Smoothed Complexity of Network Coordination Games

by

Julian T. Viera

B.S. in Electrical Engineering and Computer Science
Massachusetts Institute of Technology (2021)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author .....
Department of Electrical Engineering and Computer Science
May 6, 2022

Certified by .....
Constantinos Daskalakis
Professor
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Smoothed Complexity of Network Coordination Games

by

Julian T. Viera

Submitted to the Department of Electrical Engineering and Computer Science
on May 6, 2022, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The problem of finding or computing Nash equilibria has been an important problem in economics and computer science for decades. Classical worst-case and expected-case analyses have shown that in many cases for many types of games, computing Nash equilibria is intractable. However, it has been empirically shown that in many instances, approximate Nash equilibria can be computed efficiently. Thus, there is a growing interest in the smoothed complexity of games. That is, the complexity of computing Nash equilibria when the inputs to the problem are confined to look more like real-world inputs. This thesis provides a further analysis of the smoothed complexity of network coordination games. We specifically look at the smoothed complexity of the 2-Flip algorithm. While we do not prove that using the 2-Flip algorithm on 2-Flip-Max-Cut achieves smoothed quasipolynomial time, we discuss multiple attempts at this goal, and hope to provide other researchers with the inspiration to prove quasipolynomial time.

Thesis Supervisor: Constantinos Daskalakis
Title: Professor

Acknowledgments

I could not have written this thesis without the help of countless people along the way.

I would like to especially thank Costis Daskalakis and Noah Golowich for their endless support throughout the research and writing process. Your advice throughout our meetings has been wonderful. Both of you have provided me with the confidence and the knowledge to write this thesis.

Of course I can't forget the endless cheers I got from New Jersey along the way. To Mom, Dad, Noah, Markela, Tyler, Tanner, Aunt Laura, and all of the other Vieras and Evangelistas, I cannot thank you enough for your encouragement every step of the way.

Thank you also to my roommates at Norf! Spencer, Mark, Tyler, and Keithen, your confidence in me every day drove me forward. Thank you for the many times you let me discuss my work and my questions with you. You may not realize it, but those discussions played a major role in figuring out many of the problems discussed here.

Thank you to every teacher in Colonia and New Egypt that I ever had. Thank you to every professor in every course I've taken at MIT. Thank you to all the friends and classmates I've met along the way. Thank you, once again, to my family and my closest friends. You know who you are.

And thank you God.

Contents

Chapter 1

Introduction

At the heart of algorithmic game theory is the problem of computing Nash equilibria. The problem asks: Given a game with some players, where each player has a set of strategies they can take and a reward function dependent on the actions each player takes, find a local optimum. In other words, find a set of strategies for each player such that no player can unilaterally improve their payoff.

While it is known that Nash equilibria always exist, the problem of actually computing them for general games is known to be intractable. However, there are many games for which the complexity of computing Nash equilibria is an open problem. In fact, the creation of new complexity classes such as PLS and CLS were motivated in part by the complexity of computing Nash equilibria.

Historically, many problems in algorithmic game theory have still been shown to be intractable. However, empirical evidence has shown that many of these problems are rarely hard in realistic situations. Worst-case analysis is dominated by artificially constructed cases that are not representative of typical instances. At the same time, traditional average-case analysis is dominated by completely random inputs which have certain properties with high probability, which is also not representative of typical instance This gap between the observed efficiency of game theory algorithms and a lack of formal proofs to support said efficiency is slowly being filled by a relatively new area of research: smoothed complexity. Smoothed complexity analysis considers inputs perturbed with random noise. This is much more analogous to real-world

inputs, and allows us to get complexity results that are more in line with real-life efficiency.

This paper provides an analysis of the smoothed complexity of network coordination games and a study of 2-Flip-Max-Cut. Past research has shown that network coordination games have polynomial smoothed complexity for complete graphs, and pseudopolynomial smoothed complexity for arbitrary graphs. We provide a discussion of the smoothed complexity of 2-Flip-Max-Cut, a closely related graph problem, and analyze multiple attempts to show that this problem has a smoothed quasipolynomial time.

1.1 Game Theory Background

Before we begin, it is useful to introduce important concepts in game theory that will be mentioned throughout this paper.

1.1.1 Games and Nash Equilibria

Formally, we define a game as a set of players, where each player has some set of strategies they can choose from. We define a "strategy profile" for a player as a deterministic or probabilistic choice of strategies that player takes. We define the payoff that a player receives as the value they obtain, and it is a function of both their strategy profile and the profiles of all other players. We assume that each player is selfish, and they are only trying to maximize their individual payoffs. We also assume that players do not communicate with each other and have no ability to collude.

A Nash equilibrium is defined as a set of strategy profiles where no individual player can unilaterally improve their payoff. In other words, assuming all other players keep the strategy profiles assigned to them in the equilibrium, the current player has no better option than their equilibrium strategy profile. A Nash equilibrium in which each player determines their strategies deterministically (i.e. each player chooses one strategy to play) is referred to as a pure Nash equilibrium. On the other hand, a Nash

equilibrium in which any players choose their strategies probabilistically is referred to as a mixed Nash equilibrium. Note that a Nash equilibrium may not maximize the total payoff of all players in the game, and the strategy profiles that do maximize the total payoff do not necessarily constitute a Nash equilibrium, since in that case one player may be able to improve their individual payoff by changing their strategy. Rather, a Nash equilibrium represents a local optimum in some sense. And due to the fact that no player can improve by acting unilaterally, a Nash equilibrium is considered a point of reasonable stability (again, assuming players cannot collude).

Nash equilibria are an extremely important area of research because of their prevalence in the real world. Many situations in nature, markets, social networks, and various other fields can be modelled as games. Determining where and if these real-world games will converge to a Nash equilibria is a significant area of interest.

1.1.2 Complexity Classes

The classes P and NP are familiar to anyone in the field of computer science. However, these complexity classes do not sufficiently capture the complexity of many problems in algorithmic theory. Computing Nash equilibria for general games is known to be intractable, and even for many simple games, finding Nash equilibria is hard under worst-case analysis. However, a variety of empirical evidence shows that many different types of games do indeed converge to Nash equilibria in nature, and there are many algorithms that have been shown to efficiently compute Nash equilibria for certain games with real-world inputs. To better identify the complexity of game theoretic problems in the real world, new complexity classes have been defined to encapsulate these problems. These complexity classes also have important applications in machine learning and other areas of theoretical computer science.

Papadimitriou [8] defined the class PPAD which encapsulates the complexity of computing Nash equilibria in general games. The class can be defined by one of its complete problems, End-Of-The-Line. The problem states, given a graph G (possibly exponentially large, i.e. represented by a set of vertices and black-box predecessor/successor function to compute predecessors/successors of a given node) with no

isolated vertices, with every vertex having at most one predecessor and one successor, and a "source" vertex s in G with no predecessor, find a vertex $t \neq s$ that is either a "sink" node with no successor or another "source" node with no predecessor. The abbreviation stands for "Polynomial Parity Argument on Directed Graphs", and it is the class of problems that can be shown to be total by a parity argument on a graph. It has been shown that computing Nash equilibria in general is equivalent to End-Of-The-Line, and thus computing Nash equilibria is also a PPAD-complete problem.

Johnson et al. [7] introduced the class PLS, for "Polynomial Local Search" to describe the complexity of local search problems. The class can be defined by the problem Local-Opt: Given a polynomial-time neighborhood function f and a polynomial-time potential function p, find a point x that is a local minimum of p with respect to f. The class PLS is smaller than PPAD, but many problems in game theory and machine learning have been shown to be PLS-complete.

To analyze problems that have a solution because of both a fixed point argument (like PPAD) and a potential argument (like PLS), Daskalakis [5] introduced CLS, which is equal to PPAD $\cap$ PLS. An example of a CLS-complete problem is Banach-Fixed-Point, in which we seek to find an approximate fixed point of a contraction map f on a metric space (M, d). Because this problem is a fixed point computation, it is obviously in PPAD. But note that finding a fixed point is the equivalent of minimizing the distance $d(f(x), x)$ between a point x and its corresponding mapping $f(x)$. Therefore, we can use $p(x) \coloneqq d(f(x), x))$ as a potential function to see that this problem is in PLS. Many problems of interest are known to lie within CLS, including finding KKT points and computing mixed Nash equilibria in coordination games, congestion games, and simle stochastic games. CLS more tightly encapsulates the complexity of many game theoretic problems. This paper, and much of the current research in algorithmic game theory, seeks to find even tighter complexity bounds on specific problems in game theory, via algorithmic approaches and reductions to other problems.

1.2 Discussion Outline

Here we will provide a high-level overview of the discussion throughout the rest of the paper.

Chapter 2 introduces prior research. We will introduce smoothed complexity, network coordination games, and local max cut and results from other papers on these topics. We will introduce some useful proofs from other papers at a high level to motivate our discussion in Chapters 3 and 4.

Chapter 3 will discuss our key ideas and techniques at a high level. We will also explain variables and definitions that will be used throughout Chapter 4.

Chapter 4 will show our analysis of the 2-Flip Algorithm for 2-Flip-Max-Cut. While we do not prove a quasipolynomial runtime, we discuss attempts at a proof that can potentially act as a starting point for future research. We will also discuss how proof of a quasipolynomial runtime relates to prior work in network coordination games.

Lastly, Chapter 5 will provide a review of our discussion and how it relates to prior work. We will then discuss the most pressing questions that have yet to be answered in the literature.

Chapter 2

Related Work

In this chapter, we will discuss some of the related work around smoothed complexity and its use in network coordination games. We will introduce the definition of smoothed complexity and explain how it has been used to study network coordination games and Local-Max-Cut. We will also introduce smoothness-preserving reductions and how prior research in this area motivates our main result.

2.1 Smoothed Complexity

While there are many local search problems (traveling salesman, clustering, linear programming, etc.) with algorithms that are efficient in practice, there has been a growing desire for a theoretical support of this efficiency. In many of these cases, there exist contrived adversarial inputs that can make reaching a local optimum take exponential time, and thus worst-case analysis does not accurately represent practical efficiency. Additionally, expected case analysis considers random inputs, which often have special properties with high probability. Again, this is not an accurate representation of the real world. To solve this issue, Spielman and Teng [10] invented smoothed analysis, a more realistic approach in which inputs are perturbed by a small amount of random noise. Since this is often the case in reality with measurement errors, rounding errors, etc., smoothed analysis provides a much more useful analysis for equilibrium computation problems.

In some sense, smoothed complexity is concerned primarily with cases that could actually arise in the real world. Thus, a polynomial smoothed complexity for computing Nash equilibria for a specific game, while less strict than worst-case, formally gives us confidence that that algorithm will find a Nash equilibrium in polynomial time in practice. This also gives us confidence that in real-world situations or environments that resemble a game theory problem, the players will converge to a Nash equilibrium in polynomial time.

Since computing Nash equilibria is ultimately a local optimization problem, much of the recent research uses approaches from smoothed analysis. The following four sections provide more detail on related work on smoothed algorithms for Local-Max-Cut (a closely related optimization problem), and some recent smoothed analysis for network coordination games.

2.2 Local-Max-Cut and the FLIP Algorithm

The problem of Max-Cut is, given a graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$, to find a partition (V_1, V_2) of the vertices that maximizes the sum of the edges between V_1 and V_2. Local-Max-Cut is a variant of this problem in which we seek to find a "locally optimal" maximum cut. That is, a cut that cannot be improved upon by unilaterally moving any one node in the graph from one partition to the other. One way to solve this is the FLIP algorithm, which takes an arbitray starting partition (V_1, V_2) and iteratively moves one node from V_1 to V_2 or vice versa. Throughout this paper, we will also refer to Local-Max-Cut as 1-Flip-Max-Cut, as The algorithm continues until all possible transitions of one node do not improve the total weight of the cut. In the smoothed analysis approach, we consider an input graph $G = (V, E)$ where instead of fixing each edge weight deterministically, they are described using a probability density function $f_e : [-1, 1] \to [0, \phi]$. While it is known that finding a locally optimal solution is PLS-complete, Etscheid and Roglin [6] show that the smoothed number of steps of the FLIP algorithm is bounded from above by a polynomial in $n^{\log n}$ and ϕ.

Note that there are an exponential number of possible flips, so in the worst-case, the FLIP algorithm will take exponential time for Local-Max-Cut. However, in the smoothed case, we are guaranteed a quasipolynomial time regardless of initial cut and pivot rules [6]. While Local-Max-Cut might not seem related to game theory on the surface, it is a local optimization problem with a potential function that has many relationships to a variety of games, including network coordination games. Additionally, the smoothed analysis approach of Etscheid and Roglin [6] provides a foundation for the following related work and the work conducted for this paper.

Further research has led to significant improvements from the quasipolynomial time given here. Building on the same smoothed complexity framework, Angel et al. [1] showed that any implementation of the FLIP algorithm for Local-Max-Cut terminates in at most $n(\phi \log n)^c$ steps for complete graphs, where c is a universal constant. This bound was then improved upon even further by Bibak et al. [2], which showed that for every constant $\eta > 0$, any implementation of the FLIP algorithm terminates in $O(\phi n^{7.829+3.414\eta})$ steps. Most recently, Chen and Guo [4] improved the polynomial in $n^{\log n}$ and ϕ bound by Etscheid and Roglin [6] for general graphs to a polynomial in $n^{O(\sqrt{\log n})}$ and ϕ.

2.3 Generalizing to Max-k-Cut

The problem of Local-Max-Cut is sometimes referred to as Max-2-Cut, as the problem is to find a local maximum of the cut weight where we partition the nodes into two disjoint sets. The more general problem, Local-Max-k-Cut, seeks to find a local maximum cut weight when the vertices are partitioned into k disjoint sets.

Smoothed complexity for variable k is an active area of research. Research by Bibak et. al. [2] shows that Local-Max-3-Cut can be solved in smoothed polynomial time for complete graphs, and Local-Max-k-Cut for arbitrary k can be solved in quasipolynomial time for complete graphs. The smoothed complexity of Local-Max-k-Cut for arbitrary graphs is still an open problem.

We suspect that Local-Max-k-Cut has a close relationship with network coor-

dination games, as we can think of each of the k partitions as holding one of k strategies for each player. A formal smoothness-preserving reduction from network coordination games to Local-Max-k-Cut would provide an alternative algorithm for computing Nash equilibria in network coordination games, and potentially provide speedup results in practice.

2.4 Network Coordination Games

We define a network coordination game by an undirected graph $G = (V, E)$ where each node $v \in V$ plays a 2-player coordination game with each of its neighbors. Each node has k possible strategies to choose from, so each coordination game between each (u, v) pair is represented by a payoff matrix A_{uv}. Once each node v takes an action, the payoffs are fixed, and we define the payoff for an individual node v to be the sum of all of its payoffs. We define NetCoordNash as the problem of finding a Nash equilibrium in a network coordination game. More specifically, k-NetCoordNash is the problem where each player has k strategies.

Network coordination games are an important area of research in computer science and beyond because of their many applications. Network problems in computer science, social networks, job finding, and voting are just a few of the problems that can be modelled by a network coordination game.

Boodaghians et al. [3] show that k-NetCoordNash, a PLS-complete problem, has a smoothed quasipolynomial time algorithm. They do this both through an algorithmic approach and via reductions.

For the algorithmic approach, they show that when k is a constant and each entry of each A_{uv} is perturbed independently at random, any "sufficiently long" sequence of steps of any better-response algorithm will make only little improvement with low probability. Their analysis combines and extends much of the work done by Angel et. al. [1] on Local-Max-Cut, while also showing that Local-Max-Cut is a special case of network coordination games where $k = 2$ [3]. The proof also relies heavily on rank analysis and union bounds. This analysis mirrors the framework created by Etscheid

and Roglin [6] to show quasipolynomial smoothed complexity for 1-Flip-Max-Cut. In Chapter 4, we will use a similar framework in proving our main result.

Boodaghians et al. [3] also provide a well-defined framework for randomized, smoothness-preserving reductions. With this framework, an alternative algorithm for 2-strategy network coordination games is shown, and a conditional one for general network coordination games. We will introduce the concept of smoothness-preserving reductions and their importance in the following section.

2.5 Smoothness-Preserving Reductions

In algorithmic analysis, it is often useful to reduce a problem to some other problem. However, the traditional method of reduction does not accurately handle smoothed inputs, because we may not have guarantees that the corresponding inputs in the reduction are also smoothed. Below we provide the definition of smoothness-preserving reductions introduced by Boodaghians et al. [3]. Note that $\boldsymbol{\sigma}$ is a vector of strategies representing the chosen strategies of each player.

Definition 2.5.1 (Strong and Weak Smoothness-Preserving Reductions)**.** *A randomized, smoothness-preserving reduction from a search problem $\mathcal{P}$ to $\mathcal{Q}$ is defined by polynomial-time computable functions f_1, f_2, and f_3, and a real probability space $\Omega \subseteq \mathbb{R}^d$, such that,*

- *For any $(I, X) \in \mathcal{P}$, and for arbitrary $R \in \Omega$, $(f_1(I), f_2(X, R))$ is an instance of $\mathcal{Q}$, such that all (locally optimal) solutions $\boldsymbol{\sigma}$ map to a solution $f_3(\boldsymbol{\sigma})$ of (I, X).*
- *Whenever the entries of X and R are drawn independently at random from distributions with density at most ϕ, then $f_2(X, R)$ has entries which are independent random variables with density at most $poly(\phi, |X|, |R|)$. This is called a strong reduction.*
- *If the entries of $f_2(X, R)$ instead of being independent are linearly independent combinations of entries of X and R, and the density is similarly bounded, then call it a weak reduction.*

This framework created by Boodaghians et al. [3] allows us to reduce problems in smoothed complexity from one to another. Ideally, we can create strong smoothness-preserving reductions, because then we can be sure that the inputs to $\mathcal{Q}$ are also perturbed by random and independent noise, and thus the instance of $\mathcal{Q}$ we get from the reduction is exactly a smoothed instance of $\mathcal{Q}$. And if $\mathcal{Q}$ has an algorithm that runs efficiently on smoothed instances, then we can immediately conclude that $\mathcal{P}$ also has an algorithm that runs efficiently on smoothed instances of $\mathcal{P}$.

While they are ideal, there has been little prior work done in the area of useful strong smoothness-preserving reductions. Weak smoothness-preserving reductions, on the other hand, have been easier to find. However, they provide a challenge. Note that if we have a weak smoothness-preserving reduction from a smoothed instance of $\mathcal{P}$ to $\mathcal{Q}$, the noise in the inputs to $\mathcal{Q}$ is no longer entirely independent. Because the inputs to $\mathcal{Q}$ have some level of correlation, it is not clear or obvious that common algorithmic techniques for smoothed problems will still work on the given instance of $\mathcal{Q}$. And if they do, it may very well be case that they are no longer as efficient as if the noise was independent. As a result, weak smoothness-preserving reductions will require careful analysis of how correlated noise affects the specific problem being used for $\mathcal{Q}$.

The good news is that recent research suggests efficient algorithms exist for at least some problems with inputs with some "small" amount of correlated noise. In Boodaghians et al. [3], they provide a weak smoothness-preserving reduction from k-NetCoordNash to 2-Flip-Max-Cut. While they do not show that 2-Flip-Max-Cut admits an efficient smoothed polynomial time algorithm with the corresponding correlation in noise, they do show that if a problem $\mathcal{P}$ has a weak smoothness-preserving reduction to Local-Max-Cut (also known as 1-Flip-Max-Cut) on an arbitrary complete graph, then $\mathcal{P}$ has a quasipolynomial smoothed complexity. This also provides the framework for us to conclude that once we show that 2-Flip-Max-Cut can be solved efficiently with the FLIP algorithm, then so can the smoothed case of network coordination games.

A high-level overview of their reduction from k-NetCoordNash to 2-Flip-Max-Cut

and the motivation for our work are discussed in the following sections.

2.6 k-NetCoordNash to 2-Flip-Max-Cut

We define an instance of a network coordination game as a graph $G = (V, E)$ where each node $v \in V$ represents a player, and each edge $(u, v) \in E$ has a matrix A_{uv}, where $A_{uv}[i, j]$ is the payoff players u and v receive when u takes strategy i and v takes strategy j. We refer to a set of pure strategy profiles with the vector $\boldsymbol{\sigma}$.

For a weak smoothness-preserving reduction from k-NetCoordNash to 2-Flip-Max-Cut, Boodaghians et al. [3] take the graph G from the network coordination game instance and create a new graph G' which has locally optimal cuts that correspond to strategy profiles, and cut values that correspond to the total payoff of those strategy profiles.

The graph is created as follows. For each node $u \in V$ and each strategy i, create a node (u, i) to represent player u choosing strategy i. Also create two terminal nodes s and t, and connect all other nodes to them. Draw edges between each $(u, *)$, and draw edges between each pair $(u, *)$ and $(v, *)$ if u and v share an edge in the network coordination game. The edges are then weighted such that an optimal partition into sets S and T must have $s \in S$ and $t \in T$, and S contains exactly one pair $(u, *)$ for each player u. Here S corresponds to a pure Nash equilibrium where if (u, i) appears in S, that corresponds to player u picking strategy i.

Also note that any player u moving from strategy i to another strategy i' would be represented by taking (u, i) out of S and replacing it with (u, i'). In other words, two nodes in the graph would be moved, or a 2-flip. Therefore, any locally maximum cut up to 2 flips would exactly correspond to a pure Nash equilibria.

Also note that any player u moving from strategy i to another strategy i' would be represented by taking (u, i) out of S and replacing it with (u, i'). In other words, two nodes in the graph would be moved, or a 2-flip.

2.7 Motivation for Our Work

Below we discuss how the presented related work motivates our work on Local-Max-Cut and network coordination games.

2.7.1 Local-Max-Cut

The problem of Local-Max-Cut has been studied extensively by theoretical computer scientists and algorithmic game theorists alike. However, there remains a significant gap between the runtime of the FLIP algorithm empirically and what researches have been able to formally prove.

Additionally, it is still an open problem how generalizing Local-Max-Cut affects the runtime of the FLIP algorithm. Boodaghians et al. [3] briefly discuss d-Flip-Max-Cut, in which we wish to find a local maximum cut up to d flips of nodes. Obviously, increasing d causes some complexity increase, but it is conjectured that at least in the case of $d = 2$, the runtime of the FLIP algorithm is the same as 1-Flip-Max-Cut. We provide a definition of a 2-Flip algorithm for smoothed instances of the 2-Flip-Max-Cut problem, and seek to provide a starting point for a proof that this algorithm runs in quasipolynomial time.

2.7.2 Network Coordination Games

As Boodaghians et al. [3] have shown, 2-Flip-Max-Cut is a problem closely related to network coordination games. Additionally, the 2 flips may correspond to a player switching strategies in numerous other games, so it is possible that 2-Flip-Max-Cut has close relationships with other types of games as well.

Because Boodaghians et al. [3] does not directly show that 2-Flip-Max-Cut can be solved efficiently, the most efficient algorithm known for network coordination games (presented by Boodaghians et al. [3] in which they apply the Better Response Algorithm directly to a smoothed instance of a network coordination game) still depends exponentially on k. It is realistic to consider large values of k, and thus we wish to prove a runtime independent of k. If the 2-Flip algorithm converges in quasipolyno-

mial time for 2-Flip-Max-Cut, we can conclude, via the weak smoothness-preserving reduction presented by Boodaghians et al. [3] from k-NetCoordNash to 2-Flip-Max-Cut, that k-NetCoordNash also has a quasipolynomial algorithm independent of k.

Chapter 3

Techniques and Definitions

In this chapter, we will provide a high-level overview of the key techniques we use in our attempts to show that 2-Flip-Max-Cut has a quasipolynomial runtime using the 2-FLIP algorithm. In Chapter 4 we will see a more technical analysis of these techniques.

3.1 High-Level Proof Overview

Our framework follows the general outline provided in Etscheid et al. [6] that shows a quasipolynomial runtime for the FLIP algorithm on 1-Flip-Max-Cut. At a high level, the steps of the proof are as follows:

1. Define the 2-FLIP algorithm.

2. Show that for any "k-repeating sequence", there is some subset T of the set of repeating pairs D such that the linear combinations of these pairs are linearly independent.

3. Using Step 2, conclude that any k-repeating sequence has linear combinations of a sufficient rank.

4. Show that for any k-repeating sequence of $5n^2$ flips, the probability that all of the steps made little to no improvement is very low.

5. Show that it is impossible for any sequence of $5n^2$ flips to not be a k-repeating sequence.

6. Conclude that the expected number of flips is a quasipolynomial in ϕ and n.

The key difference here from the proof in Etscheid et al. [6] is that rather than a flip being an individual node, we are now defining a flip as a random pair of nodes that are being flipped. Careful work is needed here to ensure that the framework of the proof for 1-Flip-Max-Cut works in this case.

3.2 Relation to Network Coordination Games

In Theorem 3.6 of Boodaghians et al. [3], it is shown that k-NetCoordNash (the problem of computing a pure Nash equilibrium in network coordination games where each player has k strategies), admits a weak smoothness-preserving reduction to 2-Flip-Max-Cut. However, without any proof on the complexity of 2-Flip-Max-Cut, the conclusion we are left with from Boodaghians et al. [3] is that k-NetCoordNash has a smoothed quasipolynomial time with k in the exponent.

Suppose instead that one could show that the 2-FLIP algorithm solves smoothed instances of 2-Flip-Max-Cut in quasipolynomial time. Then, as noted in Boodaghians et al. [3], this would imply that there exists a smoothed quasipolynomial algorithm for k-NetCoordNash for non-constant k. Such a result would significantly improve upon the current best runtime analysis, which increases exponentially with k.

Also note that in the reduction from k-NetCoordNash to 2-Flip-Max-Cut in Boodaghians et al. [3], the graph produced for 2-Flip-Max-Cut is complete if and only if the graph for the given smoothed instance of k-NetCoordNash is complete as well. Therefore, a potential weaker (but still valuable) result would be to show that smoothed instances of 2-Flip-Max-Cut where the graph is complete have a quasipolynomial runtime. If this were the case, then we could conclude that smoothed instances of k-NetCoordNash where the graph is complete have a smoothed efficient algorithm for non-constant k.

The Etscheid and Roglin paper [6] shows that the FLIP algorithm (i.e. the 1-Flip algorithm where each flip is only one node) for 1-Flip-Max-Cut has a runtime of a polynomial in ϕ and $n^{O(\log n)}$. While we are unable to show the same runtime for the 2-Flip algorithm on 2-Flip-Max-Cut, it is believed that 2-Flip-Max-Cut can be solved in a similar runtime. Chapter 4 will discuss our attempts at reaching this runtime.

3.3 Definitions

Below we will provide formal definitions of the two things we

Definition 3.3.1 (2-Flip Algorithm)**.** *Let the 2-Flip Algorithm be the following process for an instance of the 2-Flip-Max-Cut problem on a graph $G = (V, E)$:*

1. *Randomly assign nodes to the two partitions to create a random starting configuration.*

2. *Pick a random pair of nodes $u, v \in V$. If flipping each of u and v to the partition opposite of the one they are currently in, perform that flip. Repeat this process until no possible pair of nodes yields such an improvement.*

Note that the complexity of the 2-Flip Algorithm is the number of moves required for any implementation of the 2-Flip Algorithm to terminate.

Definition 3.3.2 (k-Repeating Sequence)**.** *For an instance of 2-Flip-Max-Cut with the graph $G = (V, E)$, let $n = |V|$, $m = \binom{n}{2}$ and let parameter $k = \lceil 5 \log_2 m \rceil$. We call a sequence of $\ell \in \mathbb{N}$ consecutive steps k-repeating if at least $\lceil \ell/k \rceil$ different pairs of nodes move at least twice in the sequence.*

So for each two consecutive moves of a pair of nodes, we can obtain a linear combination which only contains edges to active nodes. This will be discussed further in Chapter 4.

Chapter 4

Smoothed Analysis of 2-Flip-Max-Cut

Below we provide a detailed description at our attempts of showing a quasipolynomial runtime for 2-Flip-Max-Cut. The framework we have used largely follows that of the analysis in Etscheid and Roglin [6].

In the search for a quasipolynomial algorithm for 2-Flip-Max-Cut, we have explored two main ideas. The first was, for a set D of repeating pairs, rather than find a subset $T \subseteq D$ of size $|D|/c$ for some constant c where the linear combinations in T are linearly independent, we sought to find a subset T of size $\Omega(\sqrt{D})$.

Our second area of work was to modify the lemmas from Etscheid and Roglin [6] such that the linear combinations in D itself must be linearly independent, and thus $T = D$. The key issue here is proving that D is of a sufficient size. Both of these areas of work are discussed below.

Note also that here we are looking at 2-Flip-Max-Cut where the graph G is complete, i.e. each pair of nodes shares an edge. This fact will be vital to our analysis below.

4.1 A Smaller Rank Bound

Below we show a starting point for the analysis of the 2-Flip algorithm on 2-Flip-Max-Cut if T has size at least $\Omega(\sqrt{D})$. Note that the lemmas we introduce are similar to the ones presented in Etscheid and Roglin [6], but our definitions are slightly different

because each flip and linear combination refers to a pair rather than an individual node.

Lemma 4.1.1. *Let S be a k-repeating sequence of pairs of length ℓ with an arbitrary starting configuration. Consider the union of sets of linear combinations obtained by adding the linear combinations of two consecutive moves of a pair which moves multiple times. Then the rank of these linear combinations is at least $\sqrt{\lceil \ell/k \rceil}$.*

Proof. Let S be a k-repeating sequence of length ℓ. We construct an auxiliary graph $G' = (V, E')$ in the following way: Let $D \subseteq V \times V$ be the set of pairs (u, v) which move at least twice in the sequence S. Define $n(u, v)$ as the number of occurrences and $\pi_{uv}(i)$ as the position of the ith occurrence of a pair (u, v) in the sequence S. For a pair $(u, v) \in D$ and $1 \leq i < n(u, v)$, let $L_{uv}(i)$ be the sum of the linear combinations corresponding to the moves $\pi_{uv}(i)$ and $\pi_{uv}(i+1)$. For any $L_{uv}(i)$, let E^i_{uv} be the set of edges $\{u, w\}$ and $\{v, w\}$ connecting u and v with all nodes w which occur in $L_{uv}(i)$, i.e., all nodes w which move an odd number of times between $\pi_{uv}(i)$ and $\pi_{uv}(i + 1)$. For every pair $(u, v) \in D$ and every $1 \leq i < n(u, v)$, the set E^i_{uv} is not empty because $L_{uv}(i)$ cannot be zero in every component as it is the sum of two improving steps. Set $E_{uv} = \bigcup_i E^i_{uv}$ and $E' = \bigcup_{(u,v)\in D} E_{uv}$.

Claim 4.1.2. *Let $T \subseteq D$ where every pair $(u, v) \in T$ contains at least one node $u^* \notin T' \setminus \{u, v\}$, where T' is the set of all nodes that appear in a pair in T. If for every pair $(u, v) \in T$, the node unique to T has a neighbor $w \in V \setminus T'$ in the graph G', then there are indices $1 \leq i_{uv} < n(u, v)$ for all $(u, v) \in T$ such that the linear combinations $\{L_{uv}(i_{uv}) : (u, v) \in T\}$ are linearly independent.*

Proof. Let $(u, v) \in T$ and let u^* be the node in the pair (u, v) unique to that pair. Let $w \in V \setminus T'$ be a neighbor of u^*, so $(u^*, w) \in E^{i_{uv}}_{uv}$. By definition, the edge (u^*, w) cannot be covered by an other node in $T' \setminus \{u^*, v\}$ because no other pair in T contains u^*. Hence it does not occur in any linear combination $L_{u'v'}(i)$, $(u', v') \in T \setminus \{(u, v)\}$.

As this argument holds for every $(u, v) \in T$, the linear combinations selected this way must be linearly independent. □

Claim 4.1.3. *There exists a subset $T \subseteq D$ with $|T| \geq \Omega(\sqrt{|D|})$ and $1 \leq i_{uv} < n(u, v)$, $(u, v) \in T$, such that the linear combinations $\{L_{uv}(i_{uv}) : (u, v) \in T\}$ are linearly independent.*

Proof. We first show that for a set of pairs D, there must be a subset $T \subseteq D$ where $|T| \geq \Omega(\sqrt{|D|})$.

To prove this, let $T \subseteq D$ be the largest set of pairs each with a unique node, and let $M = |T|$. We know that the most nodes T could possibly have is $2|T| = 2M$, since every node in every pair could be unique. We know that every node that appears in D must also appear in T, because otherwise we could add a new pair with a unique node into T, which would contradict T being the largest set of pairs each with a unique node. Therefore, the maximum number of pairs in D is $\binom{2M}{2} = O(M^2)$. Since $|D| \leq O(M^2)$ and $|T| \geq M$, we conclude that $|T| = \Omega(\sqrt{D})$.

By definition, the T we defined above has at least one unique node for every pair. For a given pair (u, v), let u^* be the node in (u, v) that is unique to that pair. And let w be any node in $V \setminus T'$. Because the input graph is complete, we know that the edge (u^*, w) is in the graph. We also know, since u^* is unique to that pair, that no other pair can cover that edge. Therefore for each pair we have a unique edge, and thus T is a valid subset. □

□

Lemma 4.1.4. *Denote by $\Delta(\ell)$ the smallest improvement made by any k-repeating sequence of length ℓ where every step increases the potential with an arbitrary starting configuration. Then for any $\varepsilon > 0$,*

$$\Pr[\Delta(\ell) \leq \varepsilon] \leq (2n)^{2\ell}(2\phi\varepsilon)^{\sqrt{\lceil \ell/(2k) \rceil}}$$

Proof. We first fix a k-repeating sequence of length ℓ. As there are ℓ steps in this sequence, and each step is a pair of nodes, there are at most $n^{2\ell}$ choices for the sequence. We will use a union bound over all these $n^{2\ell}$ many choices and over all possible starting configurations of the pairs that are active in the sequence. This gives the additional factor of $2^{2\ell}$ since at most 2ℓ nodes can move.

For a fixed starting configuration and a fixed sequence, we consider a pair (u, v) which moves at least twice and linear combinations L_1 and L_2 which correspond to two consecutive moves of the pair (u, v). As after these two moves node u and v are in their original partition again, the sum $L = L_1 + L_2$ contains only weights belonging to edges between u and v and other nodes that have moved an odd number of times between the two moves of the pair (u, v). In particular, L contains only weights belonging to edges between active nodes, for which we fixed the starting configuration.

Only if $L \in (0, 2\varepsilon]$, both L_1 and L_2 can takes values in $(0, \varepsilon]$. Hence it suffices to bound the probability that $L \in (0, 2\varepsilon]$. Due to Lemma 4.1.1, the rank of the set of all linear combinations constructed like L is at least $\sqrt{\lceil \ell/k \rceil}$. We can apply Lemma B.3.1 of [9] to obtain a bound of $(2\varepsilon\phi)^{\sqrt{\lceil \ell/k \rceil}}$ for the probability that all these linear combinations take values in $(0, 2\varepsilon]$. Together with the union bound this proves the claimed bound on $\Delta(\ell)$. □

Below we show that a sequence of $5m$ steps where $m = \binom{n}{2} = O(n^2)$ must be k-repeating. The proof is exactly the same as in Etscheid and Roglin [6], except here we have m instead of n.

Lemma 4.1.5. *Let $m = \binom{n}{2} = O(n^2)$ and denote by $\Delta := \min_{1 \le \ell \le 5m} \Delta(\ell)$ the minimum improvement by any k-repeating sequence of length at most $5m$ where every step increases the potential, starting with an arbitrary starting configuration. Then Δ is a lower bound for the improvement any sequence of $5m$ steps makes.*

Proof.

Definition 4.1.6. *We call a sequence $A_1, \ldots, A_q$ of sets a* non-k-repeating block sequence of length ℓ *if the following conditions hold.*

(i) *For every $1 \leq i < q$, $|A_i| = k$.*

(ii) $1 \leq |A_q| \leq k$.

(iii) $\sum_{i=1}^{q} |A_i| = \ell$.

(iv) *For every $i \leq j$, the number of elements that are contained in at least two sets from $A_i, \ldots, A_j$ is at most $j - i$.*

We denote by $n_k(\ell)$ the cardinality of $A_1 \cup \ldots \cup A_q$ minimized over all non-k-repeating block sequences of length ℓ.

It is easy to see that any sequence S of length ℓ which does not contain a k-repeating subsequence corresponds to a non-k-repeating block sequence of length ℓ with $\lceil \ell/k \rceil$ blocks if we subdivide S into blocks of length k. It then suffices to show that there is no non-k-repeating block sequence of length $5m$ with at most m elements. In other words, $n_k(5m) > m$.

If $m \leq 3$, then there are at least two blocks $5m > k$, but $k = \lceil 5 \log_2 m \rceil > m$ such that the first condition of Definition 4.1.6 cannot be satisfied. Therefore we can assume $m \geq 4$.

Let $q = \lceil 5m/k \rceil$ and let $A_1, \ldots, A_q$ be a non-k-repeating block sequence of length $5m$ with exactly $n_k(5m)$ different elements $x_1, \ldots, x_{n_k(5m)}$ contained in $A_1 \cup \ldots \cup A_q$. Construct an auxiliary graph H as follows: Introduce a vertex $i \in V(H)$ for each set A_i. For an element x_i, let $\rho(i)$ be the number of different sets which contain x_i and let $A_{i_1}, \ldots, A_{i_{\rho(i)}}$ $(i_1 < \ldots < i_{\rho(i)})$ be these sets. Define $P_i = \{(i_j, i_{j+1}) : 1 \leq j < \rho(i)\}$. Then define $E(H) = \bigcup_{i=1}^{n_k(5m)} P_i$ as the disjoint union of these sets. Therefore, edges in H represent neighbored occurrences of a repeat element.

The rest of the proof in Etscheid and Roglin [6] is to show that $|E(H)| > 4m$. We also know that each edge in H means one less different pair we need in the sequence. Thus,

$$n_k(5m) = \sum_{i=1}^{q} |A_i| - |E(H)| = 5m - |E(H)| > 5m - 4m = m$$

Since $n_k(5m) > m$, that means that a sequence of length $5m$ that is non-k-repeating must contain more than $m = \binom{n}{2}$ pairs. Since that is impossible, we can conclude that any sequence of $5m = 5\binom{n}{2}$ must be k-repeating and satisfy our rank bounds from above. □

The key problem with this process is the size of T. In Etscheid and Roglin [6], they show that $|T| \geq |D|/3$, a crucial component in giving a runtime polynomial in ϕ and $n^{O(\log n)}$. Because here we are only able to show that $|T| \geq \Omega(\sqrt{D})$, it is not entirely clear how significantly this affects the runtime. However, we do believe that the runtime in this case would be on the order of $2^{\sqrt{n}}$ at best. While this would be a marginal improvement, it is still far from the expected runtime for 2-Flip-Max-Cut based on empirical results. To determine how this smaller subset T affects the runtime of the 2-Flip algorithm, further analysis is needed.

4.2 Redefining a k-Repeating Sequence

An alternative approach to combat this issue of T being too small could be to provide a new definition of what exactly a k-repeating sequence is.

In the framework of Etscheid and Roglin [6], a k-repeating sequence of length ℓ is just any sequence with at least $\lceil \ell/k \rceil$ elements that repeat at least twice in the sequence. The rest of the results come from proving that there is some large enough subset T of these repeats that has all of the linear independence properties that we want.

A potential idea is to define a k-repeating sequence such that the set of repeats itself contains the linear independence properties that we want. In our case with 2-Flip-Max-Cut, we would like to define a k-repeating sequence as one where at least $\lceil \ell/k \rceil$ different pairs move at least twice, *and* each of these repeating pairs contains at least one unique node relative to the other repeating pairs. In this case, for complete

graphs, D itself satisfies the desired linear independence constraint, so we can just set $T = D$.

We suspect it may be the case that D is large enough to guarantee a smoothed quasipolynomial runtime where the sequence is still of a relatively small polynomial length. However, new analysis will be required to determine how to prove this. For example, if we tried to use the same proof as Etscheid and Roglin [6] that a sequence of a certain length must be k-repeating under this new definition, it is not necessarily true that each edge in H would correspond to decreasing the number of unique pairs needed by exactly 1. Therefore, either an entirely new approach or another way to conceptualize and design H is needed here.

As it stands in the current research, improvements in the complexity of k-NetCoordNash rely significantly on the complexity of 2-Flip-Max-Cut.

Chapter 5

Concluding Remarks

5.1 Overview of Techniques and Background

In this thesis we have discussed at a broad level the pressing questions in algorithmic game theory. We introduced the complexity classes PPAD, PLS, and CLS. While these classes have significance outside of the world of game theory, their creation has been heavily influenced by the goal to further pin down the complexity of computing Nash equilibria in various games.

These complexity classes, and much of the work in algorithmic game theory in general, seek to bridge the gap between what is achieved empirically and what can be proven theoretically. While worst-case analysis considers contrived, unrealistic inputs and expected-case analysis considers completely random inputs which often share certain useful properties, smoothed analysis provides researchers a way to formally analyze the runtimes of algorithms in a more realistic setting. By considering inputs perturbed by some amount of random noise, we can get a more accurate representation of how these algorithms will work in practice.

The games of special interest to our research are network coordination games. With networks appearing everywhere in the world, including nature, finance, and computer science, these games are of special interest to both theorists and pragmatists alike.

There has been significant prior work done on these games. Most recently, Boodaghi-

ans et al. [3] have shown that computing pure Nash equilibria in network coordination games (a PLS-complete problem under worst-case analysis) has a smoothed quasipolynomial runtime (in both n, the number of players, and k, the number of strategies each player has) when the network graph is complete. However, we suspect that a smoothed runtime independent of k is possible, as suggested by Boodaghians et al. [3] in their reduction of k-NetCoordNash to 2-Flip-Max-Cut.

Our work seeks to summarize our attempts made to bridge this gap. Specifically, we would like to show a smoothed quasipolynomial runtime for 2-Flip-Max-Cut where the graph is complete. This result would immediately imply that network coordination games for complete graphs have a smoothed efficient runtime. We have discussed using the same framework as Etscheid and Roglin [6], in which we get a set T such that $|T| \geq \Omega(\sqrt{D})$. Unfortunately, this set T is likely too small and will not provide an efficient runtime.

Another potential area of work is to consider redefining what a k-repeating sequence is. If we define these sequences such that D itself has the linear independence properties that we want *and* D is of some large enough size, then we can immediately conclude that a large enough set T exists since it can be D itself. While we expect that this will require some creative new analysis, we suspect that sequences may not need to be too large to guarantee a large enough set D that satisfies our desired constraints.

5.2 Remaining Questions

As stated in the section above, one of our key questions for future research is how we can build upon our two paths to show a smoothed quasipolynomial time for 2-Flip-Max-Cut.

There are also many other closely related problems that warrant further research. For one, it is of much interest to look beyond 2-Flip-Max-Cut into the more general problem discussed by Boodaghians et al. [3] of d-Flip-Max-Cut.

Another related problem of interested is of Max-k-Cut presented by Bibak [2].

Rather than consider a maximum cut across two partitions up to some number of flips, this problem considers the case where we have k different partitions, and we seek to maximize the sum of the edge weights across all of the partitions. Bibak [2] shows that Max-Cut (the problem where $k = 2$) can be solved in smoothed polynomial time, improving upon the result shown by Angel et al. [1], and also that Max-k-Cut yields a smoothed polynomial complexity for complete graphs and a smoothed quasipolynomial complexity for arbitrary graphs. We suspect it may be possible to leverage the smoothness-preserving reductions presented in Boodaghians et al. [3] to reduce k-NetCoordNash to Max-k-Cut.

Bibliography

[1] Omer Angel, Sébastien Bubeck, Yuval Peres, and Fan Wei. Local max-cut in smoothed polynomial time. *arXiv:1610.04807 [cs, math]*, April 2017. arXiv: 1610.04807.

[2] Ali Bibak, Charles Carlson, and Karthekeyan Chandrasekaran. Improving the smoothed complexity of FLIP for max cut problems. *arXiv:1807.05665 [cs]*, July 2018. arXiv: 1807.05665.

[3] Shant Boodaghians, Rucha Kulkarni, and Ruta Mehta. Smoothed Efficient Algorithms and Reductions for Network Coordination Games. *arXiv:1809.02280 [cs]*, February 2019. arXiv: 1809.02280.

[4] Xi Chen, Chenghao Guo, Emmanouil-Vasileios Vlatakis-Gkaragkounis, Mihalis Yannakakis, and Xinzhi Zhang. Smoothed complexity of local Max-Cut and binary Max-CSP. *arXiv:1911.10381 [cs]*, November 2019. arXiv: 1911.10381.

[5] Constantinos Daskalakis and Christos Papadimitriou. Continuous Local Search. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, January 2011.

[6] Michael Etscheid and Heiko Röglin. Smoothed Analysis of Local Search for the Maximum-Cut Problem.

[7] S Johnson. How Easy Is Local Search?

[8] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3), June 1994.

[9] Heiko Roeglin. The Complexity of Nash Equilibria, Local Optima, and Pareto-Optimal Solutions. April 2008.

[10] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *CoRR*, cs.DS/0111050, 2001.