types of non-zero elements.*

We can reduce the number of types of non-zero components that appear in the payoff matrices. We show that the problem PLANAR BIMATRIX GAME is NP-hard even if one player's payoff matrix consists of three types of non-zero components and the other player's payoff matrix consists of two types of non-zero components. To prove this fact, we modify the clause-variable gadget to one shown in Figure 8.5. We obtain the following theorem.

Theorem 8.15. *It is* NP*-complete to decide whether a given planar bimatrix game has uniform Nash equilibria even if one player's payoff matrix consists of three types of non-zero elements and the other player's payoff matrix consists of two types of non-zero elements.*

Proof. The proof is almost the same as the proof of Lemma 8.13. □

8.5 Conclusion

We have studied, in this chapter, the complexity of computing uniform Nash equilibria on planar bimatrix games. We have shown that the problem of deciding whether a given planar bimatrix game has a uniform Nash equilibrium is NP-complete even if both players' payoff matrix consists of two types of non-zero components.

This thesis left an open question worth considering: How hard to distinguish the existence of uniform Nash equilibria on a planar bimatrix game satisfying that both players' payoff matrices consist of two types of non-zero components?

Chapter 9

Discrete Preference Games and Network Coordination Games

9.1 Basics

A *graphical game*, introduced by Kearns, Littman, and Singh [KLS01], is a succinctly represented multi-player strategic-form game. A graphical game consists of an undirected graph $G = (V, E)$, where V is a finite set of players and every edge in E represents the interaction between its endpoints, and for each player $i \in V$, a finite set of strategies S_i and a cost function $C_i : \times_{j \in N(i) \cup \{i\}} S_j \to \mathbb{R}_{\geq 0}$, where $N(i)$ is the set of neighbors of player i, that is $N(i) = \{ j \in V \ ; \ \{i, j\} \in E \}$.

We refer to the tuple of strategies played by each player as a strategy profile. The set $S = \times_{i \in [n]} S_i$ is called a set of strategy profiles. For a player $i \in V$, we denote by S_{-i} the set of strategies for all players except i. For a strategy profile $x = (x_i)_{i \in V}$, we denote by x_i the strategy played by a player i, and by x_{-i} the strategies of all players except i.

A pure Nash equilibrium is an intuitive and essential concept of rationality. A pure Nash equilibrium is a strategy profile such that every player has no incentive to change her selected strategy. $x^* = (x_i^*)_{i \in V}$ is a pure Nash equilibrium if for each player $i \in V$, and every strategy $x_i \in S_i$, $C_i(x_i^*, x_{-i}^*) \leq C_i(x_i, x_{-i}^*)$ holds. Note that graphical games do not always have pure Nash equilibria because any two-player strategic-form game is a graphical game.

The complexity of finding a pure Nash equilibrium on a graphical game is one of the most interesting topics of Algorithmic Game Theory. Unfortunately, it is intractable to determine the existence of a pure Nash equilibrium for a graphical game. Gottlob et al. [GGS05] have proven that the problem of deciding whether there exists a pure Nash equilibrium for a given graphical game is NP-hard. On the other hand, Daskalakis and Papadimitriou [DP06] have shown that it is polynomial-time decidable whether there is a pure Nash equilibrium on a graphical game whose players' network has $O(\log n)$-treewidth. Furthermore, their result has stated that we can find a pure Nash equilibrium in polynomial time if it exists for such a graphical game.

We wish to understand what properties make it hard to compute a pure Nash equilibrium for a graphical game and make it easy to do. This paper focuses on the class of graphical games that are guaranteed the existence of pure Nash equilibria. There are well-known classes of graphical games that always have a pure Nash equilibrium; a *discrete preference game* and a *network coordination game* are examples. The formal definitions of these games can be found in Section 9.2.

The results of the hardness of computing a pure Nash equilibrium for a discrete preference game and a network coordination game are known. Lolakapuri et al. [Lol+19] have proven that finding a pure Nash equilibrium on a discrete preference game is PLS-complete even if the maximum degree of the players' network is 7. Cai and Daskalakis [DP06] have shown the PLS-completeness of computing a pure Nash equilibrium for a network coordination game even if the maximum degree of the players' network is five and each player has two strategies. On the other hand, Lolakapuri et al. [Lol+19] have proven that a pure Nash equilibrium for a discrete preference game on a tree metric space is polynomial-time computable.

The following computational aspects of pure Nash equilibria are still unknown for discrete preference games and network coordination games:

- How hard is computing a pure Nash equilibrium for a discrete preference game on a non-tree metric space?
- Can we find pure Nash equilibria in polynomial time for a discrete preference game and a network coordination game if the maximum degree of the players' networks is four?

This chapter deals with the above topics. In particular, we discuss the complexity of finding a pure Nash equilibrium for a discrete preference game on neither $O(\log n)$-treewidth nor a tree metric space. First, we estimate an upper bound of the number of iterations of the best response dynamics for a discrete preference game on a discrete metric space to compute a pure Nash equilibrium. Second, we provide a sufficient condition that we have a polynomial-time algorithm to find a pure Nash equilibrium of such a discrete preference game. Finally, we present a relationship between discrete preference games and network coordination games.

9.1.1 Our Results

Discrete preference game on the discrete metric A discrete metric space with at least three strategies is one of the simple non-tree metric spaces. It is important to consider and understand the complexity of a discrete preference game with such a metric space. Recall that a discrete preference game was formulated based on a decision-making model wherein agents decide which platform to use [Lol+19]. Note that the metric space implies that every agent is only interested in being on the same or different platforms. Namely, a discrete preference game on a discrete metric space is one of the uncomplicated settings of decision-making models.

Section 9.3 provides an upper bound for the number of iterations of the best response dynamics for a discrete preference game on the discrete metric. We show that the best response dynamics halts after quadratic iterations when we view the given parameter as a constant.

Discrete preference games on grid graph Our motive behind this work is to clarify the boundary between cases where we can find a pure Nash equilibrium in polynomial time for the numbers of players and strategies and cases where it is not[1]. As mentioned above, the complexity of finding a pure Nash equilibrium on a graph with degree four is unknown for discrete preference games and network coordination games. Hence, it is important to clarify the complexity of finding a pure Nash equilibrium for a discrete preference game on a two-dimensional grid graph. A two-dimensional grid graph is one of the graphs whose maximum degree is four.

Remark that Section 9.5.1 shows the relationship between discrete preference games and network coordination games. In particular, we prove that there is a polynomial-time reduction such that the structure of the players' network is preserved from a discrete preference game to a network coordination game. This fact implies that the hardness result for a network coordination game straightforwardly follows from the hardness results for a discrete preference game. Therefore, it is a natural approach to deal with the complexity of discrete preference games first, under negative conjecture.

Section 9.4 provides a sufficient condition that we have a polynomial-time algorithm to find a pure Nash equilibrium of a discrete preference game on a grid graph. To prove this condition, Section 9.4.1 introduces a more general discrete preference game, called a cartesian game, in which a discrete preference game is constructed from some discrete preference games. We show that it can efficiently construct a pure Nash equilibrium for a cartesian game from pure Nash equilibria for the discrete preference games that form that cartesian game. Our results are the first polynomial-time computability of discrete preference games on neither $O(\log n)$-treewidth nor tree metric spaces.

9.2 Preliminaries

Discrete Preference Games A discrete preference game with a parameter $\mathscr{G} = (G, \mathscr{M}, (\beta_i), \alpha)$, which is the fundamental model introduced by Chierichetti et al. [CKO18], consists of an unweighted graph $G = (V, E)$, a finite metric space $\mathscr{M} = (L, d)$, a preferred strategy $\beta_i \in L$ for each player $i \in V$, and a parameter $0 \leq \alpha < 1$. Every player has the identical strategy set L. Given a strategy profile $x = (x_i)_{i \in V}$, the cost for player i is:

$$c_i(x) = \alpha d(x_i, \beta_i) + (1 - \alpha) \sum_{j \in N(i)} d(x_i, x_j). \tag{9.1}$$

Network Coordination Games A network coordination game $\mathscr{G} = (G, (S_i), (C_{i,j}, C_{j,i}))$ is defined by: (i) an undirected graph $G = (V, E)$; (ii) for each edge $\{i, j\} \in E$, there are two cost functions $C_{i,j} \colon S_i \times S_j \to \mathbb{R}_{\geq 0}$ and $C_{j,i} \colon S_j \times S_i \to \mathbb{R}_{\geq 0}$ that satisfy $C_{i,j}(x_i, x_j) = C_{j,i}(x_j, x_i)$ for all $x_i \in S_i$ and $x_j \in S_j$; (iii) the total cost for a player $i \in V$ is the sum of all her costs, i.e., $C_i(x) = \sum_{j \in N(i)} C_{i,j}(x_i, x_j)$.

[1] Note that the games dealt with in this paper are guaranteed the existence of pure Nash equilibria. This fact implies that we can trivially find it in polynomial time when we regard the number of players as a constant. On the other hand, it is not always possible to compute a pure Nash equilibrium in polynomial time when the number of strategies is considered a constant.

Potential Games A game is an *exact potential game* if there exists a function $\Phi\colon S \to \mathbb{R}$ such that for all $s_{-i} \in S_{-i}$, $s_i, t_i \in S_i$, $\Phi(s_i, s_{-i}) - \Phi(t_i, s_{-i}) = C_i(s_i, s_{-i}) - C_i(t_i, s_{-i})$, where S is the set of strategy profiles, and C_i is the cost for player i. In this thesis, we call such a function an exact potential function for the game. A game is a *generalized ordinal potential game* if there is a function $\Phi\colon S \to \mathbb{R}$ such that for all $s_{-i} \in S_{-i}$, $s_i, t_i \in S_i$, $\Phi(s_i, s_{-i}) > \Phi(t_i, s_{-i})$ whenever $C_i(s_i, s_{-i}) > C_i(t_i, s_{-i})$. We call such a function a generalized ordinal potential function for the game. The existence of pure Nash equilibrium for some variants of potential games can be found in Chapter 2.2 of [LCS16]. Notice that we can easily see that an exact potential game always has a pure Nash equilibrium since the best response dynamics, which is described in Section 9.3, halts after the finite number of iterations.

9.3 Discrete Preference Games on the Discrete Metric

In this section, we estimate an upper bound of the number of iterations of the best response dynamics for a discrete preference game with a parameter on the discrete metric. Recall that the two-strategic case was studied by previous work [CKO18; FGV16]. We now focus on the case when there are three or more strategies.

Let $\mathscr{G} = \big(G = (V,E), \mathscr{M} = (L,d), (\beta_i)_{i\in V}, \alpha\big)$ be a discrete preference game with a parameter $0 \le \alpha < 1$, where the metric $\mathscr{M}$ is the discrete metric. We define the potential Φ for a strategy profile $x = (x_i)_{i\in V}$ as

$$\Phi(x) = \sum_{i\in V} \alpha d(x_i, \beta_i) + (1-\alpha) \sum_{\{i,j\}\in E} d(x_i, x_j). \tag{9.2}$$

Note that the above function Φ is an exact potential function for $\mathscr{G}$ [CKO18]. Therefore, any player decreases her cost if and only if the potential Φ also decreases by the same value.

The best response dynamics follow the following procedure: While the current strategy profile is not a pure Nash equilibrium, pick an arbitrary player who wants to deviate from the current strategy profile, and she changes her strategy to a best response. Note that there is only one player moving strategy at each step in the best response dynamics.

The following theorem gives an upper bound of the number of iterations of the best response dynamics for a discrete preference game with a parameter. Then we denote by $\Phi_{\max}$ the potential of the start point.

Theorem 9.1. *The best response dynamics for a discrete preference game on the discrete metric halts after at most $\mu(\alpha)^{-1}\Phi_{\max}$ steps, where $\mu(\alpha) = \min\{1-\alpha, \alpha + (1-\alpha)\lfloor 1-\alpha/(1-\alpha)\rfloor, -\alpha + (1-\alpha)\lfloor 1+\alpha/(1-\alpha)\rfloor\}$, and $\Phi_{\max}$ is the potential of the start point.*

Proof. We denote by $D_i(x)$ the number of neighbors that plays a strategy different from i's strategy, i.e., $D_i(x) = |\{j \in N(i); x_i \neq x_j\}|$. Then given a strategy profile $x = (x_v)_{v\in V}$, the cost of player i is: $c_i(x) = \alpha d(x_i, \beta_i) + (1-\alpha)D_i(x)$.

We consider the best response dynamics. Let $x = (x_i)_{v\in V}$ be a current strategy. In this step, a player $i \in V$ moves her strategy from x_i to y_i, and i's cost strictly decreases. Then, there are three possible cases:

- If $x_i \neq \beta_i \neq y_i$, then we have $0 < c_i(x_i, x_{-i}) - c_i(y_i, x_{-i}) = (1-\alpha)(D_i(x_i, x_{-i}) - D_i(y_i, x_{-i}))$. In this case, it satisfies that $D_i(x_i, x_{-i}) - D_i(y_i, x_{-i}) > 0$. Notice that $D_i(\cdot)$ is a non-negative integer, and hence, $\Phi(x_i, x_{-i}) - \Phi(y_i, x_{-i}) = c_i(x_i, x_{-i}) - c_i(y_i, x_{-i}) \geq (1-\alpha) > 0$ holds.

- If $x_i \neq \beta_i = y_i$, then we have $0 < c_i(x_i, x_{-i}) - c_i(y_i, x_{-i}) = \alpha + (1-\alpha)(D_i(x_i, x_{-i}) - D_i(y_i, x_{-i}))$. In this case, it satisfies that $D_i(x_i, x_{-i}) - D_i(y_i, x_{-i}) > -\alpha/(1-\alpha)$. Note that $D_i(\cdot)$ is a non-negative integer. If $-\alpha/(1-\alpha)$ is an integer, then it holds that $D_i(x_i, x_{-i}) - D_i(y_i, x_{-i}) \geq 1 - \alpha/(1-\alpha)$, and otherwise, it holds that $-\alpha/(1-\alpha) < \lceil -\alpha/(1-\alpha) \rceil = \lfloor 1 - \alpha/(1-\alpha) \rfloor \leq D_i(x_i, x_{-i}) - D_i(y_i, x_{-i})$. Hence, we have $D_i(x_i, x_{-i}) - D_i(y_i, x_{-i}) \geq \lfloor 1 - \alpha/(1-\alpha) \rfloor$. This implies that $\Phi(x_i, x_{-i}) - \Phi(y_i, x_{-i}) = c_i(x_i, x_{-i}) - c_i(y_i, x_{-i}) \geq \alpha + (1-\alpha)\lfloor 1 - \alpha/(1-\alpha) \rfloor > 0$.

- If $x_i = \beta \neq y_i$, then we have $0 < c_i(x_i, x_{-i}) - c_i(y_i, x_{-i}) = -\alpha + (1-\alpha)(D_i(x_i, x_{-i}) - D_i(y_i, x_{-i}))$. In this case, it satisfies that $D_i(x_i, x_{-i}) - D_i(y_i, x_{-i}) > \alpha/(1-\alpha)$. Note that $D_i(\cdot)$ is a non-negative integner. If $\alpha/(1-\alpha)$ is an integer, then it holds that $D_i(x_i, x_{-i}) - D_i(y_i, x_{-i}) \geq 1 + \alpha/(1-\alpha)$, and otherwise, it holds that $\alpha/(1-\alpha) < \lceil \alpha/(1-\alpha) \rceil \leq \lfloor 1 + \alpha/(1-\alpha) \rfloor \leq D_i(x_i, x_{-i}) - D_i(y_i, x_{-i})$. Therefore, we have $D_i(x_i, x_{-i}) - D_i(y_i, x_{-i}) \geq \lfloor 1 + \alpha/(1-\alpha) \rfloor$. This implies that $\Phi(x_i, x_{-i}) - \Phi(y_i, x_{-i}) = c_i(x_i, x_{-i}) - c_i(y_i, x_{-i}) \geq -\alpha + (1-\alpha)\lfloor 1 + \alpha/(1-\alpha) \rfloor > 0$.

From the above observation, at each step of the best response dynamics, the potential Φ decreases at least $\mu(\alpha)$. Therefore, the best response dynamics halts after at most $\mu(\alpha)^{-1}\Phi_{\max}$ steps, where $\Phi_{\max}$ is the potential for the start point. □

Remark 9.2. When we view the given parameter α as a constant, the best response dynamics halts after at most $O(n^2)$ iterations by Theorem 9.1 because the exact potential function $\Phi(x)$ in $O(n^2)$. Note that it halts after at most $O(n)$ iterations in the two-strategic setting by a technical way to select a player who moves her strategy at each step even if the parameter α is non-constant [CKO18].

9.4 Discrete Preference Games on Grid Graphs

We present the special case of a discrete preference game whose pure Nash equilibria can be found in polynomial time beyond $O(\log n)$-treewidth and tree metrics.

We consider a discrete preference game with a parameter on a k-dimensional grid graph. We call a graph $G = (V, E)$ a k-dimensional grid graph if there are k positive integers $M_1, \ldots, M_k$ such that $V = [M_1] \times \cdots \times [M_k]$ and there is an edge $\{i, j\} \in E$ if $\|i - j\|_1 = 1$.

Now, we prove that there is a polynomial-time algorithm to compute a pure Nash equilibrium for a discrete preference game $\mathscr{G} = (G, \mathscr{M}, (\beta_i)_{i \in V}, \alpha)$ on k dimensional grid graph G if the following two conditions hold:

(A) $\mathscr{M} = (L, d)$ is a 1-product metric of k arbitrary finite metrics $\mathscr{M}_1 = (L_1, d_1), \ldots, \mathscr{M}_k = (L_k, d_k)$; and

(B) we can select a strategy $\beta^t_{i_t} \in L_t$ for each $t \in [k]$ and each $i_t \in [M_t]$ so that the set $\{\beta^t_{i_t} \in L_t \ ; \ t \in [k], i_t \in [M_t]\}$ satisfies the following condition: for each player $i = (i_1, \ldots, i_k) \in V$, the preferred strategy β_i is a form of $(\beta^1_{i_1}, \ldots, \beta^k_{i_k}) \in L$.

In other words, the second condition implies that the t-th element of the preferred strategy β_i of a player i is $\beta^t_{i_t}$ whenever the t-th player of i is i_t. For instance, we consider a discrete preference game on a two-dimentional grid graph $G = ([N_1] \times [N_2], E)$ satisfying the above two conditions. The condition (B) implies that the player (i_1, i_2) prefers the strategy $(\beta^1_{i_1}, \beta^2_{i_2})$ if a player (i_1, j_2) and a player (j_1, i_2) prefer strategies $(\beta^1_{i_1}, \beta^2_{j_1})$ and $(\beta^1_{j_1}, \beta^2_{i_2})$, respectively.

Theorem 9.3. *We suppose that a discrete preference game $\mathscr{G} = (G, \mathscr{M}, (\beta_i)_{i \in V}, \alpha)$ on k-dimensional grid graph G satisfies the above two conditions (A) and (B). In this case, we can find a pure Nash equilibrium for $\mathscr{G}$ in polynomial time.*

To prove the above theorem, we introduce a cartesian product of discrete preference games, a game formed by some discrete preference games, in Section 9.4.1. We prove that a pure Nash equilibrium for a cartesian product of discrete preference games is efficiently constructible from pure Nash equilibria for ingredients of the original one. After that, we give the proof of Theorem 9.3 in Section 9.4.2.

9.4.1 Cartesian Products of Discrete Preference Games

A cartesian product of discrete preference games is formed by k discrete preference games with a parameter. This model represents an environment where every player belongs to k different communities and makes decisions within each community. Here, we suppose that each community forms its own network. When we assume that every community forms the same network, such a game is a discrete preference game on a product metric space — we discuss the complexity of such a model in Section 9.4.3.

Informally speaking, a players' network on a cartesian product of discrete preference games is a cartesian product of graphs that are networks for the ingredients of the original one. Each player is a tuple of players on ingredients, and they communicate along only one edge on an ingredient. Furthermore, a strategy space comprises a product metric space[2].

We define a cartesian product of graphs and a cartesian product of discrete preference games.

Definition 9.4 (Cartesian Product of Graphs)**.** Let $G_1 = (V_1, E_1), \ldots, G_k = (V_k, E_k)$ be simple graphs. We define the cartesian product of graphs $G = (V, E)$ as follows: Each node $v \in V$ is a k-tuple of nodes $(v_1, \ldots, v_k)$, where $v_i \in V_i$ for each $i \in [k]$. There is an edge $\{v, u\} \in E$ if and only if there exists only one $t \in [k]$ such that $\{v_t, u_t\} \in E_t$ and $v_i = u_i$ for all $i \neq t$. We denote $G_1 \square G_2 \square \cdots \square G_k$ by the cartesian product of k graphs $G_1, \ldots, G_k$.

Definition 9.5 (Cartesian Product of Discrete Preference Game)**.** Fix a parameter $0 \leq \alpha \leq 1$. Given k discrete preference games with a parameter $\mathscr{G}_1 = (G_1 = (V_1, E_1), \mathscr{M}_1 =$

[2]This paper deals with a case of a 1-product metric space, but it can also be generalized to any ℓ-product metric space.

$(L_1,d_1),(\beta_i^1)_{i\in V_1},\alpha)$, $\ldots,\mathscr{G}_k=(G_k=(V_k,E_k),\mathscr{M}_k=(L_k,d_k),(\beta_i^k)_{i\in V_k},\alpha)$, we define the discrete preference game $\mathscr{G}=(G=(V,E),\mathscr{M}=(L,d),(\beta_i)_{i\in V},\alpha)$ as follows: the graph $G:=G_1\,\square\,G_2\,\square\cdots\square\,G_k$, the strategy space $\mathscr{M}$ is a 1-product metric of $\mathscr{M}_1,\ldots,\mathscr{M}_k$, and, for each player $i=(i_1,\ldots,i_k)\in V$, the strategy profile $\beta_i=(\beta_{i_1}^1,\ldots,\beta_{i_k}^k)$. In this case, we call $\mathscr{G}$ the cartesian game constructed from discrete preference games $\mathscr{G}_1,\ldots,\mathscr{G}_k$.

Let $\mathscr{G}$ be the cartesian game constructed from discrete preference games $\mathscr{G}_1,\ldots,\mathscr{G}_k$. When we are given a strategy profile x^t of $\mathscr{G}_t$ for each $t\in[k]$, we interpret $x=(x^t)_{t\in[k]}$ as the strategy profile of $\mathscr{G}$ such that each player $i=(i_1,\ldots,i_k)\in V$ plays the strategy $x_i=(x_{i_1}^1,\ldots,x_{i_k}^k)$.

The next theorem states that we can efficiently construct a pure Nash equilibrium for $\mathscr{G}$ from pure Nash equilibria for $\mathscr{G}_1,\ldots,\mathscr{G}_k$.

Theorem 9.6. *Suppose that $\mathscr{G}=(G,\mathscr{M},(\beta_i)_{i\in V},\alpha)$ is a Cartesian game constructed from k discrete preference games $\mathscr{G}_t=(G_t,\mathscr{M}_t,(\beta_i^t)_{i\in V_t},\alpha)$ for $t\in[k]$. In this case, a strategy profile $\hat{x}=(\hat{x}^t)_{t\in[k]}$ is a pure Nash equilibrium for $\mathscr{G}$, where $\hat{x}^t$ is arbitrary pure Nash equilibrium for $\mathscr{G}_t$.*

Proof. For each $t\in[k]$, we denote by $\mathscr{M}_t=(L_t,d_t)$ the t-th finite metric space. Note that each player $i=(i_1,\ldots,i_k)\in V$ plays a k-tuple $(x_{i_1}^1,\ldots,x_{i_k}^k)\in L_1\times\cdots\times L_k$ as her strategy. For a strategy profile $x=(x_i)_{i\in V}$ of $\mathscr{G}$, the cost c_i for a player $i=(i_1,\ldots,i_k)\in V$ is

$$c_i(x)=\alpha d(x_i,\beta_i)+(1-\alpha)\sum_{j\in N(i)}d(x_i,x_j).$$

For each $t\in[k]$ and each player $i_t\in V_t$, we denote by $c_{i_t}^t$ the cost function for i_t on $\mathscr{G}_t$. Note that for a strategy profile $x^t=(x_{i_t}^t)_{i_t\in V_t}$, the cost for a player $i_t\in V_t$ is

$$c_{i_t}^t(x^t)=\alpha d_t(x_{i_t}^t,\beta_{i_t}^t)+(1-\alpha)\sum_{j_t\in N^t(i_t)}d_t(x_{i_t}^t,x_{j_t}^t),$$

where $N^t(i_t)$ is the set of neighbors of the player i_t on the graph G_t.

As mentioned above, given a strategy profile $x^t=(x_{i_t}^t)_{i_t\in V_t}$ of the game $\mathscr{G}_t$ for each $t\in[k]$, we interpret the tuple $x=(x^t)_{t\in[k]}$ as the strategy profile of $\mathscr{G}$ such that each player $i=(i_1,\ldots,i_k)\in V$ plays the strategy $x_i=(x_{i_1}^1,\ldots,x_{i_k}^k)$. In this case, the cost for a player $i=(i_1,\ldots,i_k)$ holds that

$$\begin{aligned}
c_i(x)&=\alpha d(x_i,\beta_i)+(1-\alpha)\sum_{j\in N(i)}d(x_i,x_j)\\
&=\sum_{t\in[k]}\alpha d_t(x_{i_t}^t,\beta_{i_t}^t)+(1-\alpha)\sum_{s\in[k]}\sum_{j\in N(i|s)}\sum_{t\in[k]}d_t(x_{i_t}^t,x_{j_t}^t)\\
&=\sum_{t\in[k]}\alpha d_t(x_{i_t}^t,\beta_{i_t}^t)+(1-\alpha)\sum_{s\in[k]}\sum_{j\in N(i|s)}\left(d_s(x_{i_s}^s,x_{j_s}^s)+\sum_{t\neq s}d_t(x_{i_t}^t,x_{j_t}^t)\right)\\
&=\sum_{t\in[k]}\alpha d_t(x_{i_t}^t,\beta_{i_t}^t)+(1-\alpha)\sum_{s\in[k]}\sum_{j\in N(i|s)}d_s(x_{i_s}^s,x_{j_s}^s)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{t\in[k]} \left(\alpha d_t(x^t_{i_t}, \beta^t_{i_t}) + (1-\alpha) \sum_{j_t \in N^t(i_t)} d_t(x^t_{i_t}, x^t_{j_t}) \right) \\
&= \sum_{t\in[k]} c^t_{i_t}(x^t),
\end{aligned}
$$

where $N(i \mid t) = \{j \in N(i) \; ; \; j_t \neq i_t\}$, which is the subset of the neighbors of i on G that are adjacent to i by an edge on G_t. The fourth equality follows from the construction of the strategy profile $x = (x^t)_{t\in[k]}$. To show the fifth equality, we use the fact that $\sum_{j\in N(i|t)} d_t(x^t_{i_t}, x^t_{j_t}) = \sum_{j\in N^t(i_t)} d_t(x^t_{i_t}, x^t_{j_t})$.

Here, we prove that if a strategy profile $\hat{x}^t$ is a pure Nash equilibrium for $\mathscr{G}_t$ for each $t \in [k]$, then the strategy profile $\hat{x} = (\hat{x}^t)_{t\in[k]}$ is a pure Nash equilibrium for $\mathscr{G}$.

For the sake of a contradiction, we assume that some player $i = (i_1, \ldots, i_k) \in V$ can improve her cost by moving her strategy to $y_i = (y^1_{i_1}, \ldots, y^k_{i_k})$, i.e., it satisfies that $c_i(\hat{x}_i, \hat{x}_{-i}) > c_i(y_i, \hat{x}_{-i})$. Then we have

$$
\begin{aligned}
0 &< c_i(\hat{x}_i, \hat{x}_{-i_t}) - c_i(y_i, \hat{x}_{-i_t}) \\
&= \sum_{t\in[k]} c^t_{i_t}(\hat{x}^t_{i_t}, \hat{x}^t_{-i_t}) - \left(\sum_{t\in[k]} \alpha d_t(y^t_{i_t}, \beta^t_{i_t}) + (1-\alpha) \sum_{s\in[k]} \sum_{j\in N(i|s)} \sum_{t\in[k]} d_t(y^t_{i_t}, \hat{x}^t_{j_t}) \right) \\
&= \sum_{t\in[k]} c^t_{i_t}(\hat{x}^t_{i_t}, \hat{x}^t_{-i_t}) - \left(\sum_{t\in[k]} \alpha d_t(y^t_{i_t}, \beta^t_{i_t}) + (1-\alpha) \sum_{s\in[k]} \sum_{j\in N^s(i_s)} d_s(y^s_{i_s}, \hat{x}^s_{j_s}) \right) - \delta \\
&= \sum_{t\in[k]} c^t_{i_t}(\hat{x}^t_{i_t}, \hat{x}^t_{-i_t}) - \sum_{t\in[k]} c^t_{i_t}(y^t_{i_t}, \hat{x}^t_{-i_t}) - \delta \\
&= \sum_{t\in[k]} \left(c^t_{i_t}(\hat{x}^t_{i_t}, \hat{x}^t_{-i_t}) - c^t_{i_t}(y^t_{i_t}, \hat{x}^t_{-i_t}) \right) - \delta,
\end{aligned}
$$

where $\delta = (1-\alpha) \sum_{s\in[k]} \sum_{j\in N(i|s)} \sum_{t\neq s} d_t(y^t_{i_t}, \hat{x}^t_{j_t})$. Note that δ is non-negative.

Recall that for every $t \in [k]$, the strategy profile $\hat{x}^t$ is a pure Nash equilibrium for $\mathscr{G}_t$. Thus, it holds that $c^t_{i_t}(\hat{x}^t_{i_t}, \hat{x}^t_{-i_t}) \leq c_{i_t}(y^t_{i_t}, \hat{x}^t_{-i_t})$. Therefore, we have

$$
0 \leq \delta < \sum_{t\in[k]} \left(c^t_{i_t}(\hat{x}^t_{i_t}, \hat{x}^t_{-i_t}) - c^t_{i_t}(y^t_{i_t}, \hat{x}^t_{-i_t}) \right) \leq 0,
$$

which is a contradiction. □

9.4.2 Polynomial-time Solvability of Discrete Preference Games

In this section, we prove Theorem 9.3. From the condition (A), the strategy space $\mathscr{M}$ on $\mathscr{G}$ is a 1-product metric space of k finite metrics $\mathscr{M}_1, \ldots, \mathscr{M}_k$. From the condition (B), we can select a strategy $\beta^t_{i_t} \in L_t$ for each $t \in [k]$ and each $i_t \in [M_t]$ so that the set $\{\beta^t_{i_t} \in L_t \; ; \; t \in [k], i_t \in [M_t]\}$ such that for each player $i = (i_1, \ldots, i_k) \in V$, the preferred strategy β_i is equal to $(\beta^1_{i_1}, \ldots, \beta^k_{i_k})$.

Note that a k-dimensional grid graph $G = ([M_1] \times \cdots \times [M_k], E)$ is a Cartesian product of graphs $G_1 = ([M_1], E_1), \ldots, G_k = ([M_k], E_k)$, where there is an edge $\{i, j\} \in E_t$ if $|i - j| = 1$ for each $t \in [k]$. We use this fact to prove this theorem.

We decompose $\mathscr{G}$ into k discrete preference games $\mathscr{G}_1, \ldots, \mathscr{G}_k$ such that $\mathscr{G}$ is to be a cartesian game constructed from these subgames. For each $t \in [k]$, we defne

the t-th subgame as $\mathscr{G}_t = (G_t, \mathscr{M}_t, (\beta^t_{i_t})_{i_t \in [N_t]}, \alpha)$. It is easy to see that $\mathscr{G}$ is a product game constructed from k discrete preference games $\mathscr{G}_1, \ldots, \mathscr{G}_k$.

For every $t \in [k]$, we can find a pure Nash equilibrium $\hat{x}^t$ for $\mathscr{G}_t$ in polynomial time from the result of the polynomial-time computability for pure Nash equilibria on graphs that have $O(\log n)$-treewidth by Daskalakis and Papadimitriou [DP06]. By Theorem 9.6, the strategy profile $\hat{x} = (\hat{x}^t)_{t \in [k]}$ constructed from pure Nash equilibria $\hat{x}^1, \ldots, \hat{x}^k$ is a pure Nash equilibrium for $\mathscr{G}$. Therefore, we can compute a pure Nash equilibrium for $\mathscr{G}$ in polynomial time.

9.4.3 Properties of Discrete Preference Games on Product Metric Spaces

In the rest of this section, we focus on a discrete preference game on a product metric space. Recall that Lolakapuri, Bhaskar, Narayanam, Parija, and Dayama [Lol+19] considered a 1-product metric space of some path metric spaces and have proven that the problem of finding a pure Nash equilibrium for a discrete preference game on such a metric space is polynomial-time computable. Their algorithm, called *Product Metric Algo* produced in [Lol+19], gives us an approach to computing pure Nash equilibria for games: It may be easier to compute it when we can decompose the strategy space into an ℓ-product metric space for some $\ell \in \mathbb{Z}_{>0} \cup \{\infty\}$.

This section discusses the conditions under which such an approach, a decomposition approach, would work well. We prove that the decomposition approach always works for a discrete preference game on a 1-product metric space of arbitrary finite metric spaces.

Before discussing, we describe the more general model of discrete preference games, introduced by Lolakapuri, Bhaskar, Narayanam, Parija, and Dayama [Lol+19]. In their model, a game has edge weights and a penalty for each strategy instead of a parameter. A *discrete preference game with penalties* $\mathscr{G} = (G, \mathscr{M}, (p_i(s)))$ is defined by: (i) an edge-weighted graph $G = (V, E, (w_e)_{e \in E})$; (ii) each player $i \in V$ has a penalty $p_i(s) \in \mathbb{R}_{\geq 0}$ for each strategy $s \in L$, where L is a finite set of strategies; (iii) given a strategy profile $x = (x_i)_{i \in V}$, the cost for player $i \in V$ is:

$$c_i(x) = \sum_{s \in L} p_i(s) d(x_i, s) + \sum_{j \in N(i)} w_{ij} d(x_i, x_j). \tag{9.3}$$

Let $(G = (V, E, (w_e)_{e \in E}), \mathscr{M} = (L, d), (p_i(s))_{i \in V, s \in L})$ be a discrete preference game. For this game, we define the function $\Phi \colon L^V \to \mathbb{R}_{\geq 0}$ as follows:

$$\Phi(x) = \sum_{i \in V} \sum_{s \in L} p_i(s) d(s, x_i) + \sum_{\{i,j\} \in E} w_{ij} d(x_i, x_j). \tag{9.4}$$

We show that Φ is an exact potential function for a discrete preference game with penalties.

Lemma 9.7. *Let $\mathscr{G} = (G, \mathscr{M}, (p_i(s)))$ be a discrete preference game with penalties, where $G = (V, E, (w_e)_{e \in E})$, $\mathscr{M} = (L, d)$. The game $\mathscr{G}$ is an exact potential game.*

Proof. To see why the function Φ defined as Eq. (9.4) is an exact potential function for $\mathscr{G}$, for each player $i \in V$, all two strategies x_i and y_i, and all strategies x_{-i} of all players expect i, it holds that

$$\begin{aligned}
&\Phi(x_i, x_{-i}) - \Phi(y_i, x_{-i}) \\
&\quad = \sum_{s \in L} p_i(s) d(s, x_i) + \sum_{j \in N(i)} w_{ij} d(x_i, x_j) \\
&\qquad - \left(\sum_{s \in L} p_i(s) d(s, y_i) + \sum_{j \in N(i)} w_{ij} d(y_i, x_j) \right) \\
&\quad = c_i(x_i, x_{-i}) - c_i(y_i, x_{-i}).
\end{aligned}$$

Thus, a discrete preference game with penalties is an exact potential game. □

Now, we consider a discrete preference game on an ℓ-product metric space. Let $\mathscr{G} = (G = (V, E, (w_e)_{e \in E}), \mathscr{M} = (L, d), (p_i(s))_{i \in V, s \in L})$ be a discrete preference game with penalties, and let $\ell \in \mathbb{Z}_{>0} \cup \{\infty\}$. Suppose that $\mathscr{M}$ is an ℓ-product metric space formed by k finite metric spaces $\mathscr{M}_1 = (L_1, d_1), \ldots, \mathscr{M}_k = (L_k, d_k)$. For a strategy profile $x = (x_i)_{i \in V}$ on $\mathscr{G}$, we interpret it as that each player $i \in V$ plays k-tuple $x_i = (x_i^1, \ldots, x_i^k) \in L_1 \times \cdots \times L_k$. We denote by $x^t = (x_i^t)_{i \in V}$ the list on L_t for each strategy profile x on $\mathscr{G}$. For a strategy profile $x = (x^t)_{t \in [k]}$ on $\mathscr{G}$, we interpret the strategy x_i of player $i \in V$ as the k-tuple of strategies $x_i = (x_i^1, \ldots, x_i^k)$, where x^t is a strategy profile on L_t for each $t \in [k]$.

We decompose $\mathscr{G}$ into k discrete preference games on the partial metric spaces. In the following, we refer to such games as subgames. For each $t \in [k]$, the t-th subgame $\mathscr{G}_t$ of $\mathscr{G}$ is defined as $\mathscr{G}_t := (G, \mathscr{M}_t, (q_i^t(s^t))_{i \in V, s^t \in L_t})$, where $q_i^t(s^t) = \sum_{u \in L:\, u^t = s^t} p_i(u)$. Then the cost $c_i^t(x^t)$ of a player $i \in V$ on the t-th subgame is

$$c_i^t(x^t) = \sum_{s^t \in L_t} q_i^t(s^t) d_t(x_i^t, s^t) + \sum_{j \in N(i)} w_{ij} d_t(x_i^t, x_j^t). \tag{9.5}$$

We denote by $\Phi^{(t)}(x^t)$ the exact potential function for the t-th subgame, i.e.,

$$\Phi^{(t)}(x^t) = \sum_{i \in V} \sum_{s^t \in L_t} q_i^t(s^t) d_t(x_i^t, s^t) + \sum_{\{i,j\} \in E} w_{ij} d_t(x_i^t, x_j^t).$$

Furthermore, we define a function $\Psi(x)$ as

$$\Psi(x) = \sum_{t \in [k]} \Phi^{(t)}(x^t). \tag{9.6}$$

Theorem 9.8. *If the function Ψ defined in Eq. (9.6) is a generalized ordinal potential function for $\mathscr{G}$, then a strategy profile $\hat{x} = (\hat{x}^t)_{t \in [k]}$ is a pure Nash equilibrium for $\mathscr{G}$, where $\hat{x}^t$ is an arbitrary pure Nash equilibrium for the t-th subgame.*

Proof. For the sake of a contradiction, we assume that $\hat{x}$ is not a pure Nash equilibrium for $\mathscr{G}$, and thus, there is a player i that can improve her cost by moving to another strategy y_i from $\hat{x}_i$. Then it holds that $c_i(\hat{x}_i, \hat{x}_{-i}) > c_i(y_i, \hat{x}_{-i})$. Since Ψ is a

generalized ordinal potential function for $\mathscr{G}$, it satisfies that

$$0 < \Psi(\hat{x}_i, \hat{x}_{-i}) - \Psi(y_i, \hat{x}_{-i}) = \sum_{t \in [k]} \left(c_i^t(\hat{x}_i^t, \hat{x}_{-i}^t) - c_i^t(y_i^t, \hat{x}_{-i}^t) \right).$$

This implies that there is at least one $t \in [k]$ such that $c_i^t(\hat{x}_i^t, \hat{x}_{-i}^t) > c_i^t(y_i^t, \hat{x}_{-i}^t)$. Note that for each $t \in [k]$, $\hat{x}^t$ is a pure Nash equilibrium for the t-th subgame, and hence, we have $c_i^t(\hat{x}_i^t, \hat{x}_{-i}^t) \leq c_i^t(y_i^t, \hat{x}_{-i}^t)$. This is a contradiction. □

Corollary 9.9. *There is a polynomial-time algorithm to find a pure Nash equilibrium for $\mathscr{G}$ when the following two conditions hold: (i) for each t-th subgame; we have a polynomial-time algorithm to find a pure Nash equilibrium; and (ii) the function Ψ is a generalized ordinal potential function for $\mathscr{G}$.*

Unfortunately, the function Ψ is not always a generalized ordinal potential function for $\mathscr{G}$. Consider a discrete preference game on a discrete metric space with 2^k points. Note that such a metric can be written straightforwardly as an ∞-product metric of k discrete metric spaces. Example 9.10 shows that the metric decomposition approach is not easily applicable in such a game.

Example 9.10. For simplicity, we consider a discrete preference game with a parameter. Let $G = (V, E)$ be an unweighted graph, and $\mathscr{M}$ be a discrete metric space on 2^k strategies. For each player $v \in V$, we denote by $\beta_v \in [2^k]$ the preferred strategy of player v. Furthermore, we are given a parameter $1/2 < \alpha < 1$.

We decompose $\mathscr{M}$ into k metric spaces $(\{0,1\}, \delta)$. Each point $x \in [2^k]$ is interpreted as the binary string, and hence, the point on the t-th metric is the the t-th bit for x. Here, the function δ is also the discrete metric, i.e., $\delta(x,y) = 1$ if $x \neq y$, otherwise $\delta(x,y) = 0$. It is easy to see that for each pair of points $x, y \in L$, it satisfies that $d(x,y) = \max_{t \in [k]} \delta(x^t, y^t)$, where x^t is the t-th bit of x. The cost for player $i \in V$ on the t-th subgame is $c_i^t(x^t) = \alpha\delta(\beta_i^t, x_i^t) + (1-\alpha)\sum_{j \in N(i)} \delta(x_i^t, x_j^t)$.

Now, we show that this game does not satisfy the condition of Theorem 9.8. We fix any player $i \in V$. Then, we take strategies x_i and y_i for i and strategies x_{-i} for all others except i such that it satisfies the following conditions:

- x_i and y_i are different at only the t-th bit for some $t \in [k]$;
- $x_i^t = \beta_i^t$ and $x_i \neq \beta_i \neq y_i$;
- $D_i(x_i, x_{-i}) > D(y_i, x_{-i})$; and
- $D_i^t(x_i^t, x_{-i}^t) - D_i^t(y_i^t, x_{-i}^t) \leq 1$,

where $D_i(x_i, x_{-i})$ denotes that the number of i's neighbors that play a different strategy from x_i, and also we denote $D_i^t(x_i^t, x_{-i}^t)$ the number of i's neighbors whose t-th strategy is not x_i^t.

In this setting, the player i can decrease her cost by moving x_i to y_i. On the other hand, for the cost c_i^t for i on the t-th subgame, it follows that

$$\begin{aligned} & c_i^t(x_i^t, x_{-i}^t) - c_i^t(y_i^t, x_{-i}^t) \\ & \qquad = -\alpha + (1-\alpha)\left(D_i^t(x_i^t, x_{-i}^t) - D_i^t(y_i^t, x_{-i}^t) \right) \\ & \qquad \leq -\alpha + (1-\alpha) = 1 - 2\alpha < 0. \end{aligned}$$

The first equality holds from the second assumption, and note that $y_i^t \neq x_i^t$ in this setting. The second inequality follows from the fourth assumption. The final inequality follows from $1/2 < \alpha < 1$.

The above observation implies that i can not improve her cost in the t-th subgame. Notice that i moves only one bit from the first assumption, the function defined in Eq. (9.6) is not a generalize ordinal potential function. □

In the 1-product case, Ψ defined in Eq. (9.6) is always a generalized ordinal potential function for $\mathscr{G}$; more precisely, Ψ is an exact potential function for $\mathscr{G}$. To prove this, it suffices to show that Ψ is equal to Φ defined in Eq. (9.4). Theorem 9.11 states this fact.

Theorem 9.11. *If a metric space $\mathscr{M}$ is a 1-product metric space, then the function Ψ defined in Eq. (9.6) is an exact potential function for $\mathscr{G}$.*

Proof. We suppose that $\mathscr{M} = (L, d)$ is a 1-product metric space of k metric spaces, i.e., $d(x,y) = \sum_{t\in[k]} d_t(x^t, y^t)$ for all $x, y \in L$. It suffices to show that Φ defined in Eq. (9.4) equals to Ψ defined in Eq. (9.6). For any strategy profile x, we have

$$\begin{aligned}
\Phi(x) &= \sum_{i\in V}\sum_{s\in L} p_i(s) d(s, x_i) + \sum_{\{i,j\}\in E} w_{ij} d(x_i, x_j) \\
&= \sum_{i\in V}\sum_{s\in L} p_i(s) \sum_{t\in[k]} d_t(s^t, x_i^t) + \sum_{\{i,j\}\in E} w_{ij} \sum_{t\in[k]} d_t(s^t, x_i^t) \\
&= \sum_{t\in[k]}\sum_{i\in V}\sum_{s^t\in L_t}\sum_{u\in L\colon u^t = s^t} p_i(u) d_t(s^t, x_i^t) + \sum_{t\in[k]}\sum_{\{i,j\}\in E} w_{ij} d_t(s^t, x_i^t) \\
&= \sum_{t\in[k]} \Phi^{(t)} = \Psi(x).
\end{aligned}$$

In the second equality, we use the fact that $\mathscr{M}$ is a 1-product metric space. □

Immediately, we obtain the following corollary, which is a generalization of the result by Lolakapuri, Bhaskar, Narayanam, Parija, and Dayama [Lol+19].

Corollary 9.12. *There is a polynomial-time algorithm to find a pure Nash equilibrium for a discrete preference game if the following two conditions hold: (i) the metric space is a 1-product metric space; (ii) we have a polynomial-time algorithm to find a pure Nash equilibrium for every subgame.*

9.5 Relationship between Discrete Preference Games and Network Coordination Games

This section presents the relationship between network coordination games and discrete preference games. First, we show that every discrete preference game is reducible to a network coordination game in polynomial time. Second, we provide a class of network coordination games that are polynomial-time reducible to discrete preference games.

9.5.1 Reduction from Discrete Preference Games to Network Coordination Games

This section shows that a discrete preference game is reducible to a network coordination game in polynomial time.

Lemma 9.13. *Let $\mathscr{G}$ be a discrete preference game on a graph G. If we have a polynomial-time algorithm to compute pure Nash equilibria for network coordination games on the graph G, then it is also polynomial-time computable to find a pure Nash equilibrium for $\mathscr{G}$.*

Proof. To prove this, it is sufficient to construct a polynomial-time reduction from a discrete preference game to a network coordination game that preserves the structure of the players' network.

For each player $i \in V$, we denote as $\Delta_i := |N(i)|$. For each edge $e = \{i, j\} \in E$, we define the cost function $C_{i,j}$ as follows: for each element $(x_i, x_j) \in L \times L$,

$$C_{i,j}(x_i, x_j) = \sum_{k \in \{i,j\}} \Delta_k^{-1} \sum_{s \in L} p_k d(s, x_k) + w_e d(x_i, x_j)$$

which means the cost for i and j when i plays x_i and j plays x_j.

The exact potential function Φ' for a network coordination game is

$$\Phi'(x) = \sum_{e=\{i,j\} \in E} C_{i,j}(x_i, x_j) \tag{9.7}$$

for each strategy profile $x = (x_i)_{i \in V}$ [CD11].

To see that every pure Nash equilibrium for our network coordination game is also a pure Nash equilibrium for the given discrete preference game, we show that Φ' is also an exact potential function for a discrete preference game (see Theorem 2.2 in Chapter 2 of [LCS16]).

$$\begin{aligned}
\Phi'(x) &= \sum_{e=\{i,j\} \in E} C_{i,j}(x_i, x_j) \\
&= \sum_{e=\{i,j\} \in E} \sum_{k \in e} \Delta_k^{-1} \sum_{s \in L} p_k(x) d(s, x_k) + \sum_{e=\{i,j\} \in E} w_{i,j} d(x_i, x_j) \\
&= \sum_{i \in V} \left(\sum_{j \in N(i)} \sum_{s \in L} \Delta_i^{-1} p_i(s) d(s, x_i) \right) + \sum_{e=\{i,j\} \in E} w_{i,j} d(x_i, x_j) \\
&= \sum_{i \in V} \sum_{s \in L} p_i(s) d(s, x_i) + \sum_{e=\{i,j\} \in E} w_{i,j} d(x_i, x_j) \\
&= \Phi(x).
\end{aligned}$$

This is an exact potential function for a discrete preference game (see Eq. (9.4)). Hence, we complete constructing a polynomial-time reduction from a discrete preference game to a network coordination game. Note that our reduction does not change the structure of the graph G. □

Recall that Daskalakis and Papadimitriou [DP06] have proven that we can find a pure Nash equilibrium for a graphical game whose players' network has $O(\log n)$-treewidth in polynomial time. Apt, de Keijer, Rahn, Schäfer, and Simon [Apt+17] showed the polynomial-time computability of a pure Nash equilibrium for a network coordination game whose players' network contains at most one cycle. Therefore, we immediately obtain the following corollary by using these previous results together with Lemma 9.13.

Corollary 9.14. *There is a polynomial-time algorithm to compute a pure Nash equilibrium for a discrete preference game if the given players' network $G = (V, E)$ satisfies at least one of the following properties: (i) G has $O(\log |V|)$-treewidth; and (ii) G contains at most one cycle.*

9.5.2 Reduction from Network Coordination Games to Discrete Preference Games

In the previous section, we show that a discrete preference game is a special case of network coordination games. This section provides a class of network coordination games that are polynomial-time reducible to discrete preference games. Note that it is known that equilibrium computation for our class of network coordination games is easy by using a submodular function minimizing algorithm, such as [LSW15; Orl09]. However, by reducing a discrete preference game, we solve equilibrium computation faster (see Remark 9.18 for details).

We consider the complexity of a two-strategic network coordination game such that for each pair of players i, j, the cost $C_{i,j}$ between i and j is symmetric, i.e., $C_{i,j}(0,1) = C_{i,j}(1,0)$ and a submodular function, i.e.,

$$C_{i,j}(1,0) + C_{i,j}(0,1) \geq C_{i,j}(1,1) + C_{i,j}(0,0), \tag{9.8}$$

where we denote by $\{0,1\}$ the set of strategies.

In this setting, we show that we can find a pure Nash equilibrium in $O(n^2\Delta)$ time by reducing it to a discrete preference game on a path metric, where n is the number of players, and Δ is the maximum degree of a given graph.

Theorem 9.15. *Suppose that a two-strategic network coordination game $\mathscr{G} = (G = (V,E), (\{0,1\})_{v\in V}, (C_e)_{e\in E})$ satisfies that for each edge $\{i,j\} \in E$, a cost function $C_{i,j}$ is a symmetric submodular function. In this setting, we can find a pure Nash equilibrium for $\mathscr{G}$ in $O(n^2\Delta)$ time, where n is the number of players, and Δ is the maximum degree of G.*

Proof. Let $\mathscr{G} = (G = (V,E), (\{0,1\})_{v\in V}, (C_e)_{e\in E})$ be a network coordination game. We now reduce this game to a discrete preference game on the path metric $\mathscr{M} = (\{0,1\}, d)$, where $d(x,y) = 1/2$ if $x \neq y$, otherwise $d(x,y) = 0$. Furthermore, our reduction preserves the construction of the players' network, and hence, the resulting discrete preference game is on the graph $G = (V,E)$.

For each edge $\{i,j\} \in E$, the weight is $w_{i,j} = 2C_{i,j}(1,0) - C_{i,j}(0,0) - C_{i,j}(1,1)$. For each player $i \in V$, the penalty is $p_i(s) = \sum_{j\in N(i)} C_{i,j}(1-s, 1-s)$ for each $s \in \{0,1\}$. Note that every weight $w_{i,j}$ on an edge $\{i,j\} \in E$ is non-negative from our restrictions.

We denote by $\mathscr{G}' = (G' = (V,E,(w_e)_{e\in E}),\mathscr{M},(p_v(0),p_v(1))_{v\in V})$ the resulting discrete preference game with penalties. From Lemma 9.7, the exact potential function Φ for $\mathscr{G}'$ is

$$\Phi(x) = \sum_{i\in V}\sum_{s\in\{0,1\}} p_i(s)d(s,x_i) + \sum_{\{i,j\}\in E} w_{i,j}d(x_i,x_i). \tag{9.9}$$

We show, in Lemma 9.16, that Ψ equals to the exact potential function defined in Eq. (9.7) for the given network coordination game $\mathscr{G}$. Proving this, we complete the reduction from $\mathscr{G}$ to a discrete preference game on a path metric.

Lemma 9.16. *The above function Φ is an exact potential function for $\mathscr{G}$.*

Proof. It suffices to show that Φ defined in Eq. (9.9) equals the function defined in Eq. (9.7). By definition, it follows that

$$\begin{aligned}
\Phi(x) &= \sum_{i\in V}\sum_{k=0,1} p_i(k)d(k,x_i) + \sum_{\{i,j\}\in E} w_{i,j}d(x_i,x_j)\\
&= \sum_{i\in V}\sum_{j\in N(i)} \big(C_{i,j}(1,1)d(0,x_i) + C_{i,j}(0,0)d(1,x_i)\big)\\
&+ \frac{1}{2}\sum_{i\in V}\sum_{j\in N(i)} \big(2C_{i,j}(0,1) - C_{i,j}(1,1) - C_{i,j}(0,0)\big)d(x_i,x_j)\\
&= \frac{1}{2}\sum_{i\in V}\sum_{j\in N(i)} \big(C_{i,j}(1,1)d(0,x_i) + C_{i,j}(0,0)d(1,x_i)\\
&+ C_{i,j}(1,1)d(0,x_j) + C_{i,j}(0,0)d(1,x_j)\big)\\
&+ \frac{1}{2}\sum_{i\in V}\sum_{j\in N(i)} \big(2C_{i,j}(0,1) - C_{i,j}(1,1) - C_{i,j}(0,0)\big)d(x_i,x_j)\\
&= \frac{1}{2}\sum_{i\in V}\sum_{j\in N(i)} \big(C_{i,j}(1,1)d(0,x_i) + C_{i,j}(0,0)d(1,x_i)\\
&+ C_{i,j}(1,1)d(0,x_j) + C_{i,j}(0,0)d(1,x_j)\\
&+ 2C_{i,j}(0,1)d(x_i,x_j) - C_{i,j}(1,1)d(x_i,x_j)\\
&- C_{i,j}(0,0)d(x_i,x_j)\big)\\
&= \frac{1}{2}\sum_{i\in V}\sum_{j\in N(i)} \Big(C_{i,j}(0,0)\big(d(1,x_i) + d(1,x_j) - d(x_i,x_j)\big)\\
&\qquad + C_{i,j}(1,1)\big(d(0,x_i) + d(0,x_j) - d(x_i,x_j)\big)\\
&\qquad + 2C_{i,j}(0,1)d(x_i,x_j)\Big)\\
&= \frac{1}{2}\sum_{i\in V}\sum_{j\in N(i)} C_{i,j}(x_i,x_j).
\end{aligned}$$

Note that, in the third equality, for each player $i \in V$, we add the additional value $C_{i,j}(1,1)d(0,x_j) + C_{i,j}(0,0)d(1,x_j)$ for every neighbor $j \in N(i)$. That value also

appears in the terms of i in the third equation. Dividing the new summation

$$\sum_{i\in V}\sum_{j\in N(i)}\Big(C_{i,j}(1,1)d(0,x_i)+C_{i,j}(0,0)d(1,x_i)+C_{i,j}(1,1)d(0,x_j)+C_{i,j}(0,0)d(1,x_j)\Big)$$

into half, it is equivalent to the the first summation of the third equation. Therefore, the fourth equality holds. The final equality follows from the following fact: for each $s\in\{0,1\}$ and each pair of $i,j\in V$,

$$d(s,x_i)+d(s,x_j)-d(x_i,x_j)=\begin{cases}1 & \text{if } x_i=x_j=1-s\\ 0 & \text{otherwise.}\end{cases}$$

Furthermore, we have

$$\frac{1}{2}\sum_{i\in V}\sum_{j\in N(i)}C_{i,j}(x_i,x_j)=\sum_{\{i,j\}\in E}C_{i,j}(x_i,x_j)$$

which is the exact potential function for the network coordination game $\mathscr{G}$, defined in Eq (9.7). Thus, we complete the proof of Lemma 9.16. □

Since any two-element finite metric space is a tree metric, we can apply *Tree Metric Algo*, proposed by Lolakapuri, Bhaskar, Narayanam, Parija, and Dayama [Lol+19], to find a pure Nash equilibrium for $\mathscr{G}'$. Here, *Tree Metric Algo* is an algorithm for computing a pure Nash equilibrium for a discrete preference game on a tree metric.

In a two-strategic setting, each player moves her strategy at most once during *Tree Metric Algo*. Recall that Lolakapuri, Bhaskar, Narayanam, Parija, and Dayama [Lol+19] showed the following theorem:

Theorem 9.17 (Lolakapuri, Bhaskar, Narayanam, Parija, and Dayama [Lol+19])**.** *For a discrete preference game that has n players and whose tree metric space has m points, the Tree Metric Algo outputs a pure Nash equilibrium on the given game, in $O(nm\cdot n\mathrm{EO})$-time. Here* EO *is the time to evaluate the cost function for a player.*

Since the metric space has only two points and the cost function for each player can be evaluated in $O(\Delta)$-time, we can compute a pure Nash equilibrium for a $\mathscr{G}'$ in $O(n^2\Delta)$ time.

Since each pure Nash equilibrium for $\mathscr{G}'$ agrees with a pure Nash equilibrium for the network coordination game $\mathscr{G}$, we obtain a pure Nash equilibrium for $\mathscr{G}$ from the above argument. We complete the proof of Theorem 9.15. □

Remark 9.18. If every cost function on a network coordination game is a submodular function, then the exact potential function Φ' defined in Eq. (9.7) is also a submodular function. This implies that we can apply an algorithm for submodular function minimization, such as [LSW15; Orl09], to find a pure Nash equilibrium. In particular, we solve it in $O(n^3\log^2(n)\cdot\mathrm{EO}+n^4\log^{O(1)}(n))$ time [LSW15], where n is the number of players, and EO is the time to evaluate Φ', which is bounded by the number of edges.

We obtain a pure Nash equilibrium that minimizes the corresponding potential function by using the submodular minimization algorithm. Note that equilibrium

computation allows any pure Nash equilibrium as a solution; that is, a solution that we obtain does not necessarily minimize the corresponding function. In Theorem 9.15, we exploit this fact, and thus, we can find a pure Nash equilibrium faster than applying an algorithm for submodular function minimization. Our result implies that equilibrium computation is solved at least $O(n)$ factor faster.

9.6 Conclusion

We have studied the complexity of computing a pure Nash equilibrium for a discrete preference game on a grid graph. As mentioned in Section 9.1, our motive behind this work is to resolve the main open question for network coordination games: Is it tractable to find a pure Nash equilibrium for a network coordination game on a graph with degree four? Under negative conjecture, we study the complexity of computing a pure Nash equilibrium for a discrete preference game, a subclass of network coordination games.

Unfortunately, it is still open whether finding a pure Nash equilibrium for a discrete preference game on a graph with degree four is tractable. We have shown the polynomial-time computability for a discrete preference game with a parameter on a k-dimensional grid graph when it satisfies the two conditions (A) and (B). It is the first result for efficient computability for a discrete preference game with neither $O(\log n)$-treewidth nor a tree metric space. Note that our result holds under somewhat artificial conditions. An interesting open question worth considering is whether it is also tractable if we remove the condition (A) or (B).

Another interesting direction would be the complexity of computing pure Nash equilibria for discrete preference games on the discrete metric space with three or more elements. We provide, in Section 9.3, an upper bound for the number of iterations of the best response dynamics for a discrete preference game with a parameter on a discrete metric space. The discrete preference metric space with three or more strategies is one of the simple environments among finite metric spaces, not a tree metric space.

Finally, we have discussed the complexity of computing a two-strategic network coordination game whose cost functions are symmetric submodular functions. In this case, the game is reducible to a discrete preference game on a path metric space, and we can find a pure Nash equilibrium faster than an algorithm for submodular function minimization. An open question worth considering is whether we can also compute a pure Nash equilibrium faster than an algorithm for submodular function minimization when a cost function is asymmetric.

Chapter 10

On the Complexity of Stable Fractional Hypergraph Matching

10.1 Basics

Gale and Shapley [GS13] introduced the stable matching model, which is one of the most important mathematical models for matching problems. The stable matching model is usually defined on undirected graphs. Thus, this model is naturally generalized to hypergraphs. We can easily see that there is an instance of the stable matching problem on hypergraphs that has no stable hypergraph matching. So, we consider a fractional matching, which is a relaxation concept in this chapter. Recall that we assign the value 0 or 1 to each edge in the original stable matching problem. In a fractional matching, we assign a real number between 0 and 1 to each edge. Fortunately, Aharoni and Fleiner [AF03] have proven that there exists a stable fractional matching in every hypergraph. Their proof was based on Scarf's Lemma [Sca67]. For example, the concept of stable fractional matchings in a hypergraph is used in [BF16; BFI16; NV15]. It should be noted that stable fractional matchings in hypergraphs are closely related to the stable matching problem with couple [BK13] that is a practical and theoretically important variant of the stable matching (see, e.g., [BFI16; NV15]). In this chapter, we consider the problem of computing a stable fractional matching in a hypergraph matching.

Kintali, Poplawski, Rajaraman, Sundram, and Teng [Kin+13] have proven that the problem of finding a stable fractional matching in a hypergraphic preference system is PPAD-complete. We consider the complexity of the problem of finding a stable fractional matching in a hypergraphic preference system whose maximum degree is bounded by some constant. It is natural to consider that in many practical applications, the length of a preference list (i.e., the degree of a vertex) is constant. Thus, it is important to reveal the complexity of this problem with a low constant degree. The proof by Kintali, Poplawski, Rajaraman, Sundraman, and Teng [Kin+13] implies the PPAD-completeness of the problem of finding a stable fractional matching in a hypergraphic preference system whose maximum degree is five. However, to the best our knowledge, the complexity of the problem of finding a stable fractional matching in a hypergraphic preference system whose maximum degree is at most four is open. In this chapter, we prove that

1. this problem is PPAD-complete even if the maximum degree three;
2. if the maximum degree is two, then this problem can be solved in polynomial time;

3. it is PPAD-complete to compute an approximate stable fractional matching in a hypergraphic preference system.

10.2 Problem Formulation and Main Results

A *hypergraphic preference sytem* P consists of the following two components:

1. a finite hypegraph (V,E) and
2. a set of strict total orders $\succ_v$ for every vertex v in V such that for each vertex $v \in V$, $\succ_v$ is a strict total order on $E(v)$,

where for each edge $e \in E$, we denote by $E(v)$ the set of hyperedges $e \in E$ such that $v \in e$. We denote by $P = (V,E,\{\succ_v\}_{v\in V})$ this hypergraphic preference system P. Notice if $|e| = 2$ for every hyperedge e in E, then P is just an instance of the well-known stable roommate problem (see, e.g., [Pap07]). Define $\deg(P) := \max_{v\in V} |E(v)|$.

Assume that we are given a hypergraphic preference system $P = (V,E,\{\succ_v\})$. Then a vector x in $\mathbb{R}^E_{\geq 0}$ is called a *fractional matching* in P if $\sum_{e\in E(v)} x(e) \leq 1$ for every vertex v in V. Furthermore, a fractional matching $x \in \mathbb{R}^E_{\geq 0}$ is said to be *stable* if for every hyperedge e in E, there exists a vertex $v \in e$ such that

$$x(e) + \sum_{f\in E(v): f\succ_v e} x(f) = 1. \tag{10.1}$$

It is known [AF03] that there exists a stable fractional matching in every hypergraphic preference system. The problem called FRACTIONAL STABLE MATCHING is defined as follows:

Definition 10.1. FRACTIONAL STABLE MATCHING
Input:

- a hypergraphic preference system $P = (V,E,\{\succ_v\})$.

Task: Find

- a stable fractional matching $x \in \mathbb{R}^E_{\geq 0}$ in P, i.e., x satisfies that
  - $\sum_{e\in E(v)} x(e) \leq 1$ for every vertex $v \in V$;
  - for each hyperedge $e \in E$, there exists a vertex $v \in e$ such that Equation 10.1 holds.

The following result about the computational complexity of FRACTIONAL STABLE MATCHING is known.

Theorem 10.2 (Kintali, Poplawski, Rajaraman, Sundaraman, and Teng [Kin+13])**.** FRACTIONAL STABLE MATCHING *is* PPAD-*complete.*

The proof by Kintali, Poplawski, Rajaraman, Sundaraman, and Teng [Kin+13] implies the PPAD-completeness of the problem of finding a stable fractional matching in a hypergraphic preference system P such that $\deg(P) = 5$. However, to the best of our knowledge, the complexity of the problem of finding a stable fractional matching

in a hypergraphic preference system P such that $2 \leq \deg(P) \leq 4$ is open. (If $\deg(P) = 1$, then the answer of FRACTIONAL STABLE MATCHING is trivial.) This chapter presents the following theorems:

Theorem 10.3. FRACTIONAL STABLE MATCHING *in a hypergraphic preference system P such that* $\deg(P) = 3$ *is* PPAD*-complete.*

Theorem 10.4. FRACTIONAL STABLE MATCHING *in a hypergraphic preference system P such that* $\deg(P) = 2$ *can be solved in polynomial time.*

In should be note that Theorem 10.3 implies the PPAD-completeness of FRACTIONAL STABLE MATCHING in a hypergraphic preference system P such that $\deg(P) = 4$. It is sufficient to add a with degree four to the instance used in the proof Theorem 10.3.

Furthermore, we consider APPROXIMATE FRACTIONAL STABLE MATCHING that is an approximate variant of FRACTIONAL STABLE MATCHING. In this problem, we are given a hypergraphic preference system $P = (V, E, \{\succ_v\})$ and a positive real number ε that may depend on $|V|$ and $|E|$. Then a fractional matching $x \in \mathbb{R}^E_{\geq 0}$ is said to be ε-*stable* if for every hyperedge $e \in E$, there exists a vertex $v \in e$ such that

$$x(e) + \sum_{f \in E(v): f \succ_v e} x(f) \geq 1 - \varepsilon. \tag{10.2}$$

Notice that an ε-stable fractional matching in P since a stable fractional matching in P always exists. The goal of this problem is to find an ε-stable fractional matching in P. We prove the following theorem:

Definition 10.5. APPROXIMATE FRACTIONAL STABLE MATCHING
Input:

- a hypergraphic preference system $P = (V, E, \{\succ_v\})$,
- a positive real value $\varepsilon > 0$.

Task: Find

- an ε-stable fractional matching $x \in \mathbb{R}^E_{\geq 0}$ in P, i.e., x satisfies that
  - $\sum_{e \in E(v)} x(e) \leq 1$ for every vertex $v \in V$;
  - for each hyperedge $e \in E$, there exists a vertex $v \in e$ such that Equation 10.2 holds.

Theorem 10.6. APPROXIMATE FRACTIONAL STABLE MATCHING *is* PPAD*-complete.*

10.3 PPAD-completeness

For proving Theorem 10.3, we need the following lemma.

Lemma 10.7. *Assume that we are given a hypergraphic preference system P such that* $\deg(P) \geq 4$. *Then there exists a hypergraphic preference system Q such that (i)* $\deg(Q) = 3$ *and (ii) we can construct a stable fractional matching in P from a stable fractional matching in Q in polynomial time. Furthermore, we can construct Q in polynomial time.*

Before proving Lemma 10.7, we prove Theorem 10.3 by using this lemma.

Proof of Theorem 10.3. It follows from Theorem 10.2 that FRACTIONAL STABLE MATCHING in a hypergraphic preference system P such that $\deg(P) = 3$ belongs to PPAD. Furthermore, Theorem 10.2 and Lemma 10.7 imply that every problem in PPAD is reducible to FRACTIONAL STABLE MATCHING in a hypergraphic preference system P such that $\deg(P) = 3$ in polynomial time. This complete the proof. □

10.3.1 Proof of Lemma 10.7

In this subsection we prove Lemma 10.7. The following proof is inspired by the proof of the PPAD-completeness of PREFERENCE GAME with degree three by Kintali, Poplawski, Rajaraman, Sundaram, and Teng [Kin+13].

Assume that we are given a hypergraphic preference system $P = (V, E, \{\succ_v\})$ such that $\deg(P) \geq 4$. Then we construct a new hypergraphic preference system $Q = (W, F, \{\rhd_v\})$ as follows. Define

$$W := \{v_i \; ; \; v \in V, i \in \{1, 2, \ldots, |E(v)|\} \cup \{\bar{v}_i \; ; \; v \in V, i \in \{1, 2, \ldots, |E(v)| - 1\}.$$

For each vertex $v \in V$ and each hyperedge $e \in E(v)$, we define

$$r(v, e) := 1 + |\{f \in E(v) \; ; \; f \succ_v e\}|.$$

For each hyperedge $e \in E$, we define $\bar{e} := \{v_{r(v,e)} \; ; \; v \in e\}$. Define $\bar{E} := \{\bar{e} \; ; \; e \in E\}$ and

$$F := \bar{E} \cup \{\{v_i, \bar{v}_i\}, \{\bar{v}_i, v_{i+1}\} \; ; \; v \in V, i\{1, 2, \ldots, |E(v)| - 1\}\}.$$

For each vertex $v \in V$ and each integer $i \in \{1, 2, \ldots, |E(v)|\}$, we denote by h_i^v the hyperedge $e \in \bar{E}$ such that $v_i \in e$. For each vertex $w \in W$, we define the strict total order $\rhd_w$ as follows. We first consider the case where $w = v_i$ for some vertex $v \in V$ and some integer $i \in \{1, 2, \ldots, |E(v)|\}$. It suffices to consider the case where $|E(v)| \geq 2$. In this case, we define

$$\begin{cases} h_1^v \rhd_w \{v_1, \bar{v}_1\} & \text{if } i = 1 \\ \{\bar{v}_{|E(v)|-1}, v_{|E(v)|}\} \rhd_w h_{|E(v)|}^v & \text{if } i = |E(v)| \\ \{\bar{v}_{i-1}, v_i\} \rhd_w h_i^v \rhd_w \{v_i, \bar{v}_i\} & \text{otherwise.} \end{cases}$$

Next, we assume that $w = \bar{v}_i$ for some vertex $v \in V$ and some integer $i \in \{1, 2, \ldots, |E(v)| - 1\}$. In this case, we define $\{v_i, \bar{v}\} \rhd_w \{\bar{v}_i, v + i + 1\}$. Since $|W| \leq 2|V||E|$ and $|F| \leq |E| + 2|V||E|$, Q can be constructed in polynomial time. Futhermore, $\deg(Q) = 3$.

In what follows, we prove that we can construct a stable fractional matching in P from a stable fractional matching in Q in polynomial time. Assume that we are given a stable fractional matching z in Q. Then, we define the vector $x \in \mathbb{R}_{\geq 0}^E$ by $x(e) := z(\bar{e})$. Clearly, we can construct x from z in polynomial time. What remains is to prove that x is a stable fractional matching in P. To prove this, we need the following lemma:

Lemma 10.8. *For every vertex $v \in V$ and every integer $i \in \{1, 2, \ldots, |E(v)| - 1\}$,*

(10.8.A) $z(\{v_i, \bar{v}_i\}) = 1 - \sum_{j=1}^{i} z(h_j^v)$, *and*

(10.8.B) $z(\{\bar{v}_i, v_{i+1}\}) = \sum_{j=1}^{i} z(h_j^v)$.

Proof. Let v be a vertex in V such that $|E(v)| \geq 2$. We prove by induction on i. We first consider the case of $i = 1$. Since z is a fractional matching in Q, we have

$$1 \geq \sum_{e \in F(v_1)} z(e) = z(h_1^v) + z(\{v_1, \bar{v}_1\}).$$

This implies that $z(\{v_1, \bar{v}_1\}) < 1 - z(h_1^v)$. For proving (10.8.A), we assume that $z(\{v_1, \bar{v}_1\}) < 1 - z(h_1^v)$. Since z is a stable fractional matching in Q, at least one of the following statement holds:

$$1 = z(\{v_1, \bar{v}_1\}) + \sum_{e \in F(v_1): e \rhd_{v_1} \{v_1, \bar{v}_1\}} z(e) = z(\{v_1, \bar{v}_1\}) + z(h_1^v); \tag{10.3}$$

$$1 = z(\{v_1, \bar{v}_1\}) + \sum_{e \in F(\bar{v}_1): e \rhd_{\bar{v}_1} \{v_1, \bar{v}_1\}} z(e) = z(\{v_1, \bar{v}_1\}). \tag{10.4}$$

However, the above assumption implies that $z(\{v_1, \bar{v}_1\}) + z(h_1^v) < 1$ and $z(\{v_1, \bar{v}_1\}) < 1$ since $z(h_1^v) \geq 0$. These observations contradict (10.3) and (10.4). Therefore, $z(\{v_1, \bar{v}_1\}) = 1 - z(h_1^v)$.

Next, we consider (10.8.B). Since z is a fractional matching in Q, we have

$$1 \geq \sum_{e \in F(\bar{v}_1)} z(e) = z(\{v_1, \bar{v}_1\}) + z(\{\bar{v}_1, v_2\}).$$

We have $z(\{\bar{v}_1, v_2\}) \leq z(h_1^v)$ since (10.8.A) for the case of $i = 1$ implies that $z(\{v_1, \bar{v}_1\}) = 1 - z(h_1^v)$. For proving (10.8.B) by contradiction, we assume that $z(\{\bar{v}_1, v_2\}) < z(h_1^v)$. Since z is a stable fractional matching in Q, at least one of the following statements holds:

$$1 = z(\{v_k, \bar{v}_k\}) + \sum_{e \in F(v_k): e \rhd_{v_k} \{v_k, \bar{v}_k\}} z(e) = z(\{v_k, \bar{v}_k\}) + z(\{\bar{v}_{k-1}, v_k\}) + z(h_k^v); \tag{10.5}$$

$$1 = z(\{v_k, \bar{v}_k\}) + \sum_{e \in F(\bar{v}_k): e \rhd_{\bar{v}_k} \{v_k, \bar{v}_k\}} z(e) = z(\{v_k, \bar{v}_k). \tag{10.6}$$

However, the above assumption and the induction hypothesis imply that

$$\begin{aligned} z(\{v_k, \bar{v}_k\}) + z(\{\bar{v}_{k-1}, v_k) + z(h_k^v) &= z(\{z_k, \bar{v}_k\}) + \sum_{j=1}^{k-1} z(h_j^v) + z(h_k^v) \\ &< 1 - \sum_{j=1}^{k} z(h_j^v) + \sum_{j=1}^{k} z(h_j^v) = 1. \end{aligned}$$

This contradicts (10.5). Furthermore, it follows that $z(\{v_k, \bar{v}_k)\} < 1$ from the above assumption since z is a non-negative real vector. This contradicts (10.6). This complete the proof of (10.8.A).

Next, we consider (10.8.B). Since z is a fractional matching in Q, we have

$$1 \geq \sum_{e \in F(\bar{v}_k)} z(e) = z(\{v_k, \bar{v}_k\}) + z(\{\bar{v}_k, v_{k+1}\}).$$

Since (10.8.A) for the case of $i = k$ implies that

$$z(\{v_k, \bar{v}_k\}) = 1 - \sum_{j=1}^{k} z(h_j^v), \tag{10.7}$$

we have

$$z(\{\bar{v}_k, v_{k+1}\}) \leq \sum_{j=1}^{k} z(h_j^k). \tag{10.8}$$

For proving (10.8.B) by contradiction, we assume that the inequality in 10.8 strictly holds. Since z is a stable fractional matching in Q, at least one of the following statements holds:

$$1 = z(\{\bar{v}_k, v_{k+1}\}) + \sum_{e \in F(\bar{v}_k): e \rhd_{\bar{v}_k} \{\bar{v}_k, v_{k+1}\}} z(e) = z(\{\bar{v}_k, v_{k+1}\}) + z(\{v_k, \bar{v}_k\}). \tag{10.9}$$

$$1 = z(\{\bar{v}_k, v_{k+1}\}) + \sum_{e \in F(v_{k+1}): e \rhd_{v_{k+1}} \{\bar{v}_k, v_{k+1}\}} z(e) = z(\{\bar{v}_k, v_{k+1}\}). \tag{10.10}$$

Notice that 10.7 and the above assumption imply that

$$z(\{\bar{v}_k, v_{k+1}\}) + z(\{v_k, \bar{v}_k\}) < \sum_{j \in [k]} z(h_j^v) + 1 - \sum_{j \in [k]} z(h_j^v) = 1.$$

This contradicts 10.9. Furthermore, 10.7 and $z \in \mathbb{R}^F_{\geq 0}$ imply that $\sum_{j \in [k]} z(h_j^v) \leq 1$. This contradicts 10.10, and complete the proof. □

We are now ready to prove that x is a stable fractional matching in P. We first prove that x is a fractional matching in P. Let v be a vertex in V. Define $k := |E(v)|$. If $k = 1$, then

$$\sum_{e \in E(v)} x(e) = z(h_1^v) \leq 1.$$

If $k > 1$, then

$$\begin{aligned}
\sum_{e \in E(v)} x(e) &= \sum_{i \in [k]} z(h_i^v) \\
&= \sum_{i \in [k-1]} z(h_i^v) + z(h_k^v) \\
&= z(\{\bar{v}_{k-1}, v_k\}) + z(h_k^v) && \text{(by (10.8.B) of Lemma 10.8)} \\
&= \sum_{e \in F(v_k)} z(e) \leq 1,
\end{aligned}$$

where the inequality follows from the fact that z is a fractional matching in Q.

Lastly, we prove that x is a stable fractional matching in P. Let e be a hyperedge in E. Then since z is a stable fractional matching in Q, there exists a vertex w in $\bar{e}$ such that

$$z(\bar{e}) + \sum_{f \in F(w): f \rhd_w \bar{e}} z(f) = 1.$$

Assume that $w = v_k$ for some vertex v in e and some integer $k \in [|E(v)|]$. Notice that $\bar{e} = h_k^v$. For each integer $i \in [k]$, we assume that $h_i^v = \bar{e}_i$. Notice that $e_k = e$, $e_1 \succ_v e_2 \succ_v \cdots \succ_v e_k$, and $e \succ_v f$ holds for every hyperedge $f \in E(v) \setminus \{e_1, e_2, \ldots, e_k\}$. For integer $i \in [k]$, $x(e_i) = z(h_i^v)$. If $k = 1$, then

$$1 = z(\bar{e}) + \sum_{f \in F(w): f \rhd_w \bar{e}} z(f) = z(\bar{e}) = x(e) = x(e) + \sum_{f \in E(v): f \succ_v e} x(f).$$

If $k > 1$, then

$$\begin{aligned}
1 &= z(\bar{e}) + \sum_{f \in F(w): f \rhd_w \bar{e}} z(f) \\
&= z(h_k^v) + z(\{\bar{v}_{k-1}, v_k\}) \\
&= z(h_k^v) + \sum_{i \in [k-1]} z(h_i^v) && \text{(by (10.8.B) of Lemma 10.8)} \\
&= x(e) + \sum_{i \in [k-1]} x(e_i) \\
&= x(e) + \sum_{f \in E(v): f \succ_v e} x(f).
\end{aligned}$$

These imply that x is a stable fractional matching in P. This completes the proof.

10.4 Polynomial-Time Computability

Throughout this section, we assume that we are given a hypergraphic preference system P such that $\deg(P) = 2$. Define V^* as the set of vertices $v \in V$ such that $|E(v)| = 2$. In addition, we define the directed graph $D = (N, A)$ as follows. For each hyperedge $e \in E$, N contains a vertex n_e. For each vertex $v \in V^*$, A contains an arc from n_f to n_e, where we assume that distinct hyperedges e, f in E contain v and $e \succ_v f$. See Figure 10.1 for an example of D.

Our algorithm is described in Algorithm 6. This algorithm can be intuitively explained as follows. If there exists a vertex n_e in N such that any arc in A does not leave n_e, then the hyperedge e is most preferred by every vertex v in e. Thus, we set the value for e to be 1. For every arc $a = (n_f, n_e)$ in A, we must set the value for f to be 0 since some vertex in V is contained in e, f. Then we can remove vertices in N whose value is determined from D. We repeat this. Finally, we obtain a directed graph D' in which the out-degree of every vertex is at least one. Thus, by setting the value for each vertex of D' to be $1/2$, we can construct a stable fractional matching in P.

Here, we apply Algorithm 6 for the example in Figure 10.1. Since the right digraph has no sinks, $x(e) = 1/2$ for every hyperedges $e \in E$.

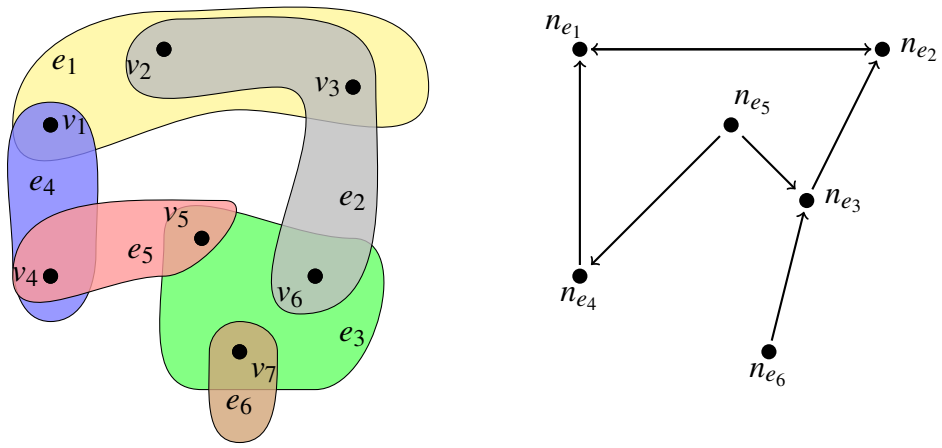

FIGURE 10.1: The left figure illustrates an hypergraphic preference system such that $e_1 \succ_{v_1} e_2 \succ_{v_2} e_1$, $e_1 \succ_{v_3} e_2$, $w_4 \succ_{v_4} e_5$, $e_3 \succ_{v_5} e_5$, $e_2 \succ_{v_6} e_3$, and $e_3 \succ_{v_7} e_6$. The right figure represents the digraph constructed from the left one.

In order to prove Theorem 10.4, we need to show that Algorithm 6 always halts and computes a stable fractional matching in the given hypergraphic preference system and computes a stable fractional matching in the given hypergraphic preference system in polynomial time. We prove, in Lemma 10.9, that Algorithm 6 computes a stable fractional matching in the given hypergraphic preference system. After that, we show that this algorithm halts in polynomial time in Lemma 10.9.

Lemma 10.9. *The output of Algorithm 6 is a stable fractional matching in P.*

Proof. Assume that Algorithm 6 halts when $t = k$. For proving this lemma, it suffices to prove the following conditions are satisfied.

(10.9.A) For every arc $a = (u, v) \in A$, we have $\xi^*(u) + \xi^*(v) \leq 1$.

(10.9.B) For every vertex $vinN$ such that $\xi^*(v) \neq 1$, there exists a vertex $w \in N$ such that an arc from v to w is contained in A and $\xi^*(v) + \xi^w = 1$.

We first prove (10.9.A). Assume that we are given an arc $a = (u, v)$ in A. If $\xi^*(u) = 0$, then (10.9.A) clearly holds. Next, we assume that $\xi^*(u) = 1$. Then there exists a positive integer t such that $u \in N_t$ and any arc of D_t does not leave u. Notice that $v \in N_t$. This implies that $\xi^*(v) \in \{0, 1\}$. If $\xi^*(v) = 1$, then there exists a positive integer t' such that $t' < t$, $v \in N_{t'}$, and any arc of $D_{t'}$ does leave v. Furthermore, the definition of $T_{t'}$ implies that $\xi^*(v) = 0$. Lastly, we consider the case where $\xi^*(u) = 1/2$, i.e., $u \in N_t$. If $\xi^*(v) = 1$, then $u \notin N_k$, which contradicts the fact that $u \in N_k$. This implies that $\xi^* \in \{0, 1/2\}$. This completes the proof of (10.9.A).

Next, we prove (10.9.B). Assume that we are given a vertex $v \in N$ such that $\xi^*(v) \neq 1$. Assume taht $\xi^*(v) = 0$. In this case, there exists a positive integer t such that $v \in T_t$. That is, there exists a vertex $w \in S_t$ such that there exists an arc of D_t from v to w. Since $w \in S_t$, $\xi^*(w) = 1$. This implies that $\xi^*(v) + \xi^*(w) = 1$. Next, we assume that $\xi^*(v) = 1/2$. In this case, there exists a vertex $w \in N_k$ such that there exists an arc of D_k from v to w. Since $\xi^*(w) = 1/2$, we have $\xi^*(v) + \xi^*(w) = 1$. This completes the proof. □

Algorithm 6 Finding a fractional stable matching

Set value $w_v^1 = -1$ for every $v \in N$.
Define $N_1 := \{v \in N \; ; \; w_v < 0\}$ and $D_1 := D$.
Set $k := 1$.
while there exists a vertex $v \in N_k$ such that $w + v^k < 0$ **do**
    Define S_k as the set of vertices $v \in N_k$ whose out-degree is 0.
    Define $T_k := \{v \in N_k \; ; \; (v,s) \in A$ for some $s \in S_k\}$.
    if S_k is empty **then**
        Define $w_v^k = 1/2$ for each $v \in N_k$
    else
        Define the value $w_s^k = 1$ for each $s \in S_k$.
        Define the value $w_t^k = 0$ for each $t \in T_k$.
    end if
    $N_{k+1} := N_k \setminus (S_k \cup T_k)$.
    D_{k+1} as teh subgraph of D_k induced by N_{k+1}.
    $k := k+1$.
end while
Define the vector $x \in \mathbb{R}^N_{\geq 0}$ by $x(e) := w_e^k$ for each hyperedge $e \in E$.
Return x, and halt.

Proof of Theorem 10.4. This theorem immediately follows from Lemma 10.9. □

10.5 Approximate

This section presents the proof of Theorem 10.6. Since a stable fractional matching is clearly an ε-stable fractional matching for any positive rational number ε, Theorem 10.2 (i.e., the fact that FRACTIONAL STABLE MATCHING is in PPAD) implies that APPROXIMATE FRACTIONAL STABLE MATCHING is in PPAD. What remains is to prove that every problem in PPAD is reducible to APPROXIMATE FRACTIONAL STABLE MATCHING. For this, Theorem 10.2 implies that it is sufficient to prove that FRACTIONAL STABLE MATCHING is reducible to APPROXIMATE FRACTIONAL STABLE MATCHING in polynomial time. This fact immediately follows from the following lemma.

Lemma 10.10. *Assume that are given a hypergraphic preference system $P = (V, E, \{\succ_v\})$. Furthermore, we define $\varepsilon := 2^{-20|E|^4}$. Then we can construct a stable fractional matching in P from an ε-stable fractional matching in P in polynomial time.*

Notice that the bit-length of ε in Lemma 10.10 is bounded by a polynomial in the size of P. More precisely, the bit-length of ε in Lemma is $O(|E|^4)$.

What remains is to prove Lemma 10.10. We prove Lemma 10.10 by using the following known result called LP compactness. Assume that we are given positive integer m, n and vectors $a \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$, where $\mathbb{Q}$ is the set of rational numbers. Then we consider the following linear inequality system whose variable is a vector

$x \in \mathbb{R}^n$.

$$\sum_{j \in [n]} a(i,j) \cdot x(j) \geq b(i) (i \in [m]). \tag{10.11}$$

For each positive real number δ and each vector $y \in \mathbb{R}_{\geq 0}{}^n$, we say that y satisfies the linear inequality system 10.11 to within δ, if

$$\sum_{j \in [n]} a(i,j) \cdot y(j) \geq b(i) - \delta$$

for every integer $i \in [m]$.

Theorem 10.11 (LP compactness (see Lemma 4.11 of [Kin+13])). *Assume that we are given positive integers m, n and vectors $a \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$. Furthermore, we assume that there exists a positive integer β satisfying the condition that for every pair of integer $i \in [m]$ and $j \in [n]$, there exist integers p, q, r, s such that $a(i,j) = p/q, b(i) = r/s$, and $|p|, |q|, |r|, |s| \leq 2^\beta$. Then we consider the following linear inequality system whose variable is a vector $x \in \mathbb{R}^n$:*

$$\sum_{j \in [n]} a(i,j) \cdot x(j) \geq b(i) (i \in [m]). \tag{10.12}$$

If there exists a vector $y \in \mathbb{R}^n$ satisfying the linear inequality system (10.12) to within $2^{-20n^4\beta}$, then there exists a vector $x \in \mathbb{R}^n$ that is feasible for the linear inequality system (10.12).

We are now ready to prove Lemma 10.10.

Proof of Lemma 10.10. Assume that we are given an ε-stable fractional matching $y \in P$. For each hyperedge $e \in E$, we define set $U(e)$ as the set of vertices $v \in e$ such that

$$y(e) + \sum_{f \in E(v) : f \succ_v e} y(f) \geq 1 - \varepsilon.$$

Notice that since y in an ε-stable fractional matching in P, $U(e) \neq \emptyset$ for any hyperedge $e \in E$. We consider the following linear inequality system whose variable is a vector $x \in \mathbb{R}^E$.

$$\begin{cases} -\sum_{e \in E(v)} x(e) \geq -1 & \forall v \in V \\ x(e) + \sum_{f \in E(v) : f \succ_v e} x(f) \geq 1 & \forall e \in E, \forall v \in U(e) \\ x(e) \geq 0 & \forall e \in E. \end{cases} \tag{10.13}$$

Notice that the number of constraints of the linear inequality system (10.13) is bounded by a polynomial in the input size of FRACTIONAL STABLE MATCHING.

Notice that y satisfies the linear inequality system (10.13) to within $2^{-20|E|^4}$. Thus, by setting $n := |E|$ and $\beta := 1$,Theorem 10.11 implies that there exists a vector $x \in \mathbb{R}^E$ that is feasible for the linear inequality system (10.13) in polynomial time by using the ellipsoid method [Kha80].

Let x be a vector in $\mathbb{R}^E$ that is feasible for the linear inequality system (10.13). Then we prove that x is a stable fractional matching in P. For this, it suffices to prove

that for every hyperedge $e \in E$, there exists a vertex $v \in e$ such that

$$x(e) + \sum_{f \in E(v): f \succ_v e} x(f) = 1. \tag{10.14}$$

Let e be a hyperedge in E. The first constraint of (10.13) implies that

$$x(e) + \sum_{f \in E(v): f \succ_v e} x(f)$$

for every vertex $v \in U(e)$. Thus, the second constraint of (10.13) implies that

$$x(e) + \sum_{f \in E(v): f \succ_v e} x(f) = 1$$

for every vertex $v \in U(e)$. Since $U(e) \neq \emptyset$, this implies that there exists a vertex $v \in e$ satisfying (10.14). This completes the proof. □

10.6 Conclusions

We have considered the complexity of the problem of finding a stable fractional hypergraph matching in a hypergraphic preference system whose maximum degree is bounded by some constant. We have shown the PPAD-completeness of DEGREE-3 FRACTIONAL STABLE MATCHING and the existence of a polynomial-time algorithm for solving DEGREE-2 FRACTIONAL STABLE MATCHING. Our results have improved the result by Previously, Kintali, Poplawski, Rajaraman, Sundaram, and Teng [Kin+13]. They proved the PPAD-completeness of DEGREE-5 FRACTIONAL STABLE MATCHING.

Note that Caáji [Csá21] recently has proven the PPAD-hardness of the problem of finding a stable fractional matching in a hypergraphic preference system even if every hyperedge contains at most three vertices and every vertex joins at most three hyperedges.

Part V

Conclusions and Open Problems

Chapter 11

Open Problems

In this thesis, we have focused on the complexity of TFNP classes from the perspective of Fixed Point Theory and Algorithmic Game Theory. Part II has organized central results around TFNP classes. We have shown the robustness of the definition EOPL based on the problem END OF POTENTIAL LINE. Furthermore, we have proven that DEGREE-4 POTENTIAL ODD is a PPA$\cap$PLS-complete problem and DEGREE-3 POTENTIAL ODD is an EOPL-complete problem. Part III has considered the complexity of finding a fixed point. We have focused on the two order-theoretic fixed point theorems: Caristi's fixed point theorem and Brøndsted's fixed point theorem. Finally, Part IV has dealt with the computational aspects of Game Theory. We have considered the complexity of equilibrium computation related to graphical games with pure Nash equilibria and bimatrix games. Moreover, we have considered the complexity of computing a stable fractional matching on a hypergraphic preference system.

To conclude this thesis, this chapter collects important and interesting open questions worth considering for the reader's convenience.

II Fundamental Theory of Computational Complexity

(1) EOPL $\stackrel{?}{=}$ UniqueEOPL $\stackrel{?}{=}$ FP

(2) EOPL $\stackrel{?}{=}$ PPA$\cap$PLS

(3) Can we formulate quantum analogs of TFNP classes, such as PLS, PPP, PPA, and PPAD?

III Fixed Point Theory

(1) Is the problem of finding a Tarski's fixed point EOPL-complete?

(2) Does TARSKI belong to UniqueEOPL?

(3) Can we improve the query lower bound for computing a Tarski's fixed point in the three-dimensional setting?

(4) Which is true that BRØNDESTED is CLS- or PPAD-complete?

IV Algorithmic Game Theory

(1) How hard is the problem of computing a mixed Nash equilibrium on a graphical game with pure Nash equilibria?

(2) Is the problem of finding a pure Nash equilibrium on a network coordination game whose players' network has degree four PLS-complete?

Bibliography

[Aar13] Scott Aaronson. *Quantum Computing since Democritus.* Cambridge University Press, 2013. ISBN: 978-0-521-19956-8. DOI: 10.1017/CBO9780511979309.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009. ISBN: 978-0-521-42426-4.

[ABB20] James Aisenberg, Maria Luisa Bonet, and Sam 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.

[AF03] Ron Aharoni and Tamás Fleiner. "On a lemma of Scarf." In: *Journal of Combinatorial Theory, Series B* 87.1 (2003), pp. 72–80. DOI: 10.1016/S0095-8956(02)00028-X.

[AKV05] Timothy G. Abbott, Daniel M. Kane, and Paul Valiant. "On the Complexity of Two-PlayerWin-Lose Games." In: *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 2005, pp. 113–122. DOI: 10.1109/SFCS.2005.59.

[AOV07] Louigi Addario-Berry, Neil Olver, and Adrian Vetta. "A Polynomial Time Algorithm for Finding Nash Equilibria in Planar Win-Lose Games." In: *Journal of Graph Algorithms and Applications* 11.1 (2007), pp. 309–319. DOI: 10.7155/jgaa.00147.

[Apt+17] Krzysztof R. Apt, Bart de Keijzer, Mona Rahn, Guido Schäfer, and Sunil Simon. "Coordination games on graphs". In: *International Journal of Game Thoery* 46.3 (2017), pp. 851–877. DOI: 10.1007/s00182-016-0560-8.

[Ban+19] Frank Ban, Kamal Jain, Christos H. Papadimitriou, Christos-Alexandros Psomas, and Aviad Rubinstein. "Reductions in PPP." In: *Information Processing Letters* 145 (2019), pp. 48–52. DOI: 10.1016/j.ipl.2018.12.009.

[Ban22] S. Banach. "Sur les operations dans les ensembles abstraits et leur application aux equations integrales." In: *Fundamenta Mathematicae* 3 (1922), pp. 133–181. DOI: 10.4064/fm-3-1-133-181.

[BF16] Péter Biró and Tamás Fleiner. "Fractional solutions for capacitated NTU-games, with applications to stable matchings." In: *Discrete Optimization* 22 (2016), pp. 241–254. DOI: 10.1016/j.disopt.2015.02.002.

[BFI16] Péter Biró, Tamás Fleiner, and Robert W. Irving. "Matching couples with Scarf's algorithm." In: *Annals of Mathematics and Artificial Intelligence* 77.3-4 (2016), pp. 303–316. DOI: 10.1007/s10472-015-9491-5.

[BIL08] Vincenzo Bonifaci, Ugo Di Iorio, and Luigi Laura. "The complexity of uniform Nash equilibria and related regular subgraph problems." In: *Theoretical Computer Science* 401.1-3 (2008), pp. 144–152. DOI: 10.1016/j.tcs.2008.03.036.

[BK13] Péter Biró and Flip Klijn. "Matching with couples: a Multidisciplinary Survey." In: *International Game Theory Review* 15.2 (2013). DOI: 10.1142/S0219198913400082.

[BM21] Vittorio Bilò and Marios Mavronicolas. "The Complexity of Computational Problems About Nash Equilibria in Symmetric Win-Lose Games." In: *Algorithmica* 83.2 (2021), pp. 447–530. DOI: 10.1007/s00453-020-00763-x.

[Bor16] Michaela Borzechowski. *The complexity class Polynomial Local Search (PLS) and PLS-complete problems.* 2016.

[BR21] Yakov Babichenko and Aviad Rubinstein. "Settling the complexity of Nash equilibrium in congestion games". In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC*. ACM, 2021, pp. 1426–1437. DOI: 10.1145/3406325.3451039.

[Bro11] L. E. J. Brouwer. "Über Abbildung von Mannigfaltigkeiten." In: *Mathematische Annalen* 71.1 (1911), pp. 97 –115. ISSN: 0025-5831. DOI: 10.1007/bf01456931.

[Brø74] Arne Brøndsted. "On a lemma of Bishop and Phelps." In: *Pacific Journal of Mathematics* 55.2 (1974), pp. 335–341. DOI: 10.2140/pjm.1974.55.335.

[Cai+16] Yang Cai, Ozan Candogan, Constantinos Daskalakis, and Christos Papadimitriou. "Zero-Sum Polymatrix Games: A Generalization of Mminmax." In: *Mathematics of Operations Research* 41.2 (2016), pp. 648–655. DOI: 10.1287/moor.2015.0745.

[Car76] J. Caristi. "Fixed point theorems for mapping satisfying inwardness conditions." In: *Transactions of the American Mathematical Society* 215 (1976), pp. 241 –251. DOI: 10.1090/S0002-9947-1976-0394329-4.

[CD06] Xi Chen and Xiaotie Deng. "Settling the Complexity of Two-Player Nash Equilibrium." In: *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 2006, pp. 261–272. DOI: 10.1109/FOCS.2006.69.

[CD07] Xi Chen and Xiaotie Deng. "Recent development in computational complexity characterization of Nash equilibrium." In: *Computer Science Review* 1.2 (2007), pp. 88–99. DOI: 10.1016/j.cosrev.2007.09.002.

[CD09] Xi Chen and Xiaotie Deng. "On the complexity of 2D discrete fixed point problem." In: *Theoretical Computer Science* 410.44 (2009), pp. 4448 –4456. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2009.07.052.

[CD11] Yang Cai and Constantinos Daskalakis. "On Minmax Theorems for Multiplayer Games." In: *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. 2011, pp. 217–234. DOI: 10.1137/1.9781611973082.20.

[CDO15] Xi Chen, David Durfee, and Anthi Orfanou. "On the complexity of nash equilibria in anonymous games". In: *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. 2015, pp. 381–390. DOI: 10.1145/2746539.2746571.

[CDT09] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. "Settling the Complexity of Computing Two-Player Nash Equilibria." In: *Journal of the ACM* 56.3 (2009), 14:1 –14:57. ISSN: 00045411. DOI: 10.1145/1516512.1516516.

[Che+09] Xi Chen, Decheng Dai, Ye Du, and Shang-Hua Teng. "Settling the Complexity of Arrow-Debreu Equilibria in Markets with Additively Separable Utilities." In: *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS*. IEEE Computer Society, 2009, pp. 273–282. DOI: 10.1109/FOCS.2009.29.

[CKO18] Flavio Chierichetti, Jon Kleinberg, and Sigal Oren. "On discrete preferences and coordination." In: *Journal of Computer and System Sciences* 93 (2018), pp. 11–29. DOI: 10.1016/j.jcss.2017.11.002.

[CL10] Ching-Lueh Chang and Yuh-Dauh Lyuu. "Optimal bounds on finding fixed points of contraction mappings." In: *Theoretical Computer Science* 411.16 (2010), pp. 1742 –1749. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2010.01.016.

[CL22] Xi Chen and Yuhao Li. "Improved Upper Bounds for Finding Tarski Fixed Points." In: *CoRR* abs/2202.05913 (2022). DOI: 10.48550/arXiv.2202.05913. arXiv: 2202.05913.

[Con92] Anne Condon. "The complexity of stochastic games." In: *Information and Computation* 96.2 (1992), pp. 203–224. DOI: 10.1016/0890-5401(92)90048-K.

[Coo71] Stephen A. Cook. "The Complexity of Theorem-Proving Procedures." In: *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*. ACM, 1971, pp. 151–158. DOI: 10.1145/800157.805047.

[CS05] Bruno Codenotti and Daniel Stefankovic. "On the computational complexity of Nash equilibria for (0, 1) bimatrix games." In: *Information Processing Letters* 94.3 (2005), pp. 145–150. DOI: 10.1016/j.ipl.2005.01.010.

[Csá21] Gergely Csáji. *On the complexity of Stable Hypergraph Matching, Stable Multicommodity Flow and related problems.* 2021.

[Das09] Constantinos Daskalakis. "Nash equilibria: Complexity, symmetries, and approximation." In: *Computer Science Review* 3.2 (2009), pp. 87–100. DOI: 10.1016/j.cosrev.2009.03.003.

[Del+21] Argyrios Deligkas, John Fearnley, Themistoklis Melissourgos, and Paul G. Spirakis. "Computing exact solutions of consensus halving and the Borsuk-Ulam theorem." In: *Journal of Computer and System Sciences* 117 (2021), pp. 75–98. DOI: 10.1016/j.jcss.2020.10.006.

[Den+21] Xiaotie Deng, Jack R. Edmonds, Zhe Feng, Zhengyang Liu, Qi Qi, and Zeying Xu. "Understanding PPA-completeness." In: *Journal of Computer and System Sciences* 115 (2021), pp. 146–168. DOI: 10.1016/j.jcss.2020.07.006.

[DFK17] Xiaotie Deng, Zhe Feng, and Rucha Kulkarni. "Octahedral Tucker is PPA-complete." In: *Electronic Coloquim on Computational Complexity* TR17-118 (2017). URL: https://eccc.weizmann.ac.il/report/2017/118.

[DFS20] Argyrios Deligkas, John Fearnley, and Rahul Savani. "Tree Polymatrix Games Are PPAD-Hard." In: *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming, ICALP*. Vol. 168. LIPIcs. 2020, 38:1–38:14. DOI: 10.4230/LIPIcs.ICALP.2020.38.

[DGP09] Constantinos. Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. "The Complexity of Computing a Nash Equilibrium." In: *SIAM Journal on Computing* 39.1 (2009), pp. 195–259. DOI: 10.1137/070699652.

[DK11] Samir Datta and Nagarajan Krishnamurthy. "Some Tractable Win-Lose Games." In: *Proceedings of the 8th International Conference on Theory and Applications of Models of Computation, TAMC*. Vol. 6648. Lecture Notes in Computer Science. Springer, 2011, pp. 365–376. DOI: 10.1007/978-3-642-20877-5_36.

[Doh+17] Jérôme Dohrau, Bernd Gärtner, Manuel Kohler, Jiří Matoušek, and Emo Welzl. "ARRIVAL: a zero-player graph game in NP ∩ coNP." In: *A journey through discrete mathematics*. Springer, 2017, pp. 367–374. DOI: 10.1007/978-3-319-44479-6_14.

[DP06] Constantinos Daskalakis and Christos H. Papadimitriou. "Computing pure nash equilibria in graphical games via markov random fields." In: *Proceedings of the 7th ACM Conference on Electronic Commerce (EC-2006)*. ACM, 2006, pp. 91–99. DOI: 10.1145/1134707.1134718.

[DP11] Constantinos Daskalakis and Christos Papadimitriou. "Continuous Local Ssearch." In: *Proceedings of the 22nd annual ACM-SIAM Symposium on Discrete algorithms*. 2011, pp. 790–804. DOI: 10.1137/1.9781611973082.62.

[DP20] Constantinos Daskalakis and Christos H. Papadimitriou. *Continuous Local Search - Corrigendum*. 2020. URL: http://people.csail.mit.edu/costis/CLS-corrigendum.pdf.

[DQY11] Chuangyin Dang, Qi Qi, and Yinyu Ye. *Computational models and complexities of Tarski's fixed points.* Tech. rep. 2011. URL: https://web.stanford.edu/~yyye/unitarski1.pdf.

[DTZ18] Constantinos Daskalakis, Christos Tzamos, and Manolis Zampetakis. "A converse to Banach's fixed point theorem and its CLS-completeness." In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 2018, pp. 44–50. DOI: 10.1145/3188745.3188968.

[EGG06] Edith Elkind, Leslie Ann Goldberg, and Paul Goldberg. "Nash Equilibria in Graphical Games on Trees Revisited." In: *Proceedings of the 7th ACM conference on Electronic Commerce, EC*. 2006, pp. 100–109. DOI: 10.1145/1134707.1134719.

[ET11] Robert Elsässer and Tobias Tscheuschner. "Settling the Complexity of Local Max-Cut (Almost) Completely." In: *Proceedings of the 38th Annual International Colloquium on Automata, Languages, and Programming*. Vol. 6755. 2011, pp. 171–182. DOI: 10.1007/978-3-642-22006-7_15.

[Ete+20] Kousha Etessami, Christos Papadimitriou, Aviad Rubinstein, and Mihalis Yannakakis. "Tarski's Theorem, Supermodular Games, and the Complexity of Equilibria." In: *Proceedings of the 11th Innovations in Theoretical Computer Science Conference, ITCS*. Vol. 151. LIPIcs. 2020, 18:1–18:19. ISBN: 978-3-95977-134-4. DOI: 10.4230/LIPIcs.ITCS.2020.18.

[EY10] Kousha Etessami and Mihalis 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.

[Fea+20] John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul 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.

[Fea+21] John Fearnley, Paul W. Goldberg, Alexandros Hollender, and Rahul Savani. "The Complexity of Gradient Descent: CLS = PPAD ∩ PLS." In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. Association for Computing Machinery, 2021, 46–59. ISBN: 9781450380539. DOI: 10.1145/3406325.3451052.

[FGV16] Diodato Ferraioli, Paul W. Goldberg, and Carmine Ventre. "Decentralized dynamics for finite opinion games." In: *Theoretical Computer Science* 648 (2016), pp. 96–115. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2016.08.011.

[FPS20] John Fearnley, Dömötör Pálvölgyi, and Rahul Savani. "A faster algorithm for finding Tarski fixed points." In: *CoRR* abs/2010.02618 (2020). DOI: 10.48550/arXiv.2010.02618. arXiv: 2010.02618.

[FPT04] Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar. "The complexity of pure Nash equilibria." In: *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*. 2004, pp. 604–612. DOI: 10.1145/1007352.1007445.

[FRG18] Aris Filos-Ratsikas and Paul W. Goldberg. "Consensus halving is PPA-complete." In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 2018, pp. 51–64. DOI: 10.1145/3188745.3188880.

[FRG19] Aris Filos-Ratsikas and Paul W. Goldberg. "The Complexity of Splitting Necklaces and Bisecting Ham Sandwiches." In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 638–649. DOI: 10.1145/3313276.3316334.

[FS21] John Fearnley and Rahul Savani. "A Faster Algorithm for Finding Tarski Fixed Points." In: *Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science, STACS*. Vol. 187. LIPIcs. 2021, 29:1–29:16. ISBN: 978-3-95977-180-1. DOI: 10.4230/LIPIcs.STACS.2021.29.

[GD03] Andrzej Granas and James Dugundji. *Fixed Point Theory.* Springer, 2003. ISBN: 0387001735.

[GGS05] Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. "Pure Nash Equilibria: Hard and Easy Games." In: *Journal of Artificial Intelligence Research* 24 (2005), pp. 347–406. DOI: 10.1613/jair.1683.

[GH19] Paul W. Goldberg and Alexandros Hollender. "The Hairy Ball Problem is PPAD-Complete." In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming*. Vol. 132. LIPIcs. 2019, 65:1–65:14. DOI: 10.4230/LIPIcs.ICALP.2019.65.

[Göö+22] Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. "Further Collapses in TFNP." In: *CoRR* abs/2202.07761 (2022). DOI: https://doi.org/10.48550/arXiv.2202.07761. arXiv: 2202.07761.

[GS13] D. Gale and L. S. Shapley. "College Admissions and the Stability of Marriage." In: *The American Mathematical Monthly* 120.5 (2013), pp. 386–391. DOI: 10.4169/amer.math.monthly.120.05.386.

[GZ11] Oded Goldreich and David Zuckerman. "Another Proof That BPP $\subseteq$ PH (and More)." In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. Vol. 6650. Lecture Notes in Computer Science. Springer, 2011, pp. 40–53. DOI: 10.1007/978-3-642-22670-0_6.

[HG18] Alexandros Hollender and Paul W. Goldberg. "The Complexity of Multisource Variants of the End-of-Line Problem, and the Concise Multilated Chessboard." In: *Electronic Coloquim on Computational Complexity* TR18-120 (2018). URL: https://eccc.weizmann.ac.il/report/2018/120/.

[HV21] Pavel Hubácek and Jan Václavek. "On Search Complexity of Discrete Logarithm." In: *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science, MFCS*. Ed. by Filippo Bonchi and Simon J. Puglisi. Vol. 202. LIPIcs. 2021, 60:1–60:16. DOI: 10.4230/LIPIcs.MFCS.2021.60.

[HY17] Pavel Hubáček and Eylon Yogev. "Hardness of Continuous Local Search: Query Complexity and Cryptographic Lower Bounds." In: *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*. 2017, pp. 1352–1371. DOI: 10.1137/17M1118014.

[IK18] Takashi Ishizuka and Naoyuki Kamiyama. "On the Complexity of Stable Fractional Hypergraph Matching." In: *Proceedings of the 29th International Symposium on Algorithms and Computation, ISAAC 2018*. Vol. 123. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 11:1–11:12. DOI: 10.4230/LIPIcs.ISAAC.2018.11.

[IK22a] Takashi Ishizuka and Naoyuki Kamiyama. *NP-hardness of Computing Uniform Nash Equilibria on Planar Bimatrix Game*. 2022. DOI: 10.48550/ARXIV.2205.03117.

[IK22b] Takashi Ishizuka and Naoyuki Kamiyama. *On Finding Pure Nash Equilibria of Discrete Preference Games and Network Coordination Games*. 2022. DOI: 10.48550/arXiv.2207.01523.

[Ish21a] Takashi Ishizuka. "On the complexity of finding a Caristi's fixed point." In: *Information Processing Letters* 170 (2021), p. 106119. DOI: 10.1016/j.ipl.2021.106119.

[Ish21b] Takashi 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.

[Jeř16] Emil 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.

[JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis 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.

[Kha80] Leonid G Khachiyan. "Polynomial algorithms in linear programming." In: *USSR Computational Mathematics and Mathematical Physics* 20.1 (1980), pp. 53–72. DOI: 10.1016/0041-5553(80)90061-0.

[Kin+13] Shiva Kintali, Laura J. Poplawski, Rajmohan Rajaraman, Ravi Sundaram, and Shang-Hua Teng. "Reducibility among Fractional Stability Problems." In: *SIAM Journal on Computing* 42.6 (2013), pp. 2063–2113. DOI: 10.1137/120874655.

[Kle+21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. "Total Functions in the Polynomial Hierarchy." In: *Proceedings of the 12th Innovations in Theoretical Computer Science Conference, ITCS*. Vol. 185. LIPIcs. 2021, 44:1–44:18. DOI: 10.4230/LIPIcs.ITCS.2021.44.

[KLS01] Michael J. Kearns, Michael L. Littman, and Satinder P. Singh. "Graphical Models for Game Theory." In: *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, UAI*. 2001, pp. 253–260.

[KNY19] Ilan Komargodski, Moni Naor, and Eylon 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.

[LCS16] Quang Duy Lã, Yong Huat Chew, and Boon-Hee Soong. *Potential Game Theory.* Springer, Cham, 2016. ISBN: 9783319809038.

[Lev73] Leonid A. Levin. "Universal sequential search problems." In: *Problems of Information Transmission* 9.3 (1973), pp. 265–266.

[Lol+19] Phani Raj Lolakapuri, Umang Bhaskar, Ramasuri Narayanam, Gyana R. Parija, and Pankaj S. Dayama. "Computational Aspects of Equilibria in Discrete Preference Games." In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI*. 2019, pp. 471–477. DOI: 10.24963/ijcai.2019/67.

[LSW15] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. "A Faster Cutting Plane Method and its Implications for Combinatorial and Convex Optimization." In: *Proceeding of the IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS*. IEEE Computer Society, 2015, pp. 1049–1065. DOI: 10.1109/FOCS.2015.68.

[MP91] Nimrod Megiddo and Christos H. Papadimitriou. "On total functions, existence theorems and computational complexity." In: *Theoretical Computer Science* 81.2 (1991), pp. 317–324. ISSN: 0304-3975. DOI: 10.1016/0304-3975(91)90200-L.

[MS21a] Serge Massar and Miklos Santha. "Characterising the intersection of QMA and coQMA." In: *Quantum Information Processing* 20.12 (2021), p. 396. DOI: 10.1007/s11128-021-03326-3.

[MS21b] Serge Massar and Miklos Santha. "Total functions in QMA." In: *Quantum Information Processing* 20.1 (2021), p. 35. DOI: 10.1007/s11128-020-02959-0.

[NJ50] John F Nash Jr. "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.

[NV15] Thanh Nguyen and Rakesh Vohra. "Near Feasible Stable Matchings." In: *Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC '15, Portland, OR, USA, June 15-19, 2015*. Ed. by Tim Roughgarden, Michal Feldman, and Michael Schwarz. ACM, 2015, pp. 41–42. DOI: 10.1145/2764468.2764471.

[OPR16] Abraham Othman, Christos Papadimitriou, and Aviad Rubinstein. "The complexity of fairness through equilibrium." In: *ACM Transactions on Economics and Computation (TEAC)* 4.4 (2016), pp. 1–19. DOI: 10.1145/2956583.

[Orl09] James B. Orlin. "A faster strongly polynomial time algorithm for submodular function minimization." In: *Mathematical Programming* 118.2 (2009), pp. 237–251. DOI: 10.1007/s10107-007-0189-2.

[Pap05] Christos H. Papadimitriou. "Computing correlated equilibria in multiplayer games." In: *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*. ACM, 2005, pp. 49–56. DOI: 10.1145/1060590.1060598.

[Pap07] Christos H Papadimitriou. "The complexity of finding Nash equilibria." In: *Algorithmic game theory* 2 (2007), p. 30.

[Pap94a] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994. ISBN: 978-0-201-53082-7.

[Pap94b] Christos 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.

[Pol95] Svatopluk Poljak. "Integer Linear Programs and Local Search for Max-Cut." In: *SIAM Journal on Computing* 24.4 (1995), pp. 822–839. DOI: 10.1137/S0097539793245350.

[Sca67] Herbert E Scarf. "The core of an N person game." In: *Econometrica: Journal of the Econometric Society* (1967), pp. 50–69. DOI: 10.2307/1909383.

[Sha53] Lloyd S Shapley. "Stochastic games." In: *Proceedings of the national academy of sciences* 39.10 (1953), pp. 1095–1100. DOI: 10.1073/pnas.39.10.1095.

[Sip97] Michael Sipser. *Introduction to the theory of computation.* PWS Publishing Company, 1997. ISBN: 978-0-534-94728-6.

[Spe28] E. Sperner. "Neuer beweis für die invarianz der dimensionszahl und des gebietes." In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 6.1 (1928), pp. 265 –272. ISSN: 0025-5858. DOI: 10.1007/BF02940617.

[SS04] Rahul Savani and Bernhard von Stengel. "Exponentially Many Steps for Finding a Nash Equilibrium in a Bimatrix Game". In: *Proceedings of the 45th Symposium on Foundations of Computer Science*. IEEE Computer Society, 2004, pp. 258–267. DOI: 10.1109/FOCS.2004.28.

[SZZ18] Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. "PPP-Completeness with Connections to Cryptography." In: *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*. IEEE Computer Society, 2018, pp. 148–158. DOI: 10.1109/FOCS.2018.00023.

[Tar55] Alfred Tarski. "A lattice-theoretical fixpoint theorem and its applications." In: *Pacific journal of Mathematics* 5.2 (1955), pp. 285–309. DOI: 10.2140/pjm.1955.5.285.

[Tsc10] Tobias Tscheuschner. "The local max-cut problem is PLS-complete even on graphs with maximum degree five." In: *CoRR* abs/1004.5329 (2010). DOI: 10.48550/arXiv.1004.5329. arXiv: 1004.5329.

[Yan09] Mihalis Yannakakis. "Equilibria, fixed points, and complexity classes." In: *Computer Science Review* 3.2 (2009), pp. 71–85. DOI: 10.1016/j.cosrev.2009.03.004.

[ZF87] Stathis Zachos and Martin Furer. "Probabilistic quantifiers vs. distrustful adversaries." In: *International Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer. 1987, pp. 443–455. DOI: 10.1007/3-540-18625-5_67.