The Complexity of Solution Concepts in Lipschitz Games



Matthew Katzman
Magdalen College
University of Oxford

Thesis submitted for the degree of
Doctor of Philosophy
Trinity 2022

Abstract

Nearly a decade ago, Azrieli and Shmaya introduced the class of λ-Lipschitz games in which every player's payoff function is λ-Lipschitz with respect to the actions of the other players. They showed via the probabilistic method that n-player Lipschitz games with m strategies per player have *pure* ϵ-approximate Nash equilibria, for $\epsilon \geq \lambda\sqrt{8n\log(2mn)}$. They left open, however, the question of how hard it is to find such an equilibrium. In this work, we develop an efficient reduction from more general games to Lipschitz games. We use this reduction to study both the query and computational complexity of algorithms finding ϵ-approximate *pure* Nash equilibria of λ-Lipschitz games and related classes.

We show a query lower bound exponential in $n\lambda/\epsilon$ against randomized algorithms finding ϵ-approximate pure Nash equilibria of n-player, λ-Lipschitz games. We additionally present the first **PPAD**-completeness result for finding pure Nash equilibria in a class of finite, non-Bayesian games (we show this for λ-Lipschitz polymatrix games for suitable pairs of values ϵ and λ) in which both the proof of **PPAD**-hardness and the proof of containment in **PPAD** require novel approaches (in fact, our approach implies containment in **PPAD** for any class of Lipschitz games in which payoffs from mixed-strategy profiles can be deterministically computed), and present a definition of "randomized **PPAD**". We define and subsequently analyze the class of "Multi-Lipschitz games", a generalization of Lipschitz games involving *player-specific* Lipschitz parameters in which the value of interest appears to be the *average* of the individual Lipschitz parameters. We discuss a dichotomy of the *deterministic* query complexity of finding ϵ-approximate Nash equilibria of general games and, subsequently, a query lower bound for λ-Lipschitz games in which any non-trivial value of ϵ requires exponentially-many queries to achieve. We examine which parts of this extend to the concepts of approximate correlated and coarse correlated equilibria, and in the process generalize the edge-isoperimetric inequalities to generalizations of the hypercube. Finally, we improve the block update algorithm presented by Goldberg and Marmolejo to break the potential boundary of a 0.75-approximation factor, presenting a randomized algorithm achieving a 0.7368-approximate Nash equilibrium making polynomially-many profile queries of an n-player $\frac{1}{n-1}$-Lipschitz game with an unbounded number of actions.

Acknowledgements

I would like to thank my supervisors Paul Goldberg and Rahul Santhanam for their years of guidance, my fellow students Ninad Rajgopal and Francisco J. Marmolejo Cossío for talking through countless ideas with me (and Alex Brandts for...less productive conversations), and my friends and family, especially my mother Sandy Katzman and wife Natalie Katzman, for their enduring support. I would also like to thank my examiners Elias Koutsoupias, Panagiotis Kanellopoulos, and Edith Elkind for their helpful feedback through every milestone, and Standa Živný for his teaching mentorship. This work was supported by an Oxford-DeepMind Graduate Scholarship and a Studentship provided by the Department of Computer Science.

Declarations

I certify that I have written this thesis by myself. Parts of the thesis have appeared in [42] and [43].

Contents

1 Introduction **1**

1.1 Game Theory Origins and Applications 1

1.2 Algorithmic Game Theory 2

1.3 Lipschitz Games 2

1.4 Query Complexity of Nash Equilibria 4

1.5 Computational Complexity of Nash Equilibria 4

1.6 Outline and Main Results 5

2 Preliminaries **7**

2.1 Definitions and Concepts 7

2.1.1 Game Theory Concepts 7

2.1.2 Complexity Theory Concepts 14

2.1.3 Problem Definitions 17

2.2 Literature Overview 19

2.2.1 General Games 19

2.2.2 Restricted Games 27

2.2.3 Best-Response Queries 34

3 Query Complexity **35**

3.1 Overview 35

3.2 Techniques 36

3.2.1 The Induced Population Game 36

3.2.2 Approximate Equilibria of Perturbed Games 38

3.3 Pure Equilibria 41

3.3.1 Warmup . . . 41
3.3.2 Main Result . . . 43
3.4 Multi-Lipschitz Games . . . 45
3.4.1 Warmup . . . 45
3.4.2 Generalized Result . . . 46

4 Time Complexity **49**
4.1 Overview . . . 49
4.1.1 A Comparison with the Previous Chapter . . . 50
4.2 **PPAD**-Completeness . . . 52
4.2.1 Containment in **PPAD** . . . 52
4.2.2 Hardness for **PPAD** . . . 65
4.3 Randomized **PPAD** . . . 66
4.3.1 Mapping Reduction Definition of **RPPAD** . . . 66
4.3.2 Turing Reduction Definition of **BPPPAD** . . . 67
4.3.3 Equivalence of Definitions . . . 68
4.3.4 Pure Nash Equilibria . . . 76

5 Upper Bounds **79**
5.1 Overview . . . 79
5.2 Preliminaries . . . 79
5.2.1 The Game Model . . . 80
5.2.2 Block Equilibrium . . . 80
5.3 Result Overview . . . 82
5.3.1 Outline . . . 82
5.3.2 The Algorithm . . . 82
5.4 Stage 2 . . . 84
5.4.1 The Worst-Case Regret . . . 84
5.4.2 The Worst-Case Best Response . . . 85
5.4.3 Bounding the Average Worst-Case Regret . . . 89
5.5 Stage 3 . . . 90
5.5.1 Approximations . . . 90

5.5.2 The Worst-Case Best-Response 90
5.5.3 Bounding the Worst-Case Regret 96
5.5.4 Calculating $\boldsymbol{\rho_1(\epsilon_0)}$ 97
5.5.5 Calculating $\boldsymbol{\rho_2(\epsilon_0)}$ 98
5.5.6 Computing $\boldsymbol{\rho_3(\epsilon_0)}$ 101
5.5.7 Justifying the Approximations 111

6 General Games **115**

6.1 Overview 115
6.2 A Nash Lower Bound 115
6.3 Correlated Equilibria 121
6.3.1 The Hyper-m-Clique 121
6.3.2 Correlated (and Coarse Correlated) Equilibria 126

7 Conclusion **135**

7.1 Open Questions 135
7.2 Final Thoughts 137

List of Figures

3.1 Approximate Nash Equilibria of Perturbed Coordination Game 40

4.1 `EndOfTheLine` instance to be queried three times. 70

4.2 The standard `EndOfTheLine` solution. 70

4.3 The `EndOfTheLine` solution returning an end of the alternate line for the first query. . 71

4.4 The `EndOfTheLine` solution returning an end of the alternate line for the second query. 71

4.5 The `EndOfTheLine` solution returning an end of the alternate line for the third query. 71

4.6 The `EndOfTheLine` solution returning the *start* of the alternate line for the first query and an end of the alternate line for the second query. 72

4.7 The `EndOfTheLine` solution returning the *end* of the alternate line for the first query and an end of the alternate line for the second query. 72

4.8 The `EndOfTheLine` solution returning the *start* of the alternate line for the first query and an end of the alternate line for the third query. 72

4.9 The `EndOfTheLine` solution returning the *end* of the alternate line for the first query and an end of the alternate line for the third query. 72

4.10 The `EndOfTheLine` solution returning the *start* of the alternate line for the second query and an end of the alternate line for the third query. 72

4.11 The `EndOfTheLine` solution returning the *end* of the alternate line for the second query and an end of the alternate line for the third query. 73

4.12 The `EndOfTheLine` solution returning the *start* of the alternate line for the first and second queries and an end of the alternate line for the third query. 73

4.13 The `EndOfTheLine` solution returning the *start* of the alternate line for the first query, the *end* of the alternate line for the second query, and an end of the alternate line for the third query. 73

4.14 The `EndOfTheLine` solution returning the *end* of the alternate line for the first query, the *start* of the alternate line for the second query, and an end of the alternate line for the third query. 73

4.15 The `EndOfTheLine` solution returning the *end* of the alternate line for the first and second queries and an end of the alternate line for the third query. 73

4.16 The remaining well-formed strings involving nodes on the loop. 73

5.1 Best Response Dynamics. There will be a number of figures of this form to follow. The blue line describes the payoff of the best-response as time goes forward. The orange line shows, for a time t, what the payoff of the action played at time t has dropped to by the time we are considering (here time $t = 0$). Finally, the green line at time t shows the regret of the action played at time t by the time we are considering. The total regret is thus the area under the green curve. 92

5.2 The Worst-Case Best Response for $0 \leq T' \leq 1 - v_i^{(\epsilon_0)}(0)$, where $v_i(0) > v_i^{(\epsilon_0)}(0)$ by the application of Lemma 5.10 (recall that $v_i^{(\epsilon_0)}(0)$ is merely a lower-bound for $v_i(0)$ rather than a true initial condition). 94

5.3 The Worst-Case Best Response for $1 - v_i^{(\epsilon_0)}(0) < T_2 < 1 + v_i^{(\epsilon_0)}(0)$ 95

5.4 The Worst-Case Best Response for $1 + v_i^{(\epsilon_0)}(0) \leq T_2$ 96

5.5 $\rho_1(\epsilon_0)$ Dynamics for $0 \leq T_2 \leq \min\{1 - v_i^{(\epsilon_0)}(0), v_i^{(\epsilon_0)}(0)\}$ 99

5.6 $\rho_1(\epsilon_0)$ Dynamics for $1 - v_i^{(\epsilon_0)}(0) < T_2 \leq v_i^{(\epsilon_0)}(0)$ 100

5.7 $\rho_2(\epsilon_0)$ Dynamics for $0 \leq T_2 \leq \min\{1 - v_i^{(\epsilon_0)}(0), v_i^{(\epsilon_0)}(0)\}$ 102

5.8 $\rho_2(\epsilon_0)$ Dynamics for $1 - v_i^{(\epsilon_0)}(0) < T_2 \leq v_i^{(\epsilon_0)}(0)$ 103

5.9 In order for ρ_3 to be nonnegative in Case 1 as required, T_2 must lie below this curve (for $\epsilon_0 > \frac{1}{2}$). Since $T_2 \leq 0.5$ by assumption, this is guaranteed to be the case. 104

5.10 $\rho_3(\epsilon_0)$ Dynamics for $0 \leq T_2 \leq \min\{1 - v_i^{(\epsilon_0)}(0), v_i^{(\epsilon_0)}(0)\}$ 105

5.11 In order for ρ_3 to be nonnegative in Case 2 as required, T_2 must lie below this curve (for $\epsilon_0 \in (\frac{1}{4}, \frac{3}{4}]$). Since $T_2 \leq 0.5$ by assumption, this is guaranteed to be the case. . . . 106

5.12 $\rho_3(\epsilon_0)$ Dynamics for $1 - v_i^{(\epsilon_0)}(0) < T' \leq v_i^{(\epsilon_0)}(0)$ 107

5.13 A contour plot of $r_i^{(v)}(T_2)$ from 0 (blue) to 1 (yellow) with T_2 on the x-axis and ϵ_0 on the y-axis. 108

5.14 Figure 5.13 zoomed in on the relevant area with a point denoting the best choice of T_2 for the adversarial choice of ϵ_0. 109

5.15 The graph of $\max_{\epsilon_0 \in [0,.75]} r_i^{(v)}(T_2)$ as a function of T_2. 110

5.16 The graph of $r_i^{(v)}(T_2)$ as a function of ϵ_0 when $T_2 = 0.057$ is selected. 110

5.17 Best Response, Payoff, and Regret Dynamics in the Discrete Case (left) and the Continuous Approximation (right) 112

6.1 The payoff matrices of $G_{1,2}$ and $G_{1,3}$. 117

6.2 The region of possible values for $(\mathbf{p}_1^*, \mathbf{p}_2^*)$ in any $\left(\frac{1}{2} - \alpha\right)$-approximate Nash equilibrium of $G_{k,2}$. The only exact Nash equilibrium is shown by the red point, and the corresponding values of ρ are displayed as the orange lines. 118

6.3 The 2-dimensional hyper-3-gon $H_{2,3}$, color-coded for clarity. The fill color represents the first dimension and the border color represents the second, with edges between points of the same fill but different border or points of the same border but different fill. 122

List of Algorithms

1 Polynomial Local Search Algorithm 17

2 Relevant Set Recurrence 60

3 Check Well-Formed Strings 74

4 Definition of Circuit S 75

5 Definition of Circuit P 76

6 Block Equilibrium Computation `BU` (performed by each player i) 81

7 The `BlockStep` algorithm 81

8 Extended Block Equilibrium Computation `EBU` (performed by each player i) 83

Chapter 1

Introduction

1.1 Game Theory Origins and Applications

The concepts of game theory appear in many aspects of life. [78] explains many evolutionary biological concepts through the lens of *evolutionary game theory*. Economists and businesses constantly employ game theoretic ideas in their everyday activities and transactions [77]. Traders and retailers set their prices and costs according to models they derive from game theory [59]. Popular poker strategy is moving further towards *game theoretic optimal* play every year [16]. Politicians make important high-pressure decisions by modeling the situation as a game [74]. More recently, many data scientists performing machine learning and artificial intelligence tasks are even relying heavily on game theory to successfully train and deploy their models [35].

Though many of the concepts now falling under the umbrella of Game Theory existed long before, the field exploded with the publication of the foundational work of John Nash [65]. While Nash did not himself formulate the concept of equilibrium in games, he showed the following, for which the name "Nash equilibrium" was applied to the type of solution Nash studied.

Theorem 1.1 ([65])**.** *Every finite, non-cooperative game has a (possibly mixed) Nash equilibrium.*

Nash showed this through a reduction to the existence of Brouwer fixed points. Since the publication of this result, a primary focus of researchers and practitioners alike has been the search for and classification of such equilibria and related points. While there is a large body of work focused on more general types of equilibria (and a large number of open questions in this area as well), this work is focused primarily on Nash equilibria and their approximations.

1.2 Algorithmic Game Theory

While the claim that every game has an equilibrium point is important and interesting, it is not of much practical use if such a point cannot be found. For example, [65] provides an example of a simple three-player poker game which possesses only a single equilibrium point in which players must place irrational probabilities on each of their strategies. This shows that the solution space is uncountably large - can the problem be solved in general?

These questions are a subset of those that led to the formation of the field of *Algorithmic Game Theory*. This field is often broken down further into three subfields: the design of games exhibiting certain properties, the analysis of the dynamics and equilibria of such games, and the computation of such equilibria [66]. It is, again, primarily the latter which is the focus of this work.

Of course, there are many quantities of interest when considering the tractability of the search for equilibria, each associated with a specific resource. The standard notion of time complexity is a heavily-studied consideration for this problem. However, games presented in normal form possess the unusual quality that even the length of their description can be exponential in the number of players. As such, query complexity is an equally important consideration, and will be addressed throughout this work as well.

1.3 Lipschitz Games

A Lipschitz game is a multi-player game in which there is an additive limit λ (called the Lipschitz constant of the game) on how much any player's payoffs can change due to a deviation by any other player. Thus, any player's payoff function is λ-Lipschitz continuous as a function of the other players' mixed strategies. Such games are also often called 'large' games, as they most frequently arise in natural settings with a large number of players, such as financial markets and various types of network games.

Lipschitz games were introduced about ten years ago by Azrieli and Shmaya [2]. They exhibit many fascinating properties, such as various metrics of fault tolerance (explored in [50]) and ex-post stability of equilibria (studied in [30]). A key feature of Lipschitz games, and the one we primarily address in this work, is that they are guaranteed to have approximate Nash equilibria *in pure strategies*, where the quality of the approximation depends on the number of players n, the number of actions m, and the Lipschitz constant λ. In particular, [2] showed that this guarantee holds (keeping the number of

actions constant) for Lipschitz constants of size $o(1/\sqrt{n \log n})$ (existence of pure approximate equilibria is trivial for Lipschitz constants of size $o(1/n)$ since then, players have such low effect on each others' payoffs that they can best-respond independently to get a pure approximate equilibrium). The general idea of the existence proof is to take a mixed Nash equilibrium (guaranteed to exist by Nash's theorem [65]), and prove that there is a positive probability that a pure profile sampled from it will constitute an approximate equilibrium.

As noted in [2] (and reiterated frequently), solutions in pure-strategy profiles are a more plausible and satisfying model of a game's outcome than solutions in mixed-strategy profiles. On the other hand, the existence guarantee raises the question of how to *compute* an approximate equilibrium. In contrast with potential games, in which pure-strategy equilibria can often be found via best- and better-response dynamics [66], there is no obvious natural approach in the context of Lipschitz games, despite the existence guarantee. The general algorithmic question addressed in this work is:

> *Given a Lipschitz game, how hard is it to find a pure-strategy profile that constitutes an approximate equilibrium?*

Recent work [41] has identified algorithms achieving additive constant approximation guarantees, but as noted by Babichenko [7], the extent to which we can achieve the pure approximate equilibria that are guaranteed by [2] (or alternatively, potential lower bounds on query or computational complexity) is unknown.

Variants and special cases of this question include classes of Lipschitz games having a concise representation (analyzed in Chapter 4), as opposed to unrestricted Lipschitz games for which an algorithm has query access to the payoff function (explored in Chapter 3). In the latter case, the question subdivides into what we can say about the query complexity, and about the computational complexity (for concisely-represented games the query complexity is low, by Theorem 3.3 of [45]). Moreover, if equilibria can be easily computed, can they be found easily via some kind of natural-looking decentralized process? Positive results for these questions help us to believe in "approximate pure Nash equilibrium" as a solution concept for Lipschitz games. Alternatively, it is of interest to identify computational obstacles to the search for a Nash equilibrium. Both upper and lower bounds for these questions are discussed below.

In fact, positive algorithmic results have been found for classes of games that combine the Lipschitz property with others, such as *anonymous* games [28] and *aggregative* games [6]. For anonymous games (in which each player's payoffs depend only on the *number* of other players playing each action, and not

which players), Daskalakis and Papadimitriou [28] improved upon the upper bound of [2] to guarantee the existence of ϵ-approximate pure Nash equilibria for $\epsilon = O(\lambda)$ (the only dependence on n coming from λ itself), presenting a polynomial-time algorithm to find one. Peretz et al. [69] analyze Lipschitz values that result from δ-perturbing anonymous games, in the sense that every player is assumed to randomize uniformly with probability δ. Goldberg and Turchetta [46] showed that a 3λ-approximate pure Nash equilibrium of a λ-Lipschitz anonymous game can be found querying $O(n \log n)$ individual payoffs. These results are described in more detail in Chapter 2.

1.4 Query Complexity of Nash Equilibria

In general, lower bounds on the query complexity for finding Nash equilibria are known that are exponential in n (which motivates a focus on subclasses of games, such as Lipchitz games, and others). Hart and Mansour [52] showed that the deterministic communication (and thus query) complexity of computing an exact Nash equilibrium (pure or mixed) in a game with n players is $2^{\Omega(n)}$. Subsequent results have iteratively strengthened this lower bound. First, Babichenko [5] showed an exponential lower bound on the *randomized* query complexity of computing an ϵ-*well supported* Nash equilibrium (an approximate equilibrium in which every action in the support of a given player's mixed strategy is an ϵ-best response) for a constant value of ϵ, even when considering δ-distributional query complexity, as defined in Definition 2.1.12. Shortly after, Chen, Cheng, and Tang [17] showed a $2^{\Omega(n/\log n)}$ lower bound on the randomized query complexity of computing an (not necessarily well-supported) ϵ-approximate Nash equilibrium for a constant value of ϵ, which Rubinstein [72] improved to a $2^{\Omega(n)}$ lower bound, even allowing a constant fraction of players to experience regret greater than ϵ (taking regret as defined in Definition 2.1.4). These intractability results motivate us to consider a restricted class of games in Chapter 3 (Lipschitz games) which contain significantly more structure than do general games.

1.5 Computational Complexity of Nash Equilibria

A key feature of the problem of computing a mixed Nash equilibrium (either an exact one, or an ϵ-approximate one for some $\epsilon > 0$) is that due to the guaranteed existence [65] of a solution that once found can be easily checked, it cannot be **NP**-hard unless **NP** = **coNP** [61]. Beginning with [24, 21], the complexity class **PPAD** (defined below) emerged as the means of identifying whether a

class of games is likely to be hard to solve. Until recently, **PPAD**-completeness has mostly been used to identify computational obstacles to finding mixed equilibria rather than pure ones.

In Chapter 4, we present the first known **PPAD**-completeness result for a class of finite, non-Bayesian games. In the setting of Bayes-Nash equilibrium, Filos-Ratsikas et al. [39] obtain a **PPAD**-completeness result for pure-strategy solutions of a class of first-price auctions. Closer to the results we present in Chapter 4 is a **PPAD**-completeness result of Papadimitriou and Peng [68] for a class of public goods games with divisible goods, where the continuous action-space of the players can represent probabilities in the solution of a corresponding 2-player game.

1.6 Outline and Main Results

This work is composed of the following sections.

- Chapter 2 provides the important definitions and concepts used throughout the work, and places the work within its larger context.
- Chapter 3 presents some of the query lower bound results published in the 14th Symposium on Algorithmic Game Theory [42]. The main result of this chapter is an exponential lower bound on the query complexity of finding approximate pure Nash equilibria in Lipschitz games. This chapter also introduces some of the critical lemmas and techniques discussed throughout the remainder of the work.
- Chapter 4 presents some of the time complexity lower bound results published in the 15th Symposium on Algorithmic Game Theory [43]. It provides the first **PPAD**-completeness result for finding pure equilibria in a class of finite, non-Bayesian games. Specifically, for certain functions λ and ϵ, it is **PPAD**-complete to find ϵ-approximate pure equilibria of λ-Lipschitz polymatrix games. At the end of this chapter, we define a randomized analogue to **PPAD** and uncover a few interesting properties.
- Chapter 5 improves upon the algorithms presented by [41] to uncover a stronger upper bound for finding pure Nash equilibria in n-player, m-action games with a Lipschitz constant of $\frac{1}{n}$.
- Chapter 6 presents results on the deterministic query complexity of general finite games that are not necessarily Lipschitz. Some of these results generalize those of [53].

Chapter 2

Preliminaries

2.1 Definitions and Concepts

2.1.1 Game Theory Concepts

Throughout, we use the following notation.

- Boldface capital letters denote matrices, and boldface lowercase letters denote vectors.
- The symbol $\mathbf{a}$ is used to denote a pure action profile, and $\mathbf{p}$ is used when the strategy profiles may be mixed. Furthermore, $\mathbf{X}$ is used to denote *correlated* strategies.
- $[n]$ and $[m]$ denote the sets $\{1, \ldots, n\}$ of players and $\{1, \ldots, m\}$ of actions, respectively. Furthermore, $i \in [n]$ will always refer to a player, and $j \in [m]$ will always refer to an action.
- Whenever a query returns an approximate answer, the payoff vector $\tilde{\mathbf{u}}$ will be used to represent the approximation and $\mathbf{u}$ will represent the true value.

The Game Model

We introduce standard concepts of strategy profiles, payoffs, regret, and equilibria for pure, mixed, and correlated strategies.

Definition 2.1.1 (Strategy Profiles)**.** We define the following types of strategy profiles.

- A *pure* action profile $\mathbf{a} = (a_1, \ldots, a_n) \in [m]^n$ is an assignment of one action to each player. We use

$$\mathbf{a}_{-i} = (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n) \in [m]^{n-1}$$

to denote the set of actions played by players in $[n] \setminus \{i\}$.

- A (possibly *mixed*) strategy profile $\mathbf{p} = (p_1, \ldots, p_n) \in (\Delta[m])^n$ (where $\Delta(S)$ is the probability simplex over S) is a collection of n independent probability distributions, each taken over the action set of a player, where p_{ij} is the probability with which player i plays action j. The set of distributions for players in $[n] \setminus \{i\}$ is denoted

$$\mathbf{p}_{-i} = (p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n).$$

When $\mathbf{p}$ contains just 0-1 values, $\mathbf{p}$ is equivalent to some action profile $\mathbf{a} \in [m]^n$.

Furthermore, when considering binary-action games with action set $\{1, 2\}$, we instead describe strategy profiles by vectors $\mathbf{p} = (p_1, \ldots, p_n)$, where p_i is the probability that player i plays action 1.

- A *correlated* strategy profile $\mathbf{X} \in \Delta([m]^n)$ is a single joint probability distribution taken over the space of all pure action profiles $\mathbf{a}$.

Definition 2.1.2 (Payoffs)**.** Given player i, action j, and pure action profile $\mathbf{a}$,

- $u_i(j, \mathbf{a}_{-i})$ is the payoff that player i obtains for playing action j when all other players play the actions given in $\mathbf{a}_{-i}$.

- $u_i(\mathbf{a}) = u_i(a_i, \mathbf{a}_{-i})$ is the payoff that player i obtains when all players play the actions given in $\mathbf{a}$.

- Similarly for mixed-strategy profiles:

$$u_i(j, \mathbf{p}_{-i}) = \mathbb{E}_{\mathbf{a}_{-i} \sim \mathbf{p}_{-i}}[u_i(j, \mathbf{a}_{-i})] \qquad u_i(\mathbf{p}) = \mathbb{E}_{\mathbf{a} \sim \mathbf{p}}[u_i(\mathbf{a})].$$

- For a given player $i \in [n]$, consider a deviation function $\phi : [m] \to [m]$. Then, similarly,

$$u_i^{(\phi)}(\mathbf{X}) = \mathbb{E}_{\mathbf{a} \sim \mathbf{X}}[u_i(\phi(a_i), \mathbf{a}_{-i})], \qquad u_i(\mathbf{X}) = \mathbb{E}_{\mathbf{a} \sim \mathbf{X}}[u_i(\mathbf{a})].$$

Furthermore, given an event E, $u_i(\mathbf{X} \mid E) = \mathbb{E}_{\mathbf{a}\sim\mathbf{X}}[u_i(\mathbf{a}) \mid E]$.

Games are uniquely specified by the payoff functions for each player.

Definition 2.1.3 (Best Response)**.** Given a player i and a strategy profile $\mathbf{p}$, define the *best response*

$$\mathrm{br}_i(\mathbf{p}) = \arg\max_{j\in[m]} u_i(j, \mathbf{p}_{-i}).$$

In addition, for $\epsilon > 0$, any action j' satisfying

$$\mathrm{br}_i(\mathbf{p}) - u_i(j', \mathbf{p}_{-i}) \leq \epsilon$$

is labelled an *ϵ-best response*.

Definition 2.1.4 (Regret)**.** We define regret for each type of profile:

- Given a player i and a strategy profile $\mathbf{p}$, define the regret

$$\mathrm{reg}_i(\mathbf{p}) = \max_{j\in[m]} u_i(j, \mathbf{p}_{-i}) - u_i(\mathbf{p})$$

  to be the difference between the payoffs of player i's best response to $\mathbf{p}_{-i}$ and i's strategy p_i.

- Given a player i and a correlated strategy profile $\mathbf{X}$, define

$$\mathrm{reg}_i^{(\phi)}(\mathbf{X}) = u_i^{(\phi)}(\mathbf{X}) - u_i(\mathbf{X}), \qquad \mathrm{reg}_i(\mathbf{X}) = \max_{\phi:[m]\to[m]} \mathrm{reg}_i^{(\phi)}(\mathbf{X}),$$

  the regret $\mathrm{reg}_i(\mathbf{X})$ being the difference between the payoffs of player i's best deviation from $\mathbf{X}$, and $\mathbf{X}$.

Definition 2.1.5 (Equilibria)**.** We introduce a number of different types of equilibrium:

- An *ϵ-approximate Nash equilibrium* (ϵ-ANE) is a strategy profile $\mathbf{p}^*$ such that, for every player $i \in [n]$, $\mathrm{reg}_i(\mathbf{p}^*) \leq \epsilon$.

- An *ϵ-well supported Nash equilibrium* (ϵ-WSNE) is an ϵ-ANE $\mathbf{p}^*$ for which every action j in the support of p_i^* is an ϵ-best response to $\mathbf{p}_{-i}^*$.

- An *ϵ-approximate pure Nash equilibrium* (ϵ-PNE) is a pure action profile $\mathbf{a}$ such that, for every player $i \in [n]$, $\mathrm{reg}_i(\mathbf{a}) \leq \epsilon$.

- An ϵ-*approximate correlated equilibrium* (ϵ-ACE) is a correlated strategy profile $\mathbf{X}^*$ such that, for every player $i \in [n]$, $\text{reg}_i(\mathbf{X}^*) \leq \epsilon$.

- Finally, an ϵ-*approximate coarse correlated equilibrium* (ϵ-CCE) is a correlated strategy profile $\mathbf{X}^*$ such that, for every player $i \in [n]$, $\max_m \text{reg}_i^{(m)}(\mathbf{X}) \leq \epsilon$ where $\text{reg}_i^{(m)}$ is the regret player i experiences for playing $\mathbf{X}$ rather than the unilateral deviation of playing action m.

Note that any ϵ-PNE is an ϵ-WSNE, and any ϵ-ANE constitutes an ϵ-ACE, which in turn is an ϵ-CCE. Consequently, any algorithmic lower bounds on correlated equilibria also apply to Nash equilibria.

With these definitions in mind we present two games below that will be mentioned later in this work.

Definition 2.1.6 (Matching Pennies)**.** The Matching Pennies game is a 2-player, binary-action game (with actions H and T for each player) defined by the following payoffs (where Player 1's payoffs are listed first, followed by Player 2's):

		Player 2	
		H	T
Player 1	H	$1, 0$	$0, 1$
	T	$0, 1$	$1, 0$

(i.e. Player 1 wins if the players match and Player 2 wins if the players mismatch).

The Matching Pennies game has only one exact Nash equilibrium - both players must randomize between H and T with 50% probability of each.

Definition 2.1.7 (Coordination Game)**.** The Coordination Game is a 2-player, binary-action game

defined by the following payoffs:

		Player 2: 1	Player 2: 2
Player 1	1	1, 1	0, 0
	2	0, 0	1, 1

(i.e. the players both win if they match and both lose if they mismatch).

The Coordination Game has three exact Nash equilibria - in addition to the 50/50 equilibrium it shares with Matching Pennies, the pure profiles in which both players play 1 or both players play 2 are also equilibria.

Finally, to end this section, we introduce the class of games that is the focus of this work. Lipschitz games were initially studied by Azrieli and Shmaya [2].

We first reproduce the definition provided in [2]:

Definition 2.1.8 (Lipschitz Games [2])**.** A game is λ-*Lipschitz* if, for every player $i \in [n]$, all action profiles $\mathbf{a}, \mathbf{a}'$ such that

$$||\mathbf{a}_{-i} - \mathbf{a}'_{-i}||_0 = 1$$

satisfy

$$|u_i(a_i, \mathbf{a}_{-i}) - u_i(a_i, \mathbf{a}'_{-i})| \leq \lambda.$$

This is the definition we will use, but we will frame it slightly differently. We will first define binary-action Lipschitz games, before extending the definition to more actions.

Definition 2.1.9 (Lipschitz Games)**.** For any value $\lambda \in (0, 1]$, a binary-action λ-*Lipschitz game* is a game in which a change in strategy of any given player can affect the payoffs of any other player by at most an additive λ, i.e. for every player i and pair of action profiles $\mathbf{a}, \mathbf{a}'$ with $a_i = a'_i$,

$$|u_i(\mathbf{a}) - u_i(\mathbf{a}')| \leq \lambda ||\mathbf{a}_{-i} - \mathbf{a}'_{-i}||_0$$

(equivalently, the ℓ_1 norm can be used, and is more relevant when it comes to mixed strategies). Note that, under the definition of payoffs to mixed strategies presented above, because mixed strategies of

binary-action games are still represented as vectors (of scalars), this can be easily extended to mixed profiles $\mathbf{p}, \mathbf{p}'$ with $p_i = p'_i$:

$$|u_i(\mathbf{p}) - u_i(\mathbf{p}')| \leq \lambda ||\mathbf{p}_{-i} - \mathbf{p}'_{-i}||_1.$$

The above definition, while presented in a different way, is mathematically equivalent to that given by [2] (just provided here in a form more relevant to the current work).

In order to extend the definition above to $m > 2$ actions, we will require a measure of distance for probability distributions:

Definition 2.1.10 (Total Variation Distance)**.** The *total variation distance* between distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over discrete sample space $\mathcal{X}$ (each represented by m-dimensional vectors) is

$$d_{\mathrm{TV}}(\mathcal{D}_1, \mathcal{D}_2) = \frac{1}{2}||\mathcal{D}_1 - \mathcal{D}_2||_1 = \frac{1}{2}\sum_{x \in \mathcal{X}} \left| \Pr_{x' \sim \mathcal{D}_1}(x' = x) - \Pr_{x' \sim \mathcal{D}_2}(x' = x) \right|.$$

Definition 2.1.11. For $m > 2$, the pure action definition remains the same as in the $m = 2$ case. However, now that mixed strategies are represented by vectors of distributions, the Lipschitz property becomes

$$|u_i(\mathbf{p}) - u_i(\mathbf{p}')| \leq \lambda \sum_{i'=1}^{n} d_{\mathrm{TV}}(p_{i'}, p'_{i'})$$

where d_{TV} is the total variation distance between two distributions (i.e. an extension of the ℓ_1 norm definition above).

Remark 1. We will rely on the total variation distance implicitly throughout the remainder of this work. It follows from Definition 2.1.11 that when a player i moves probability ρ from one action to another in a mixed strategy profile, every other player's payoffs may be affected by at most an additive $\lambda\rho$. In addition, because the payoff function u_i is λ-Lipschitz in this sense, the regret function reg_i is 2λ-Lipschitz in the same way, as the payoff of player i's best response can increase with a slope of at most λ while the payoff of player i's current strategy can decrease with a slope of at most λ.

The set of n-player, m-action, λ-Lipschitz games will be denoted $\mathcal{G}(n, m, \lambda)$.

The Query Model

This section introduces the model of queries we consider.

Definition 2.1.12 (Queries)**.** We will rely on the following different types of queries.

- A *profile query* of a pure action profile $\mathbf{a}$ of a game G, denoted $\mathcal{Q}^G(\mathbf{a})$, returns a vector $\mathbf{u}$ of payoffs $u_i(\mathbf{a})$ for each player $i \in [n]$.[1]

- A *δ-distribution query* of a strategy profile $\mathbf{p}$ of a game G, denoted $\mathcal{Q}^G_\delta(\mathbf{p})$, returns a vector $\tilde{\mathbf{u}}$ of n values such that $||\tilde{\mathbf{u}} - \mathbf{u}||_\infty \leq \delta$, where $\mathbf{u}$ is the players' expected utilities from $\mathbf{p}$. We also define a *(δ, γ)-distribution query* to be a δ-distribution query of a strategy profile $\mathbf{p}$ in which every action j in the support of p_i is allocated probability at least γ for every player $i \in [n]$.

- The *(profile) query complexity* of an algorithm A on input game G is the number of calls A makes to $\mathcal{Q}^G$. The *δ-distribution query complexity* of A is the number of calls A makes to $\mathcal{Q}^G_\delta$.

Babichenko [5] points out that it is uninteresting to consider 0-distribution queries, as any game in which every payoff is a multiple of $\frac{1}{M}$ for some $M \in \mathbb{N}$ can be completely learned by a single 0-distribution query. On the other hand, additive approximations to the expected payoffs can be computed via sampling from $\mathbf{p}$. Indeed, for general binary-action games we have from [45]:

Theorem 2.1 ([45])**.** *Take $G \in \mathcal{G}(n, 2, 1), \eta > 0$. Any (δ, γ)-distribution query of G can be simulated with probability at least $1 - \eta$ by*

$$\max\left\{\frac{1}{\gamma\delta^2}\log\left(\frac{8n}{\eta}\right), \frac{8}{\gamma}\log\left(\frac{4n}{\eta}\right)\right\}$$

profile queries.

Corollary 2.2. *Take $G \in \mathcal{G}(n, 2, 1), \eta > 0$. Any (δ, γ)-distribution query of G can be simulated with probability at least $1 - \eta$ by*

$$\frac{8}{\gamma\delta^2}\log\left(\frac{8n}{\eta}\right)$$

profile queries. Furthermore, any algorithm making q (δ, γ)-distribution queries of G can be simulated with probability at least $1 - \eta$ by

$$\frac{8q}{\gamma\delta^2}\log\left(\frac{8nq}{\eta}\right) = \text{poly}\left(n, \frac{1}{\gamma}, \frac{1}{\delta}, \log\frac{1}{\eta}\right) \cdot q\log q$$

profile queries.

[1] One might point out that, for large games, this is an unrealistically powerful model of query. However, since our results are lower bounds, they apply to any less powerful model as well.

Proof. The first claim is a weaker but simpler version of the upper bound of Theorem 2.1. The second claim follows from the first by a union bound. □

[41] are able to obtain the following corollary for m-action games:

Corollary 2.3 ([41])**.** *Any δ-distribution query of an n-player, m-action game can be simulated with probability at least $1-\eta$ by*

$$\frac{64m^2}{\delta^3}\log\left(\frac{8n}{\eta}\right)$$

profile queries.

Should equilibria of such games be easy to find, this would provide a starting point in determining at what point finding equilibria changes from easy to hard. However, since they are hard to find (as we show below), it indicates the need for an even more pessimistic view of the landscape of games that might be efficiently solved.

2.1.2 Complexity Theory Concepts

Definition 2.1.13. A *decision problem* associates with each possible input x an answer in $\{0,1\}$. A *search problem*, or *function problem*, is instead a binary relation R on inputs x and outputs y that encodes the statement "y is a solution to x". A *total function problem* is one for which every x admits some y such that $(x,y)\in R$.

Definition 2.1.14 (**P** and **NP**)**.** The class of problems **P** consists of decision problems L for which some algorithm A exists taking input x and returning the correct output in time polynomial in the length of the input.

The class of problems **NP** consists of decision problems L for which some polynomial-time algorithm A (a verifier) exists taking as input x and a certificate y of polynomial length such that for every string x

$$\exists y(A(x,y)=1)\Leftrightarrow x\in L.$$

The class of problems **coNP** consists of decision problems L for which some polynomial-time algorithm A exists taking as input x and a certificate y of polynomial length such that for every string x

$$\forall y(A(x,y)=1)\Leftrightarrow x\in L.$$

In other words, $\mathbf{coNP} = \{L \mid \bar{L} \in \mathbf{NP}\}$.

Remark 2. Clearly $\mathbf{P} \subseteq \mathbf{NP} \cap \mathbf{coNP}$, though the inclusion is not known to be strict and it is generally believed that **NP** and **coNP** are incomparable.

Definition 2.1.15 (**NP**-hard)**.** A problem L is **NP**-hard if, for every language $L' \in \mathbf{NP}$ there is a polynomial-time computable function f such that for every string x

$$x \in L' \Leftrightarrow f(x) \in L.$$

In other words, L is **NP**-hard if it is "at least as hard" as every problem in **NP**.

A problem L is **NP**-complete if L is **NP**-hard and $L \in \mathbf{NP}$.

Remark 3. The above definitions of "hard" and "complete" apply to all complexity classes considered in this work.

Definition 2.1.16 (**BPP**)**.** **BPP** is the class of decision problems L for which there exists a polynomial-time *randomized* algorithm which, for any given input, outputs the correct answer with probability $\geq \frac{2}{3}$.

Definition 2.1.17 (***#P***)**.** ***#P*** is the class of search problems $f(x)$ for which there exists an **NP** verifier A such that there are exactly $f(x)$ certificates y for which $A(x, y) = 1$.

While ***#P*** is the class of problems asking "how many solutions?", we can also define **FP** and **FNP** which are instead made up of problems asking "find a solution".

Definition 2.1.18 (**FP** and **FNP**)**.** The class of problems **FP** consists of binary relations R for which some polynomial-time algorithm A exists taking input x and returning some y of polynomial length such that $R(x, y)$ if such y exists and $\perp$ otherwise.

The class of problems **FNP** consists of binary relations R for which some polynomial-time algorithm A exists taking input x and y (of polynomial length) and calculating $R(x, y)$.

Remark 4. Clearly $\mathbf{FP} \subseteq \mathbf{FNP}$, though the inclusion is not known to be strict.

Definition 2.1.19 (**TFNP**)**.** The class **TFNP** (Total Function **NP**) is defined to be the set of binary relations R for which there is a deterministic polynomial-time algorithm V deciding $R(x, y)$ and a polynomial $p(n)$ such that

$$\forall x \in \{0,1\}^* \exists y \in \{0,1\}^{p(|x|)} V(x, y) = 1.$$

Remark 5. **TFNP** (equivalently defined as **F(NP ∩ co-NP)** [61]) is strongly believed not to have complete problems. Furthermore, if any **NP**-complete problem lies in **TFNP**, then **NP** = **coNP**.

The classes **PPAD** and **PLS** mentioned in this work are subclasses of **TFNP**. Their definitions are included below. First, **PPAD** is defined by a problem that is complete for the class, the `EndOfTheLine` problem.

Definition 2.1.20 (`EndOfTheLine`)**.** Define `EndOfTheLine` to be the problem taking as input a graph $G = (V, E)$, where the vertex set V is the exponentially-large set $\{0,1\}^n$, and E is encoded by two circuits P and S, where G contains directed edge (u, v) if and only if $S(u) = v$ and $P(v) = u$. Every vertex has both in-degree and out-degree at most one, and $0^n \in V$ has indegree 0, outdegree 1. The problem is to output another vertex v with total degree 1.

Definition 2.1.21 (**PPAD**)**. PPAD** is the set of problems many-one reducible to `EndOfTheLine` in polynomial time:

$$L \in \mathbf{PPAD} \Leftrightarrow L \leq_m^p \texttt{EndOfTheLine}.$$

Remark 6. Note that Definition 2.1.21 implies that `EndOfTheLine` is, in fact, complete for **PPAD**.

The class **PLS** was first defined in [57]. Although our results pertain to **PPAD**, we provide the definition of **PLS** below for comparison.

Definition 2.1.22 (**PLS**)**.** The class **PLS** consists of total search problems L such that, for every instance x, there is

- a finite set $F_L(x)$ of solutions of polynomially bounded size. Every $s \in F_L(x)$ is associated with
  - a non-negative integer cost $c_L(s, x)$.
  - a subset $N_L(s, x) \subseteq F_L(x)$ of neighbors of s.
- a polynomial time algorithm A_L which, given some instance x, outputs a value $A_L(x) \in F_L(x)$.
- a polynomial time algorithm B_L which, given some instance x and string s, decides if $s \in F_L(x)$ and, if so, outputs $c_L(s, x)$.
- a polynomial time algorithm C_L which, given some instance x and string $s \in F_L(x)$, outputs a string $s' \in N_L(x)$ such that $c_L(s', x) > c_L(s, x)$, or that no such string exists.

Remark 7. Note that membership in **PLS** implies that the following algorithm will return a locally optimal solution to x (because the $F_L(x)$ is finite and C_L can never output the same solution twice) but is not guaranteed to run in polynomial time:

Algorithm 1: Polynomial Local Search Algorithm

```
s = A_L(x)
while True:
    s' = C_L(s, x)
    if s' == Terminate:
        return s
    s = s'
```

Remark 8. C_L itself also acts as a verifier, so $L \in$ **TFNP**. So the inclusions (none of which are known to be strict) look like **FP** $\subseteq$ **PLS** $\subseteq$ **TFNP** $\subseteq$ **FNP**.

There is oracle-based evidence that the classes **PPAD** and **PLS** are incomparable [63], so historically the former is considered to embody the hardness of finding mixed equilibria while the latter is associated with pure equilibria in settings that guarantee their existence. One of the included results challenges this paradigm using the non-potential based nature of the pure equilibrium guarantee of [2].

2.1.3 Problem Definitions

We define below many of the problems used in this work.

Definition 2.1.23. Define (m, ϵ, λ)-PureLPG to be the problem of finding ϵ-PNEs of n-player, m-action, λ-Lipschitz polymatrix games, or alternatively finding a witness that the game is not λ-Lipschitz (note once more that m, ϵ, and λ are parameters of the problem while n is provided as input). Furthermore, define (m, ϵ, λ)-MixedLPG to be the equivalent problem for mixed equilibria.

Remark 9. Note that, while we will often fix the number of actions m, both ϵ and λ are often functions of the number of players n. We generally think of these as being decreasing functions of n.

We are interested in the complexity of the problem, for various pairs of functions $\epsilon(n)$, $\lambda(n)$.

Remark 10. One basic observation we make is that, for any $a \in (0, 1)$, (m, ϵ, λ)-PureLPG reduces to $(m, a\epsilon, a\lambda)$-PureLPG, by rescaling payoffs.

We now define a number of **PPAD**-complete problems that will be mentioned throughout this work.

Definition 2.1.24 (`Brouwer`)**.** Define `Brouwer` to be the problem taking as input a circuit C_F that computes the function $F : [0,1]^m \to [0,1]^m$, the Lipschitz parameter λ of F, and a rational constant $\epsilon > 0$ and outputting an ϵ-fixed point of F.

Lemma 2.4 ([67])**.** `Brouwer` *is* **PPAD**-*complete.*

Definition 2.1.25 (`nNash`)**.** Fix some $n \in \mathbb{N}$. Define `nNash` to be the problem taking as input the description (i.e. the explicit payoffs of every pure strategy profile) of an n-player, m-action game G and a rational constant $\epsilon > 0$ and outputting an ϵ-ANE of G.

[25] and [67] utilize Lemma 2.4 to show:

Lemma 2.5 ([25, 67])**.** *For $n \geq 4$,* `nNash` *is* **PPAD**-*complete.*

Definition 2.1.26 (d-Graphical Game)**.** A d-Graphical game is an undirected degree-d graph $G = (V, E)$ where each vertex v represents a player i whose payoffs depend only on the actions of i and the players represented by the neighbors of v.

Remark 11. As opposed to a general game, which requires a description length of nm^n, d-Graphical games belong to a subset of games which can be concisely represented using only nm^{d+1} explicit payoffs. For constant d and m, this takes the description length from exponential in the number of players to linear in the number of players.

Definition 2.1.27 (`(d,m,`ϵ`)-GraphicalNash`)**.** Define `(d,m,`ϵ`)-GraphicalNash` to be the problem taking as input an m-action d-Graphical game G and outputting an ϵ-ANE of G.

Lemma 2.6 ([44])**.** *Even for $m = 2, d = 3$, there exists some $\epsilon > 0$ such that* `(d,m,`ϵ`)-GraphicalNash` *is* **PPAD**-*complete.*

This result forms the foundation of a line of work culminating in Theorem 4.7 below (that (m, ϵ, λ)-MixedLPG is hard for nontrivial settings of ϵ and λ).

2.2 Literature Overview

2.2.1 General Games

Computational Complexity

It is uninteresting to consider the decision problem "does this game have a Nash equilibrium?". Theorem 1.1 shows that the answer to this question is always yes. However, there are a few decision problems that are still of interest.

Theorem 2.7 ([22])**.** *The question "how many Nash equilibria does this game have?" is* $\#\boldsymbol{P}$*-hard. Furthermore, even in cases as simple as* 2*-player symmetric games, it is* **NP***-hard (as the number of actions available to each player grows) to determine whether there exists a Nash equilibrium...*

- *...with expected total utility* $\geq k$*.*
- *...in which each player has expected utility* $\geq k$*.*
- *...that is Pareto-optimal (i.e. no other distribution exists in which* every *player receives at least the same payoff* and at least one player *receives a* strictly higher *payoff).*
- *...in which a given player* i *has expected utility* $\geq k$*.*

Additionally,

Theorem 2.8 ([8])**.** *Even in cases as simple as* 2*-player zero-sum games, it is* **NP***-hard under Turing reductions to determine the* sparsest *Nash equilibrium (i.e. the equilibrium with the smallest support).*

With many of these interesting questions resolved, the problem of *finding* a Nash equilibrium becomes of primary focus.

Definition 2.2.1 (`2Nash`)**.** `2Nash` is the parametrization of `nNash` to normal-form games of two players (also called *bimatrix* games).

We now define a related problem which, despite appearances, is not the same as `nNash`.

Definition 2.2.2 (ϵ`-Exact-nNash`)**.** The search problem ϵ`-Exact-nNash` takes as input a general normal-form game G in the form of a payoff tensor. ϵ`-Exact-nNash` outputs a strategy profile $\mathbf{p}'$ such that there exists a Nash equilibrium $\mathbf{p}^*$ with $||\mathbf{p}' - \mathbf{p}^*||_\infty \leq \epsilon$. Call this an ϵ-near approximate Nash equilibrium.

Remark 12. As it turns out, ϵ-approximate Nash equilibria can be far from all Nash equilibria, so the problem ϵ-`Exact-nNash` uses a different approximation metric. Furthermore, by Nash's Theorem, ϵ-`Exact-nNash` must be a total search problem.

While it is ϵ-approximate Nash equilibria that we will primarily concern ourselves with in this work, we include discussion of ϵ-near approximate Nash equilibria specifically to remind the reader that certain properties of Nash equilibria do not "almost hold" at ϵ-approximate Nash equilibria like they might at ϵ-near approximate equilibria.

Pure Nash Equilibria We first define the following subclass of games.

Definition 2.2.3 (Congestion Games)**.** A congestion game takes as input

- a set of n players.
- a finite set R of resources.
- for each player i, an action set $A_i \subseteq 2^R$.
- a congestion function $c : R \times \{0, \ldots, n\} \to \mathbb{Z}$ non-decreasing in the second argument.

The utility function for each player i is calculated as

$$u_i(a_1, \ldots, a_n) = -\sum_{r \in a_i} c\Big(r, \big|\{i' \mid r \in a_{i'}\}\big|\Big).$$

In essence, R is a set of shared resources, and the more players selecting the resource the more expensive it is.

Certain classes of games are guaranteed to admit *pure* Nash equilibria. For example,

Theorem 2.9 ([71])**.** *Every congestion game has a pure Nash equilibrium.*

So the natural question becomes: how difficult is it to find such an equilibrium? **PLS** certainly seems a natural class for the problem of finding pure Nash equilibria, should they exist. Define the following problems.

Definition 2.2.4. We define two subclasses of congestion games:

- A network congestion game is a congestion game which can be modeled as a (directed) graph (V, E). We set $R = E$, and associate with each player i a source s_i and sink t_i such that A_i is the set of all paths from s_i to t_i.

- A symmetric congestion game is a congestion game in which $A_1 = \ldots = A_n$.

(both of these are still guaranteed to admit pure Nash equilibria by Theorem 2.9). With these in mind, we define the problems

- `SymNetCongNash` is the problem of finding a pure Nash equilibrium given a symmetric network congestion game as input.
- `CongNash` is the problem of finding a pure Nash equilibrium given a general congestion game as input.
- `SymCongNash` is the problem of finding a pure Nash equilibrium given a general symmetric congestion game as input.
- `AsymNetCongNash` is the problem of finding a pure Nash equilibrium given a general network congestion game as input.

Based on these definitions,

Theorem 2.10 ([34])**.** *Congestion games admit the following dichotomy.*

- *`SymNetCongNash`* $\in$ **FP**
- *`CongNash` is* **PLS***-complete.*
- *`SymCongNash` is* **PLS***-complete.*
- *`AsymNetCongNash` is* **PLS***-complete.*

Mixed Nash Equilibria With the goal of classifying the problem of finding *mixed* Nash equilibria (which are guaranteed to exist in the general case), we shift our focus more concretely to **TFNP**.

Theorem 2.11 ([61])**.** `nNash` $\in$ **TFNP**.

In an effort to obtain a tighter bound, the authors also showed

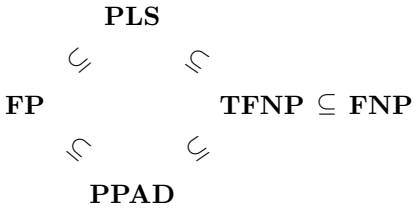
Theorem 2.12 ([61])**.** *If any* **FNP***-complete problem is in* **TFNP***, then* **NP** = **coNP**.

This implies that attempting to show `nNash` to be **FNP**-complete may be setting our sights too high:

Corollary 2.13. *If* `nNash` *is* **FNP***-complete, then* **NP** = **coNP**.

Instead, advancements were focused on subsets of **TFNP**. Refer back to Definition 2.1.20 for the definition of `EndOfTheLine`. Note that the parity argument immediately ensures `EndOfTheLine` $\in$ **TFNP**. Furthermore, because $|V| = 2^n$, it is not clear that `EndOfTheLine` $\in$ **P** (in fact, it is widely believed not to be).

Remark 13. We can offer the following list of inclusions:



It is unknown how **PLS** and **PPAD** compare (though there is a known oracle relative to which they are not equal [11]), however [27] show that there are a number of interesting problems in their intersection that are not believed to be in **FP**, such as finding *mixed* Nash equilibria in congestion games (and gradient descent, known to be complete for **PPAD** $\cap$ **PLS** [36]).

When defining **PPAD**, [67] included a list of problems residing in the class.

Theorem 2.14 ([67])**.** `2Nash` $\in$ **PPAD**.

However, for another decade it was unknown whether this upper bound was tight. A group of results from 2005-2009 iteratively refined each other (strengthening Lemma 2.5 to settle on the following result:

Theorem 2.15 ([24], [29], [18], [19], [20])**.** `2Nash` *is* **PPAD***-complete.*

For games with ≥ 4 players, hardness is shown by [24] using a reduction from the problem of finding a Brouwer fixed point. This is independently improved to games with ≥ 3 players by [29] and [18] (who reduce the problem to the four-player situation), before [19] and [20] did the same for equilibria with ≥ 2 players. Furthermore, while approximate equilibria are not necessarily close to exact equilibria, it is *also* hard to find ϵ-near approximate Nash equilibria. To capture this complexity, we present the following class.

Definition 2.2.5 (**FIXP**)**.** Consider the following problem:

- Input: An algebraic circuit over the basis $\{+, -, *, /, \max, \min\}$ with rational constants on n variables representing a function $f : [0,1]^n \to [0,1]^n$.
- Output: An ϵ-near approximate fixed point of f.

FIXP is the class of problems for which the above problem is complete. In a sense, **PPAD** captures a combinatorial hardness while **FIXP** (sometimes additionally) captures a numerical hardness.

Remark 14. It is known that **FIXP** $\subseteq$ **PSPACE** [33]. In fact, the class defined by the same problem over *linear* algebraic circuits (circuits over the basis $\{+, \max\}$ and *scalar* multiplication only), denoted **Linear-FIXP**, is equivalent to **PPAD**. So

$$\begin{array}{ccccccc} & & & & \mathbf{PLS} & & \\ & & \subsetneqq & & \subseteq & & \\ \mathbf{FP} & & & & & \mathbf{TFNP} \subseteq & \mathbf{FNP} \\ & & \subseteq & & \subsetneqq & & \cap \\ & & \mathbf{PPAD} = \mathbf{Linear\text{-}FIXP} & & \subseteq \mathbf{FIXP} & \subseteq & \mathbf{PSPACE} \end{array}$$

Theorem 2.16 ([33])**.** *ϵ-`Exact-nNash` is* **FIXP***-complete.*

Remark 15. By Remark 14, note that ϵ-`Exact-nNash` is a *harder* problem than `nNash` (assuming separation between **PPAD** and **FIXP**).

So finding exact, approximate, and near-exact Nash equilibria are all unlikely to admit polynomial time algorithms. Although these lower bounds seem very strong, there are some settings of n and m for which a Nash equilibrium can be found in quasipolynomial time.

Theorem 2.17 ([60],[8])**.** *Every n-player, m-action game admits a k-uniform ϵ-approximate Nash equilibrium (where k-uniform means that the probability given to each action is a multiple of $\frac{1}{k}$) for any*

$$k > \frac{8(\ln m + \ln n - \ln \epsilon + \ln 8)}{\epsilon^2}.$$

This implies that, for $m = \text{poly}(n)$ and input length $N \sim nm^n$, there is an algorithm for finding an ϵ-approximate Nash equilibrium in time $\text{poly}\left(N^{\log\log N}\right)$ *(it simply enumerates the possibilities). Furthermore, if instead $m = O(1)$, there is an algorithm for finding an ϵ-approximate Nash equilibrium in time* $\text{poly}\left(N^{\log\log\log N}\right)$.

Query Complexity

The time complexity of an algorithm, however, is not the only important consideration. As the description of an n-player game on m actions itself consists of nm^n payoffs, even a polynomial-time algorithm for finding a Nash equilibrium could have runtime *exponential* in the number of players. Situations arise in which access to the payoff functions is itself a very expensive resource; consider, for example, a multi-agent learning problem in which the computation of any given payoff requires an involved simulation [79].

Many results are also focused on the *communication complexity* of finding a Nash equilibrium when every player has access to only her own payoffs (when there is no communication, the situation is said to be *uncoupled*). However, it is easy to see that any communication lower bounds must also be query lower bounds; if a player with some knowledge of the game needs a certain amount of additional information to find a Nash equilibrium, then so does an algorithm beginning with no knowledge of the game.

Remark 16. Unless otherwise stated, we consider the profile query model in which a query consists of a pure strategy profile $\vec{a}$ and outputs the vector $\vec{u}(\vec{a})$ of the payoff to each player.

Bimatrix Games

Theorem 2.18 ([23])**.** *The communication complexity of finding a Nash equilibrium in a two-player game on m actions is $\Theta(m^2)$, even in the nondeterministic case when all payoffs belong to $\{0, 1\}$.*

Corollary 2.19. *The randomized* query *complexity of finding a Nash equilibrium in a two-player game on m actions is $\Theta(m^2)$ (even when all payoffs belong to $\{0, 1\}$).*

In fact, [35] show this is true even for *zero-sum* two-player games, and take the result further. Because the description of a two-player game on m actions (with payoffs of fixed description length) is of length $\Theta(m^2)$, this is quite strong. As for approximate equilibria in this setting:

Theorem 2.20 ([45])**.** *Let G be a two-player* zero-sum *game on m actions. There is a randomized algorithm which, with probability $\geq 1 - m^{-\frac{1}{8}}$, finds an ϵ-approximate Nash equilibrium of G making $O\left(\frac{m \log m}{\epsilon^2}\right)$ payoff queries.*

Certain improvements over this result are already known. One can find an ϵ-WSNE instead, or expand from zero-sum games:

Theorem 2.21 ([37]). *The following improvements over Theorem 2.20 are known.*

- *Let G be a two-player* zero-sum *game on m actions. There is a randomized algorithm which, with probability $\geq \left(1 - m^{-\frac{1}{8}}\right)\left(1 - \frac{2}{m}\right)^2$, finds an ϵ-WSNE of G making $O\left(\frac{m \log m}{\epsilon^4}\right)$ payoff queries.*
- *Let G be a* general *two-player game on m actions.*
  - *For any $\epsilon > 0$, there is a randomized algorithm which, with probability at least*

$$\left(1 - m^{-\frac{1}{8}}\right)\left(1 - \frac{2}{m}\right)^2,$$

finds a $\left(\frac{3-\sqrt{5}}{2} + \epsilon\right)$-approximate Nash equilibrium of G making $O\left(\frac{m \log m}{\epsilon^2}\right)$ payoff queries.

  - *For any $\epsilon > 0$, there is a randomized algorithm which, with probability at least*

$$\left(1 - m^{-\frac{1}{8}}\right)\left(1 - \frac{2}{m}\right)^3,$$

finds a $\left(\frac{2}{3} + \epsilon\right)$-WSNE of G making $O\left(\frac{m \log m}{\epsilon^4}\right)$ payoff queries (a $\left(\frac{1}{2} + \epsilon\right)$-WSNE if the payoffs of G belong to $\{0, 1\}$).

Considering specifically $\epsilon \geq \frac{1}{2}$, stronger improvements are known for general Nash equilibria.

Theorem 2.22 ([26]). *Let G be a two-player game on m actions. There is a deterministic algorithm which finds a $\frac{1}{2}$-approximate Nash equilibrium of G making $2m$ payoff queries.*

Proof Sketch. This can be done quite easily; taking any arbitrary action j_1 for player 1, find the best response j_2 for player 2, then find the best response j_1' for player 1 to j_2. Player 1 playing $\frac{1}{2}(j_1 + j_1')$ and player 2 playing j_2 is a $\frac{1}{2}$-approximate Nash equilibrium. □

In fact, [1] showed that some two-player games admit only $\frac{1}{4}$-approximate Nash equilibria of support size logarithmic in the number of actions. [38] later improved on this result:

Theorem 2.23 ([38]). *In a uniformly random 0/1 two-player zero-sum game on m actions, taking $\epsilon \in (0, 1)$, with high probability no $\frac{1}{1+\epsilon}$-approximate Nash equilibrium can be obtained from a pair of strategies with supports of size $o\left(\frac{\log n}{\epsilon^2}\right)$.*

This shows that Theorem 2.22 is, in fact, optimal for sub-logarithmic support size. The result of Theorem 2.22 can even be generalized and improved using a similar proof technique:

Theorem 2.24 ([35]). *Let G be a two-player game on m actions. For any i such that $2 \leq i \leq m-1$, the payoff query complexity of finding a $\left(1-\frac{1}{i}\right)$-approximate Nash equilibrium is in $[m-i+1, 2m-i+1]$.*

One cannot obtain results of this form for any smaller value of ϵ:

Theorem 2.25 ([37]). *The following query lower bounds hold for two-player games on m actions (even for zero-sum games over payoffs in $\{0,1\}$):*

- *For any $\epsilon < \frac{1}{2}$, the* deterministic *query complexity of finding an ϵ-approximate Nash equilibrium is $\Theta(m^2)$.*
- *For any $\epsilon < \frac{1}{4m}$, the* randomized *query complexity of finding an ϵ-approximate Nash equilibrium is $\Theta(m^2)$.*

We move on to consider games with a variable number of players and a fixed number of actions.

n-player games

Theorem 2.26 ([52]). *Though the following results are shown in the communication realm, once again they hold for query complexity as well:*

- *The communication complexity of finding a* pure *Nash equilibrium in an n-player game on a fixed number of actions is $2^{\Omega(n)}$.*
- *The communication complexity of finding a Nash equilibrium in an n-player game on a fixed number of actions is $2^{\Omega(n)}$.*

Once more, because the description of an n-player game on a fixed number of actions (with payoffs of fixed description length) is $2^{\Theta(n)}$, this is a strong bound. The situation grows even more grim.

Theorem 2.27 ([5]). *For any fixed number of actions, there is some fixed constant $\epsilon > 0$ such that the randomized query complexity of finding an ϵ-WSNE with probability $\geq 2^{-\Omega(n)}$ is $2^{\Omega(n)}$.*

Shortly after, this result was generalized

Theorem 2.28 ([17], [72]). *For any fixed number of actions, there is some fixed constant $\epsilon > 0$ such that the randomized query complexity of finding an ϵ-WSNE with probability $\geq 2^{-\Omega(n)}$ is $2^{\Omega(n)}$ (in fact, this even holds when considering "equilibria" in which a small fraction of players have regret greater than ϵ).*[2]

[2] More recently this constant ϵ has been improved to any $\epsilon < \frac{1}{3}$ [54].

This result is maximally pessimistic. Even weak approximations of Nash equilibria require a number of queries exponential in the number of players (note that the query complexity of ϵ-approximate Nash equilibria as a function of constant ϵ is *not* known) [7]. Of course, these results are focused on general games. We next consider restrictions to subclasses of games which might appear in important applications.

2.2.2 Restricted Games

Lipschitz Games

We produce the formal definition of Lipschitz games in Definitions 2.1.8, 2.1.9, and 2.1.11 above. These games, in which there are assumed to be a large number of players, each of whom has a payoff function that is Lipschitz continuous in the actions of the other players, evolved over the course of many works. [75] examines the existence of pure Nash equilibria in games with a continuum of players associated with a non-atomic measure. [58] refines this to a study of equilibria in finite but semi-anonymous games (typed games in which every player's payoff function depends on the action distribution of players of each type) with a large number of players and Lipschitz continuous payoff functions. Several papers follow iterating on these results and exploring the properties of these equilibria (e.g. [14], [15], [30], [49]). The work we will build most build upon is that of Azrieli and Shmaya ([2]) extending [58] to non-anonymous games.

For the remainder of this work, we consider only games with payoffs in the range $[0, 1]$. Lipschitz games are natural to consider; in essence they arise in situations in which no single player's actions can affect any other player *significantly*, but aggregate behavior still can. These games seem to capture the behavior of situations in which there are a large number of players of similar levels of influence; consider individuals trading in a market or certain types of network games as defined above (no single car has significant impact on another car's travel time). Interestingly,

Theorem 2.29 ([2])**.** *Given a λ-Lipschitz game G on n players and m actions and any*

$$\epsilon \geq \lambda\sqrt{8n\log(2mn)},$$

G admits an ϵ-approximate pure *Nash equilibrium. Furthermore, for all large enough n there exists a $\frac{60}{\sqrt{n}}$-Lipschitz binary-action game that does not admit a pure $\frac{1}{3}$-approximate Nash equilibrium (so the bound on ϵ is tight up to logarithmic factors).*

The first part of the above theorem is the foundation for the remainder of this work.

In addition, [2] show a reduction from exact Nash equilibria to pure ϵ-approximate Nash equilibria, reversing the direction of their proof and showing that if Theorem 2.29 holds, then Theorem 1.1 holds (more on this in Section 3.2.1). The query complexity of finding such pure approximate equilibria, however, was unknown prior to the work in Chapter 3 [7].

Given their commonplace nature, Lipschitz games have been studied further and some properties have been uncovered.

Definition 2.2.6 (Immunity)**.** In a game G, a strategy profile $\mathbf{p} \in \Delta A_1 \times \ldots \times \Delta A_n$ is (ϵ, t)-immune if for every set $S \subset [n]$ with $|S| \leq t$, every $b_S \in \prod_{i \in S} \Delta A_i$, and every player $i \notin S$,

$$u_i(\mathbf{p}) \geq u_i(\mathbf{p}_{-S}, b_S) - \epsilon.$$

In other words, if at most t players change their strategy, no other player's payoff is reduced by more than ϵ.

Definition 2.2.7 (Tolerance)**.** Similarly, a strategy profile $\mathbf{p}$ is (ϵ, t)-tolerant if, for every subset $S \subset [n]$ with $|S| \leq t$ every $b_S \in \prod_{i \in S} \Delta A_i$, every $i \notin S$, and every strategy $\vec{p}_i'$,

$$u_i(\mathbf{p}_{-S}, b_S) \geq u_i(\vec{p}_i', \mathbf{p}_{-(S \cup \{i\})}, b_S) - \epsilon.$$

In other words, the *regret* of player i is at most ϵ when at most t players change their strategy.

Remark 17. Note that any (ϵ, t)-tolerant strategy profile is an ϵ-Nash equilibrium.

Definition 2.2.8 (Coalitional Nash Equilibrium)**.** Finally, a strategy profile $\mathbf{p}$ is an (ϵ, t)-coalitional Nash equilibrium if, for every subset $S \subset [n]$ with $|S| \leq t$, every strategy profile $\mathbf{p}'_S$, and every player $i \in S$,

$$u_i(\mathbf{p}) \geq u_i(\mathbf{p}_{-S}, \mathbf{p}'_S) - \epsilon.$$

In other words, even collaborating with a group of $t - 1$ other players, no player can improve her payoff by more than ϵ.

Remark 18. Once again, clearly any (ϵ, t)-coalitional Nash equilibrium is also an ϵ-Nash equilibrium of G.

With these three definitions out of the way,

Theorem 2.30 ([50]). *Let G be any n-player λ-Lipschitz game with Nash equilibrium $\mathbf{p}$. Then, for any $t \in [n]$, $\mathbf{p}$ is*

- $\left(\frac{\lambda t}{n}, t\right)$*-immune.*
- $\left(\frac{2\lambda t}{n}, t\right)$*-tolerant.*
- $\left(\frac{3\lambda t}{n}, t+1\right)$*-coalitional.*

These are indeed interesting properties of exact Nash equilibria in λ-Lipschitz games. So how difficult is it to find such equilibria?

In answering this question, it is often convenient to consider $\frac{1}{n}$-Lipschitz games.

Remark 19. The problem of finding an ϵ-approximate Nash equilibrium in a λ-Lipschitz game is equivalent to that of finding a $\frac{\epsilon}{k}$-approximate equilibrium in a $\frac{\lambda}{k}$-Lipschitz game.

With this in mind, we proceed with results on the complexity of finding equilibria.

Theorem 2.31 ([41]). *For any $\alpha, \eta > 0$, the* uncoupled *query complexity of finding a $\left(\frac{1}{8}+\alpha\right)$-approximate Nash equilibria of games in $\mathcal{G}\left(n, 2, \frac{1}{n}\right)$ with success probability $1-\eta$ is $O\left(\frac{1}{\alpha^4}\log\left(\frac{n}{\alpha\eta}\right)\right)$. Holding α, η constant, this bound is tight at $\Theta(\log n)$.*

Allowing communication, this approximation factor can be improved to $\frac{1}{8} - \frac{1}{220}$.

This logarithmic complexity comes from a constant number of queries on mixed strategy profiles - one must perform logarithmically many "pure queries" to approximate a single "mixed query". If the number of actions is unfixed, however, the situation becomes much less clear.

Theorem 2.32 ([41]). *Given a λ-Lipschitz game G on n players and m actions and any $\alpha, \eta, N > 0$, taking*

$$\epsilon = \begin{cases} \lambda n\left(\frac{m-1}{m}\right)\left(1+\frac{1}{N}+\frac{\alpha}{\lambda n}\right)^2 & \frac{1}{2n}\left(\frac{N}{N+1}\right) < \lambda < \frac{1}{2n}\left(\frac{m}{m-1}\right)\left(\frac{N}{N+1}\right) \\ 1 - \frac{1}{4\lambda n}\left(\frac{m}{m-1}\right) + \frac{1}{N} + \frac{\alpha}{\lambda n} & \textit{otherwise} \end{cases}$$

there is an uncoupled algorithm finding an ϵ-approximate Nash equilibrium of G with success probability $1-\eta$ making

$$O\left(\frac{m^2}{\alpha^3}\log\left(\frac{nN^2}{\eta}\right)\right)$$

queries. In particular, with any number of actions and taking $\lambda = \frac{1}{n}$, $\epsilon \leq \frac{3}{4} + \frac{1}{2N}$.

Remark 20. The above results also indirectly show an $O(\log \log N)$-time algorithm for finding such equilibria (where N is the length of the description of a game with payoffs of fixed description length). None of these results are currently known for ϵ-WSNE. In fact, despite the importance of such games, not much more is known about general Lipschitz games at all. Instead, many researchers focus on the restriction to *anonymous* Lipschitz games. Note that all congestion games are anonymous, but one can imagine a network game in which each player travelling an edge affects the congestion function differently (for example a truck vs a motorcycle on a road, or a person checking their email vs streaming "The Dark Knight").

Lipschitz Anonymous Games

Definition 2.2.9 (Anonymous Game)**.** An anonymous game on n players and m actions has a utility function that depends only on a_i and the number of players playing each action in $[m]$. A game is *self-anonymous* if the utility function does not even depend on a_i. An anonymous game is *symmetric* if every player shares the same utility function. A self-anonymous symmetric game is called *self-symmetric*.

As it turns out, the result of Theorem 2.29 can be improved past the general case's lower bound when restricting attention to Lipschitz anonymous games.

Theorem 2.33 ([28])**.** *Every λ-Lipschitz anonymous game on n players and m strategies has an ϵ-approximate* pure *Nash equilibrium for $\epsilon = \Omega(\lambda m)$. In particular, ϵ does not depend on n. Such an equilibrium can be found deterministically in time $(n+m)\text{polylog}(n)$ times the description length of the payoffs.*

Furthermore, when restricting further to binary-action games,

Theorem 2.34 ([46])**.** *There is an algorithm taking as input an n-player binary-action λ-Lipschitz anonymous game that finds a $3(\lambda+\delta)$-approximate pure Nash equilibrium querying at most $4n \log n$ payoffs (total payoffs, not profiles)* even if the queries may have additive error δ.

Anonymous Games

Many more results are known on anonymous games than on the restrictions mentioned so far. Those presented below are among the more relevant to this work.

Existence Claims It is once again infeasible to search for exact equilibria in anonymous games:

Theorem 2.35 ([46]). *There exists a three-player binary-action anonymous game with a unique Nash equilibrium in which all three players must randomize with irrational probabilities.*

So we once again turn our attention to approximate equilibria (Theorem 2.40 below provides further motivation for this decision). In considering ϵ-WSNE of anonymous games,

Theorem 2.36 ([28]). *For every $\epsilon > 0$, any binary-action anonymous game admits an ϵ-WSNE such that, for some integer $\ell = O\left(\frac{1}{\epsilon}\right)$, either*

- *at most ℓ^3 players play mixed strategies, and their probabilities are all integer multiples of $\frac{1}{\ell^2}$ OR*
- *all players who randomize use the same mixed strategy, and all probabilities are integer multiples of $\frac{1}{\ell n}$.*

Additionally, one can expand the results of Theorem 2.30 to Lipschitz anonymous games:

Theorem 2.37 ([50]). *For any $m \in \mathbb{N}, \epsilon > 0$, there is some $t = \Theta(\sqrt{n})$ such that G has an ϵ-approximate Nash equilibrium that is*

- *(ϵ, t)-immune,*
- *$(3\epsilon, t)$-tolerant,*
- *$(4\epsilon, t+1)$-coalitional.*

Computational Complexity The main computational result is presented in Theorem 2.38, though Theorem 2.43 should also be kept in mind.

Theorem 2.38 ([28]). *There exists some function g such that, for any $\epsilon > 0$, an ϵ-WSNE of an anonymous game on n players and m actions can be found in randomized time $n^{g(m,\frac{1}{\epsilon})}$ times the description length of the payoffs by an* oblivious *(non-adaptive) algorithm (or $n^{g(m,\frac{1}{\epsilon},t)}$ in* typed *anonymous games with t types in which payoffs can depend on the number of players of each type playing each action).*

On binary-action games, this can be improved to randomized time $\left(\frac{1}{\epsilon}\right)^{O\left(\frac{1}{\epsilon^2}\right)} \text{poly}(n)$ times the description length of the payoffs. Furthermore, the polynomial dependence on $\frac{1}{\epsilon}$ in the exponent cannot be removed for any oblivious algorithm.

Removing the oblivious requirement, this can be further improved to $\left(\frac{1}{\epsilon}\right)^{O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)}\mathrm{poly}(n)$ *times the description length of the payoffs.*

While better algorithms can be found for more general equilibria, one cannot hope to improve further by slightly restricting the class of games considered.

Theorem 2.39 ([46])**.** *If there is an FPTAS (fully polynomial-time approximation scheme, or an algorithm running in polynomial time in* every *argument, including* $\frac{1}{\epsilon}$*) for computing an* ϵ*-Nash equilibrium of binary-action* self-anonymous *games, then there is also an FPTAS for computing* ϵ*-approximate Nash equilibria of general binary-action anonymous games.*

Query Complexity

Theorem 2.40 ([46])**.** *Any randomized algorithm needs to query* $\Omega(n^2)$ *payoffs in expectation in order to find an exact Nash equilibrium, even of an* n*-player binary-action self-anonymous game.*

So instead we turn our attention to approximate equilibria. However,

Theorem 2.41 ([46])**.** *For any* $\epsilon < \frac{1}{2}$*, any randomized algorithm must query* $\Omega(n)$ *strategy profiles (i.e.* $\Omega(n^2)$ *total payoffs) to find an* ϵ*-WSNE in an* n*-player binary-action anonymous game.*

Since these games only admit $2n^2$ total payoffs, querying $\Omega(n^2)$ of them rules out any query-efficient algorithm. We can either restrict the class of games we consider or relax the requirements on the solution. First, restricting the class of games,

Theorem 2.42 ([46])**.** *In any* n*-player binary-action* self-anonymous game*, the uniform mixed strategy profile is an* $O\left(\frac{1}{\sqrt{n}}\right)$*-WSNE. As a result, such an equilibrium can be found with query complexity* 0*.*

Now instead we relax the requirement of a *well-supported* equilibrium,

Theorem 2.43 ([46])**.** *For* ϵ *satisfying* $\frac{1}{\epsilon} = O\left(n^{\frac{1}{4}}\right)$*, there is a randomized algorithm querying* $O\left(\sqrt{n}\log^2 n\right)$ *payoffs finding an* ϵ*-approximate Nash equilibrium in time* $O\left(n^{\frac{3}{2}}\log^2 n\right)$*.*

Remark 21. This running time is significantly better than that of Theorem 2.38, but the resulting equilibrium is no longer well-supported.

Instead consider further restrictions in which concerns raised in Theorem 2.35 and 2.40 no longer apply.

Theorem 2.44 ([46]). *A pure Nash equilibrium of any n-player binary-action* symmetric *game can be found querying* $O(\log n)$ *payoffs.*

This can be done using a binary search approach once showing that such an equilibrium is guaranteed to exist. In order to expand the number of actions, we restrict the class of games further to *self*-symmetric games.

Theorem 2.45 ([46]). *The randomized query complexity of finding a pure Nash equilibrium of a self-symmetric game on n players and m actions is* $\Theta\left(n^{\frac{m-1}{2}}\right)$ *for any* $m \geq 5$*, and the deterministic complexity is* $\Theta\left(n^{m-2}\right)$ *for any* $m > 2$.

Polymatrix Games

There is one more game restriction that we consider in this work - *polymatrix* games. Here, we define such games, introduced in [56]:

Definition 2.2.10 (Polymatrix Games)**.** A polymatrix game is an n-player, m-action game in which each pair of players i_1, i_2 plays a bimatrix game and receives as payoff the sum of their $n-1$ payoffs from each separate bimatrix game. If player i_1 plays action j_1 and player i_2 plays action j_2, we denote the payoff to player i_1 from this bimatrix game as β_{i_1,i_2,j_1,j_2}.

Polymatrix games are a well-studied restriction withma many interesting properties, and applications such as finding approximate equilibria of general games by approximating them with a sequence of polymatrix games [48], visual labeling with neural networks [62], or graph transduction in semi-supervised machine learning [32]. The important question, of course, is how challenging they are to solve:

Theorem 2.46 ([13]). *Nash equilibria of* zero-sum *polymatrix games can be found in polynomial time (by a reduction to linear programming). Furthermore, all coarse correlated equilibria are Nash equilibria.*

However, when these games are not zero-sum, they become significantly more difficult to solve:

Theorem 2.47 ([55]). *Exact Nash equilibria of polymatrix games can be found in* exponential *time via Lemke's algorithm.*

This is the best we can easily do at the moment:

Theorem 2.48 ([73]). *There exists some constant $\epsilon > 0$ such that given a binary-action polymatrix game, finding an ϵ-ANE is* **PPAD**-*complete.*

Some amount of progress has still been made over the trivial upper bound of finding $\frac{n-1}{n}$-approximate Nash equilibria:

Theorem 2.49 ([31]). *Regardless of the number of players, a $(0.5+\delta)$-approximate Nash equilibrium of a polymatrix game can be found in time polynomial in the size of the input and $\frac{1}{\delta}$ (via a gradient descent algorithm).*

Chapter 4 presents results primarily focused on *Lipschitz* polymatrix games. In particular, we extend the result of Rubinstein presented in Theorem 2.48 to pure equilibria in *Lipschitz* polymatrix games.

2.2.3 Best-Response Queries

Of additional interest, another branch of study considers modifying the query model referenced in Remark 16. In this model, query oracles take the same input, but instead of returning the payoff each player receives, they return only each player's best response to the queried profile. This model is strictly weaker than the payoff query model considered above. In fact, even the $2^{O(n)}$ trivial upper bound on all of the problems considered above no longer applies.

Theorem 2.50 ([40]). *The* best-response *query complexity of computing an* exact *Nash equilibrium is* infinite. *Furthermore, in a two-player binary-action game, the best-response query complexity of finding an ϵ-WSNE is $\Omega\left(\log\left(\frac{1}{\epsilon}\right)\right)$.*

This seems like an unhelpful model to consider. However, for certain restricted classes of games, the situation appears less grim.

Theorem 2.51 ([3]). *For any $\epsilon > 0$, there is a best-response query dynamic on λ-Lipschitz anonymous binary-action games which arrives at a pure $(\epsilon\lambda)$-approximate Nash equilibrium in expected time $O\left(n^2 \log n\right)$.*

It is yet unknown if a similar result can be shown for general λ-Lipschitz games.

Chapter 3

Query Complexity

3.1 Overview

The material of this chapter was presented at the 14th International Symposium on Algorithmic Game Theory [42]. Below we establish the following results:

- In Section 3.3, Theorem 3.8 shows a lower bound exponential in $\frac{n\lambda}{\epsilon}$ on the randomized query complexity of finding ϵ-approximate *pure* Nash equilibria of games in $\mathcal{G}(n, 2, \lambda)$.
- In Section 3.4, we generalize the concept of Lipschitz games. Theorem 3.10 provides a reduction from finding approximate equilibria in our new class of "Multi-Lipschitz" games to finding approximate equilibria of Lipschitz games.

These results also use the following simple lemma (which holds for all types of queries and equilibria mentioned in Section 2.1.1).

Lemma 3.1. *For any constants $\lambda' < \lambda \leq 1, \epsilon > 0$, there is a query-free reduction from finding ϵ-approximate equilibria of games in $\mathcal{G}(n, m, \lambda)$ to finding $\frac{\lambda'}{\lambda}\epsilon$-approximate equilibria of games in $\mathcal{G}(n, m, \lambda')$.*

In other words, query complexity *upper bounds* hold as λ and ϵ are scaled up together, and query complexity *lower bounds* hold as they are scaled down. The proof is quite simple:

Proof. Take any constants $\lambda' < \lambda \leq 1, \epsilon > 0$ and any game $G \in \mathcal{G}(n, m, \lambda)$. Consider an algorithm A that finds $\frac{\lambda'}{\lambda}\epsilon$-approximate equilibria of games in $\mathcal{G}(n, m, \lambda')$. Run A on G, multiplying every payoff

queried by $\frac{\lambda'}{\lambda}$ (making no additional queries) and output the result. This is guaranteed to be an ϵ-approximate equilibrium of G. □

Remark 22. Note that the lemma does not hold for $\lambda' > \lambda$, as the reduction could introduce payoffs that are larger than 1.

3.2 Techniques

3.2.1 The Induced Population Game

Background

This section introduces a reduction utilized by [2] in an alternative proof of Nash's Theorem, and by [4] to upper bound the support size of ϵ-ANEs.

Definition 3.2.1. Given a game G with payoff function $\mathbf{u}$, we define the *population game* induced by G, $G' = g_G(L)$ with payoff function $\mathbf{u}'$ as follows. Every player i is replaced by a population of L players (v_ℓ^i for $\ell \in [L]$), each playing G against the aggregate behavior of the other $n-1$ populations. More precisely,

$$u'_{v_\ell^i}(\mathbf{p}') = u_i\left(p'_{v_\ell^i}, \mathbf{p}_{-i}\right)$$

where

$$p_{i'} = \frac{1}{L}\sum_{\ell=1}^{L} p'_{v_\ell^{i'}}$$

for all $i' \neq i$.

Population games date back even to Nash's thesis [64], in which he uses them to justify the consideration of mixed equilibria. To date, the reduction to the induced population game has been focused on proofs of existence. We show that the reduction can be made query-efficient: an equilibrium of $g_G(L)$ induces an equilibrium on G *which can be found with few additional queries.* This technique is the foundation for many of the results of this work.

An Efficient Reduction

One of the primary contributions of this work is the discovery of a technique to make the reduction to the induced population game efficient (in different senses). It turns out that, if this can be achieved,

the problem of finding equilibria in general games can be reduced to the problem of finding equilibria in Lipschitz games. Armed with the efficient reduction technique described below, we are able to derive lower bounds on Lipschitz games from existing known lower bounds.

Lemma 3.2. *Given an n-player, m-action game G and a population game $G' = g_G(L)$ induced by G, if an ϵ-PNE of G' can be found by an algorithm making q (δ, γ)-distribution queries of G', then an ϵ-WSNE of G can be found by an algorithm making $n \cdot m \cdot q$ $(\delta, \gamma/L)$-distribution queries of G.*

We first establish the following preliminary claim:

Claim 1. Any δ-distribution query of $G' = g_G(L)$ can be simulated by Ln δ-distribution queries of G.

At a high level, this is because we will make a single query of the original game for each player $v_\ell^{(i)}$ in the population game in which we assume that player i plays the strategy of player $v_\ell^{(i)}$ and the remaining $n-1$ players play the aggregate behavior of their entire populations.

Proof of Claim 1. Consider any δ-distribution query of the Ln-player game G'. Such a query takes a mixed strategy profile $\mathbf{p'}$ and returns a δ-approximate payoff for each of the Ln players. If we consider the aggregate strategy profile $\mathbf{p}$ directly, the query $\mathcal{Q}_\delta^G(\mathbf{p})$ will return the payoff to each of the n players in G. However, we cannot derive the payoff to a player $v_\ell^{(i)}$ in G' from the average payoff to each such player. Instead, for every $i \in [n], \ell \in [L]$, consider the strategy profile $\mathbf{p}_\ell^{(i)}$ replacing player i with player $v_\ell^{(i)}$ and replacing the remaining $n-1$ players by the aggregate behavior of their populations. Then, a query $\mathcal{Q}_\delta^G(\mathbf{p}_\ell^{(i)})$ will return a vector of n payoffs, the i^{th} element of which will correctly be a δ-approximation of the payoff to player $v_\ell^{(i)}$ when $\mathbf{p'}$ is played in game G'. □

Now, to complete the proof of the Lemma, we will improve the blowup factor from Ln to mn. In essence, this is because we do not actually need to make a query for every single player in each population. Instead, if we query every pure strategy for player i against the aggregate behavior of the other $n-1$ populations, we can then calculate the payoff to *any* mixed strategy of a player $v_\ell^{(i)}$ in population i.

Proof of Lemma 3.2. Consider any δ-distribution query of the Ln-player game G'. In the proof above, this query is simulated by a single query for each player in G'. Instead, we can simulate it by m queries for every player in G. Instead of queries $\mathbf{p}_\ell^{(i)}$ for every $i \in [n], \ell \in [L]$, consider queries $\mathbf{p}_j^{(i)}$ for every $i \in [n], j \in [m]$ in which player i plays action j and the remaining $n-1$ players still play the aggregate behavior of their populations. From these $n \cdot m$ queries, one can derive the payoffs from the Ln

queries $\mathbf{p}_\ell^{(i)}$ above by directly calculating the expected payoff to the mixed strategy played by player $v_\ell^{(i)}$ given the payoffs for each pure action. So each δ-distribution query of G' can be simulated by $n \cdot m$ δ-distribution queries of G. Furthermore, if the query was a (δ, γ)-distribution query of G', the aggregate behavior of L players guarantees that the new queries are all $(\delta, \gamma/L)$-distribution queries. Finally, any ϵ-PNE $\mathbf{a}'^*$ of G' must also induce an ϵ-WSNE $\mathbf{p}^*$ of G in which each player of G plays the aggregate strategy of their population. If $\mathbf{p}^*$ were not an ϵ-WSNE, then some player in G' must suffer regret $> \epsilon$ in $\mathbf{a}'^*$. This completes the proof of Lemma 3.2. □

Remark 23. A similar, but much simpler, result holds for time complexity and will become relevant for the hardness proof in Chapter 4.

3.2.2 Approximate Equilibria of Perturbed Games

The lemmas in this section consider games that may be learned from non-exact query algorithms, and describe the knowledge we can still glean from such perturbations of the game of interest. The first lemma expresses a subtle but important point regarding approximations that we will make:

Lemma 3.3. *Given two games G, G' such that for every pure action profile $\mathbf{a}$, $||\mathbf{u}(\mathbf{a}) - \mathbf{u}'(\mathbf{a})||_\infty \leq \delta$, any ϵ-ANE of G' is an $(\epsilon + 2\delta)$-ANE of G.*

Proof. Consider any ϵ-ANE $\mathbf{p}'^*$ of G'. Then, for any player i, the payoff for playing $\mathbf{p}_i'^*$ against $\mathbf{p}_{-i}'^*$ in G cannot be that different from that in G': in fact,

$$u_i(\mathbf{p}_i'^*, \mathbf{p}_{-i}'^*) \geq u_i'(\mathbf{p}_i'^*, \mathbf{p}_{-i}'^*) - \delta.$$

Furthermore, for any action j, we similarly have

$$u_i(j, \mathbf{p}_{-i}'^*) \leq u_i'(j, \mathbf{p}_{-i}'^*) + \delta.$$

Finally, because $\mathbf{p}'^*$ is an ϵ-ANE of G', for any action j:

$$u_i'(j, \mathbf{p}_{-i}'^*) - u_i'(\mathbf{p}_i'^*, \mathbf{p}_{-i}'^*) \leq \epsilon.$$

Substituting the first two inequalities into the final one yields, once again for any player i and any

action j:

$$u_i(j, \mathbf{p}'^*_{-i}) - u_i(\mathbf{p}'^*_i, \mathbf{p}'^*_{-i}) \leq \epsilon + 2\delta$$

completing the proof. □

With this lemma in mind, we consider the situations in which we might wish to apply it:

Lemma 3.4. *Take any $\delta \leq \min\{\epsilon/4, \lambda/4\}$. Given a game $G \in \mathcal{G}(n, 2, \lambda)$ and a set S of pure profiles, there is another game $G' \in \mathcal{G}\left(n, 2, \frac{3\lambda}{2}\right)$ that is consistent with the outputs of all δ-distribution queries of the profiles in S of G (i.e. $\mathbf{u}'(\mathbf{a}) = \tilde{\mathbf{u}}(\mathbf{a})$ for all $\mathbf{a} \in S$) for which any $\epsilon/2$-PNE is an ϵ-PNE of G. Figure 3.1 visually depicts this observation.*

Proof. In order to prove the Lemma, we need to find a $\frac{3\lambda}{2}$-Lipschitz game G' consistent with the queries in which any action profile satisfies $\mathbf{a} \in \{0,1\}^n$, $||\mathbf{u}'(\mathbf{a}) - \mathbf{u}(\mathbf{a})||_\infty \leq \delta$. We can then apply Lemma 3.3 to complete the proof.

So consider the incomplete payoff function $\tilde{\mathbf{u}}$ learned by δ-distribution queries of G. Define the game G' agreeing with $\tilde{\mathbf{u}}$ on all queried actions and agreeing with G on all others:

$$\mathbf{u}'(\mathbf{a}) = \begin{cases} \tilde{\mathbf{u}}(\mathbf{a}) & \mathbf{a} \text{ was queried} \\ \mathbf{u}(\mathbf{a}) & \text{otherwise} \end{cases}$$

G' is well-defined, and for every action profile $\mathbf{a} \in \{0,1\}^n$,

$$||\mathbf{u}'(\mathbf{a}) - \mathbf{u}(\mathbf{a})||_\infty \leq ||\tilde{\mathbf{u}}(\mathbf{a}) - \mathbf{u}(\mathbf{a})||_\infty \leq \delta,$$

so applying Lemma 3.3 (keeping in mind that $\delta \leq \frac{\epsilon}{4}$), any $\epsilon/2$-PNE of G' is an ϵ-PNE of G. It remains to show that G' has a sufficiently small Lipschitz constant.

Consider any player $i \in [n]$ and any action profiles $\mathbf{a}^{(1)}, \mathbf{a}^{(2)} \in \{0,1\}^n$ such that

$$\left|\left|\mathbf{a}^{(1)}_{-i} - \mathbf{a}^{(2)}_{-i}\right|\right|_1 = c \geq 1.$$

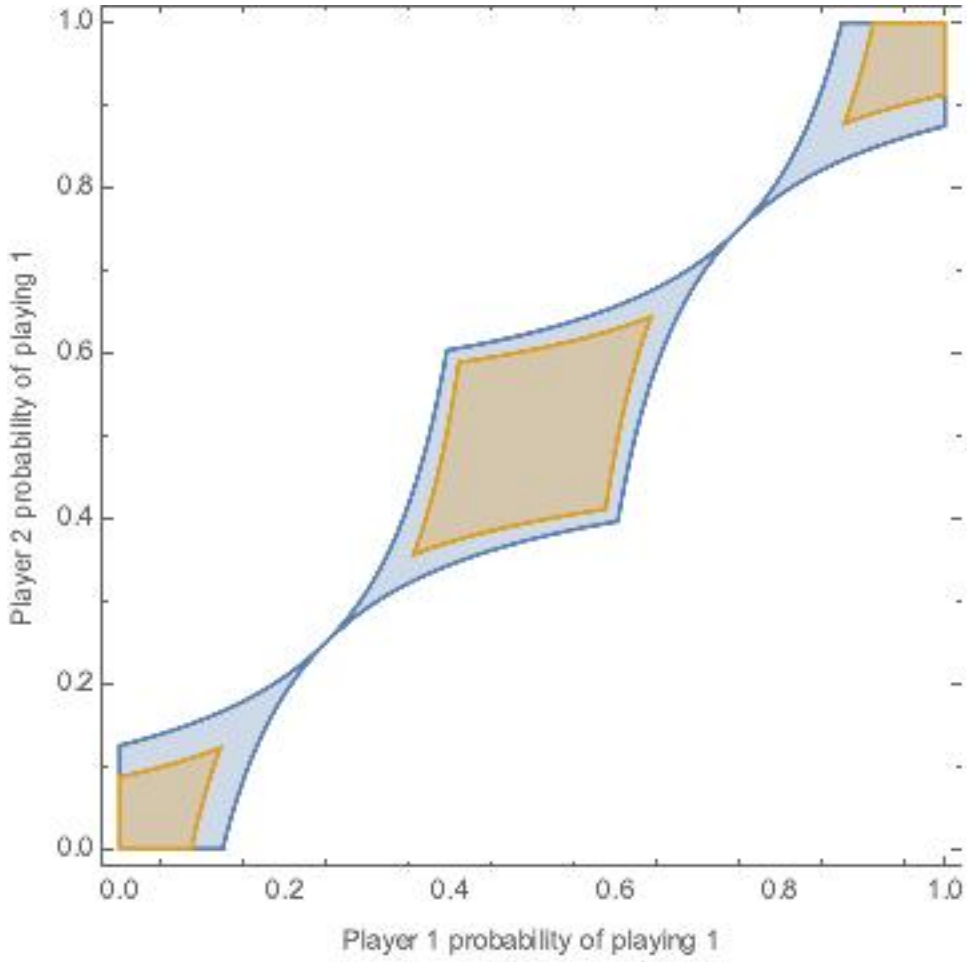


Figure 3.1: Taking G' to be the Coordination Game and considering fixed values of ϵ and δ, if each axis represents the probability of one of the players playing H, then the blue region shows the set of ϵ-approximate equilibria of G' (the acceptable outputs of algorithm B) while the orange region shows the set of all $\frac{\epsilon}{2}$-approximate equilibria of any possible game G'' in which each payoff may be perturbed by at most δ (the possible outputs of algorithm B).

Then, since G is λ-Lipschitz,

$$\begin{aligned}\left|u_i'\left(\mathbf{a}^{(1)}\right) - u_i'\left(\mathbf{a}^{(2)}\right)\right| &\leq \left|u_i\left(\mathbf{a}^{(1)}\right) - u_i\left(\mathbf{a}^{(2)}\right)\right| + 2\delta \\ &\leq c\lambda + 2\delta \\ &\leq c\left(\lambda + 2\delta\right) \\ &< c\frac{3\lambda}{2}\end{aligned}$$

using the fact that $\delta \leq \lambda/4$. This completes the proof. □

3.3 Hardness of Approximate Pure Equilibria

In this section we will rely heavily on the following result of Babichenko.

Theorem 3.5 ([5])**.** *There is a constant $\epsilon_0 > 0$ such that, for any $\beta = 2^{-o(n)}$, the randomized δ-distribution query complexity of finding an ϵ_0-WSNE of n-player binary-action games with probability at least β is $\delta^2 2^{\Omega(n)}$.*

For the remainder of this chapter and Chapter 4, the symbol ϵ_0 refers to this specific constant. A simple application of Lemma 3.1 yields:

Corollary 3.6. *There is a constant $\epsilon_0 > 0$ such that, for any $\beta = 2^{-o(n)}$, the randomized δ-distribution query complexity of finding an $\epsilon_0\lambda$-WSNE of games in $\mathcal{G}(n, 2, \lambda)$ with probability at least β is $\delta^2 2^{\Omega(n)}$.*

3.3.1 Warmup

As a warmup, we remove a degree of freedom and prove a slightly less general result.

Theorem 3.7. *For every $k \geq 4$, for any $\lambda(n)$-Lipschitz binary-action game G on n players, the N-player, $\lambda'(N)$-Lipschitz induced population game $g_G\left(\left(\frac{4}{\epsilon_0}\right)^k n^{k-1}\right)$ is guaranteed to have an ϵ-approximate pure Nash equilibrium for*

$$\epsilon = 4N^{\frac{k-1}{k}}\lambda'(N)$$

(i.e. $o(N\lambda'(N))$). However, any deterministic algorithm finding such an equilibrium needs to perform $2^{\Omega\left(n^{\frac{1}{k}}\right)}$ δ-distribution queries of G for any $\delta = 1/\mathrm{poly}(n)$.

Proof. Take any $\lambda(n)$-Lipschitz game G, and let $G' = g_G\left(\left(\frac{4}{\epsilon_0}\right)^k n^{k-1}\right)$. We first prove the existence of the equilibrium, demonstrating that the result is non-trivial. Because $k \geq 4$, note that

$$L = \left(\frac{4}{\epsilon_0}\right)^k n^{k-1} \geq \frac{256}{\epsilon_0^4} n^3.$$

As such (multiplying both sides by L),

$$L^2 \geq \frac{256}{\epsilon_0^4} L n^3$$

so (taking the square root of both sides)

$$L \geq \frac{8}{\epsilon_0^2} n\sqrt{4Ln} \geq \frac{8}{\epsilon_0^2} n \log(4Ln)$$

and (again multiplying both sides by L)

$$L^2 \geq \frac{8}{\epsilon_0^2} Ln \log(4Ln).$$

Once more taking the square root,

$$L \geq \frac{1}{\epsilon_0}\sqrt{8Ln\log(4Ln)}$$

so

$$\frac{\lambda(n)}{L} \leq \frac{\lambda(n)\epsilon_0}{\sqrt{8Ln\log(4Ln)}}$$

and thus Theorem 2.29 ensures that G' (an Ln player game) has a $\lambda(n)\epsilon_0$-approximate *pure* Nash equilibrium.

So define $N = Ln$ to be the number of players in G', and note that

$$\lambda'(N) = \frac{\lambda\left(\frac{N}{L}\right)}{L}$$

is the Lipschitz parameter of G'. Since

$$L = \left(\frac{4}{\epsilon_0}\right)^k n^{k-1} = \left(\frac{4}{\epsilon_0}\right)^k \left(\frac{N}{L}\right)^{k-1}$$

we can isolate L to obtain

$$L^k = \left(\frac{4}{\epsilon_0}\right)^k N^{k-1}$$

or

$$L = \frac{4}{\epsilon_0} N^{\frac{k-1}{k}}$$

to obtain L as a function of N instead of n. As such,

$$\lambda(n)\epsilon_0 = L\lambda'(N)\epsilon_0 = 4N^{\frac{k-1}{k}}\lambda'(N)$$

proving the existence of the equilibrium.

To prove the lower bound, assume that we can find this equilibrium making $2^{O(n^{\frac{1}{k}})}$ δ-distribution queries of G. We can directly apply Lemma 3.2 to see that a $\lambda(n)\epsilon_0$-WSNE of G can be obtained by an algorithm making $2^{o(n^{\frac{1}{k}})}$ δ-distribution queries, which contradicts Corollary 3.6 and completes our proof. □

3.3.2 Main Result

We are now ready to state the main result of this chapter – an exponential lower bound on the randomized query complexity of finding ϵ-PNEs of λ-Lipschitz games. This result will heavily generalize Theorem 3.7.

Theorem 3.8 (Main Result)**.** *There exists some constant ϵ_0 such that, for any*

$$n \in \mathbb{N}, \qquad \epsilon < \epsilon_0, \qquad \lambda \leq \frac{\epsilon}{\sqrt{8n \log 4n}},$$

while every game in $\mathcal{G}(n, 2, \lambda)$ has an ϵ-PNE, any randomized algorithm finding such equilibria with probability at least $\beta = 1/\text{poly}(n)$ must make

$$\lambda^2 2^{\Omega(n\lambda/\epsilon)}$$

profile queries.

We will break the proof into parts, proceeding via contradiction. As an overview:

Step 1 (Algorithm A) Take any

$$q = \lambda^2 2^{o(n\lambda/\epsilon)}$$

and assume such an algorithm A exists making q profile queries.

Step 2 (Algorithm B) Convert it to an algorithm B making q δ-distribution queries.

Step 3 (Algorithm C) Use Lemma 3.2 to derive an algorithm C finding $\epsilon_0\lambda$-WSNE in λ-Lipschitz games contradicting the lower bound of Corollary 3.6.

Proof. Assume that some such algorithm A exists finding ϵ-PNEs of games in $\mathcal{G}(n, 2, \lambda)$ making at most

$$\lambda^2 2^{o(n\lambda/\epsilon)}$$

profile queries. Consider any

$$\epsilon < \epsilon_0, \qquad \lambda' < \frac{\epsilon}{\sqrt{8n \log 4n}},$$

and define

$$\lambda = \frac{\epsilon}{\epsilon_0}, \qquad L = \frac{\lambda}{\lambda'}, \qquad N = Ln.$$

Over the course of three steps we derive an algorithm C (with an intermediate algorithm B) that contradicts Corollary 3.6.

Step 1: Algorithm A

Given a game $G'' \in \mathcal{G}\left(N, 2, \frac{3\lambda'}{2}\right)$, A can find an $\epsilon/2$-PNE of G'' with probability at least $2^{-o(N)}$ making at most

$$\left(\frac{3\lambda'}{2}\right)^2 2^{o\left(\frac{3N\lambda'/2}{\epsilon/2}\right)} = \lambda'^2 2^{o\left(N\lambda'/\epsilon\right)}$$

profile queries. In fact, the success probability can be amplified to constant by running A $O(N)$ times (since a pure approximate equilibrium is a poly-time checkable result).

Step 2: Algorithm B

Consider an algorithm that only has access to distribution queries. We will want to show that such an algorithm can still simulate A well enough to find an approximate equilibrium of its input.

Let

$$\delta = \frac{\epsilon_0 \lambda'}{4}.$$

For any game $G' \in \mathcal{G}(N, 2, \lambda')$, consider an algorithm making δ-distribution queries of *pure action profiles* of G' (introducing the uncertainty without querying mixed strategies). Since $\epsilon_0 < 1$ and $\lambda' < \epsilon$, δ satisfies the condition of Lemma 3.4 and thus this algorithm learns a game $G'' \in \mathcal{G}\left(N, 2, \frac{3\lambda'}{2}\right)$ for which every $\epsilon/2$-PNE is also an ϵ-PNE of G'.

So define the algorithm B that takes input G' and proceeds as though it is algorithm A (but makes δ-distribution queries instead). By the claim above, after at most

$$\lambda'^2 2^{o\left(N\lambda'/\epsilon\right)}$$

queries, it has found an $\epsilon/2$-PNE of some $G'' \in \mathcal{G}\left(N, 2, \frac{3\lambda'}{2}\right)$ that it believes it has learned, which is also guaranteed to be an ϵ-PNE of G' (we can still promise a constant success probability for this simulation). Figure 3.1 shows the acceptable outputs of B in blue and the possible outputs of B in

orange.

Step 3: Algorithm C

Consider any game $G \in \mathcal{G}(n, 2, \lambda)$, and let $G' = g_G(L)$ be the population game induced by G. There is an algorithm C described by Lemma 3.2 (and Remark 23) that takes input G and simulates B on G' (making

$$2n \cdot \lambda'^2 2^{o\left(N\lambda'/\epsilon\right)} = \delta^2 2^{o(n\lambda/\epsilon)}$$

δ-distribution queries) and correctly outputs an ϵ-WSNE (i.e. an $\epsilon_0\lambda$-WSNE) of G with constant success probability (so certainly at least $2^{-o(n)}$).

The existence of algorithm C directly contradicts the result of Corollary 3.6, proving that algorithm A cannot exist. □

Remark 24. Note that, if we instead start the proof with the assumption of such an algorithm B, we can also show a $\delta^2 2^{o(n\lambda/\epsilon)}$ lower bound for the δ-distribution query complexity of finding ϵ-PNEs of λ-Lipschitz games.

3.4 Multi-Lipschitz Games

In this section, we consider a generalization of Lipschitz games in which each player $i \in [n]$ has a "player-specific" Lipschitz value λ_i in the sense that, if player i changes actions, the payoffs of all other players are changed by at most λ_i.

This section will also utilize a more general population game $G' = g_G(L_1, \ldots, L_n)$ in which player i is replaced by a population of size L_i (where the L_i may differ from each other), and the queries in Lemma 3.2 become $(\delta, \min_{i\in[n]}\{\gamma/L_i\})$-distribution queries (this will now be relevant, as we need to apply Corollary 2.2).

3.4.1 Warmup

As a warmup, we consider a form of "typed" Lipschitz game. Consider a game with $n+1$ players - n "individual" traders with Lipschitz constant $\frac{1}{n}$, and one "institutional" trader with Lipschitz constant 1 (this player does not have a bounded impact on the other players). Call this game the "Institutional Trader Game".

Theorem 3.9. *Finding an ϵ-ANE of the Institutional Trader Game requires at most as many distribution queries as finding an ϵ-ANE of a game $G \in \mathcal{G}\left(2n, m, \frac{1}{2n}\right)$.*

Proof. Take an Institutional Trader Game G and apply the generalized population game reduction to obtain a game $G' = g_G(1, \ldots, 1, n)$ - the $2n$ player game in which the individual players are left untouched and the institutional player becomes a population of n individuals. If an ϵ-ANE of this population game is found, an ϵ-ANE of G can be backed out along the lines of Lemma 3.2. However, since $G' \in \mathcal{G}\left(2n, m, \frac{1}{2n}\right)$, this is sufficient to prove the theorem. □

Remark 25. To continue the warmup, we can expand the type of game we consider. For example, we can instead examine a game with n individual traders and N institutional traders. Along the same lines as the above, finding an equilibrium in such a game G is at most as hard as finding an equilibrium of a game $G' \in \mathcal{G}\left(n(N+1), m, \frac{1}{n(N+1)}\right)$.

3.4.2 Generalized Result

We now generalize from an example of a "typed" Lipschitz game to the concept of Multi-Lipschitz games, in which each player may have a different Lipschitz constant:

Definition 3.4.1 (Multi-Lipschitz Game)**.** A Λ-Multi-Lipschitz game G is an n-player, m-action game G in which each player $i \in [n]$ is associated with a constant $\lambda_i \leq 1$ such that

$$\frac{1}{n}\sum_{i'=1}^{n} \lambda_{i'} = \Lambda$$

and, for any player $i' \neq i$ and action profiles $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}$ with $\mathbf{a}^{(1)}_{-i} = \mathbf{a}^{(2)}_{-i}$,

$$\left|u_{i'}\left(\mathbf{a}^{(1)}\right) - u_{i'}\left(\mathbf{a}^{(2)}\right)\right| \leq \lambda_i.$$

The class of such games is denoted $\mathcal{G}_\Lambda(n, m)$, and for simplicity it is assumed that $\lambda_1 \leq \ldots \leq \lambda_n$.

The consideration of this generalized type of game allows real-world situations to be more accurately modeled. Geopolitical circumstances, for example, naturally take the form of Multi-Lipschitz games, since individual countries have different limits on how much their actions can affect the rest of the world. Financial markets present another instance of such games; they not only consist of individual traders who have little impact on each other, but also include a number of institutions

that might each have a much greater impact on the market as a whole. This consideration is further motivated by the recent GameStop frenzy [70]; the institutions still wield immense power, but so do the aggregate actions of millions of individuals.

Notice that a λ-Lipschitz game is also a λ-Multi-Lipschitz game. Equivalently, any algorithm that finds ϵ-ANEs of Λ-Multi-Lipschitz games is also applicable to Λ-Lipschitz games. Theorem 3.10 shows a kind of converse for query complexity, reducing from finding ϵ-ANE of Λ-Multi-Lipschitz games to finding ϵ-ANE of λ-Lipschitz games, for λ a constant multiple of Λ.

Theorem 3.10 (Multi-Lipschitz Reduction)**.** *If $\epsilon/2$-ANEs of games in $\mathcal{G}\left(2n, 2, \frac{3\Lambda}{2}\right)$ can be computed with probability at least $1-\eta$ by an algorithm making q queries, then ϵ-ANEs of games in $\mathcal{G}_\Lambda(n, 2)$ can be computed with probability at least $1-\eta$ by an algorithm making*

$$\text{poly}(n, \frac{1}{\epsilon}, \log\frac{1}{\eta})q\log q$$

queries.

As we now consider ϵ-ANEs, existence is no longer a question: such equilibria are *always* guaranteed to exist by Nash's Theorem [65]. Other than utilizing the more general population game reduction, the proof follows along similar lines to that of Theorem 3.8.

Proof. Consider some $\epsilon > 0$ and a game $G \in \mathcal{G}_\Lambda(n, 2)$ (WLOG take $\lambda_1 \leq \ldots \leq \lambda_n$). First, if $\Lambda < \epsilon/n^2$, finding an ϵ-ANE is trivial (each player can play their best-response to the uniform mixed strategy, found in $2n$ queries). So assume $\Lambda \geq \epsilon/n^2$. Define $L_i = \max\{\frac{\lambda_i}{\Lambda}, 1\}$ and, taking $i' = \max_{i\in[n]}\{i : L_i = 1\}$, note that

$$\sum_{i=1}^{n} L_i = \sum_{i=1}^{i'} 1 + \sum_{i=i'+1}^{n} \frac{\lambda_i}{\Lambda} = \sum_{i=1}^{i'} 1 + \frac{1}{\Lambda}\sum_{i=i'+1}^{n} \lambda_i \leq \sum_{i=1}^{i'} 1 + \frac{n\Lambda}{\Lambda} \leq 2n.$$

Thus the population game $G' = g_G(L_1, \ldots, L_n) \in \mathcal{G}(2n, 2, \Lambda)$.

Step 1: Algorithm A

Consider an algorithm A that finds $\epsilon/2$-ANEs of games in $\mathcal{G}\left(2n, 2, \frac{3\Lambda}{2}\right)$, with probability at least $1-\eta$ making q profile queries.

Step 2: Algorithm B

Taking

$$\delta = \frac{\epsilon^2}{4n^2} \leq \frac{\epsilon\Lambda}{4},$$

we can once again apply Lemma 3.4 to construct the same algorithm B from the proof of Theorem 3.8 that simulates A but makes $(\delta, 1)$-distribution queries and finds an ϵ-ANE of G'.

Step 3: Algorithm C

By Lemma 3.2, there is an algorithm C on input $G \in \mathcal{G}_\Lambda(n, 2)$ that simulates B (replacing each $(\delta, 1)$-distribution query of G' with $2n$ $(\delta, \frac{1}{2n})$-distribution queries of G since $1/L_n \geq 1/2n$) finding an ϵ-ANE of G' (and thus of G) with probability at least $1 - \eta$.

Applying Corollary 2.2 (using $\delta = \frac{\epsilon^2}{4n^2}, \gamma = \frac{1}{2n}$) to create a profile-query algorithm from C completes the proof. □

As an example application of Theorem 3.10, an algorithm of [41] finds $\left(\frac{1}{8} + \alpha\right)$-approximate Nash equilibria of games in $\mathcal{G}\left(n, 2, \frac{1}{n}\right)$; Theorem 3.11 states that result in detail, and Corollary 3.12 extends it to Multi-Lipschitz games.

Theorem 3.11 ([41], Theorem 2.31 Restated)**.** *Given constants $\alpha, \eta > 0$, there is a randomized algorithm that, with probability at least $1 - \eta$, finds $\left(\frac{1}{8} + \alpha\right)$-approximate Nash equilibria of games in $\mathcal{G}\left(n, 2, \frac{1}{n}\right)$ making $O\left(\frac{1}{\alpha^4} \log\left(\frac{n}{\alpha\eta}\right)\right)$ profile queries.*

We now have some ability to apply this to Multi-Lipschitz games; for a small range of values we can improve upon the trivial $\frac{1}{2}$-approximate equilibrium of Proposition 6.1.

Corollary 3.12. *For $\alpha, \eta > 0, \Lambda \geq \frac{1}{n}, \epsilon \geq \frac{n\Lambda}{8} + \alpha$, there is an algorithm finding ϵ-ANEs of games in $\mathcal{G}_\Lambda(n, 2)$ with probability at least $1 - \eta$ making at most $\text{poly}(n, \frac{1}{\alpha}, \log\frac{1}{\eta})$ profile queries.*

Remark 26. This is actually a slight improvement over just combining Theorems 3.10 and 3.11, since the choice of δ can be made slightly smaller to shrink α as necessary.

Chapter 4

Time Complexity

4.1 Overview

The material of this chapter was presented at the 15th International Symposium on Algorithmic Game Theory. In this chapter, the problem we address is, for $\lambda \leq \frac{\epsilon}{\sqrt{8n \log 4n}}$:

> *Given as input an n-player game, find either an ϵ-approximate pure Nash equilibrium, or a witness that the game is* not *λ-Lipschitz for the above λ (a witness consists of two pure-strategy profiles whose payoffs show the game is not λ-Lipschitz).*

We will consider m, ϵ, and λ to be parameters of the problem statement rather than provided in the input, however the problem is similar if these are instead passed in as input.

Notice that the two alternative kinds of solutions (an equilibrium *or* a witness against the λ-Lipschitz property) avoids a promise problem: there is no hard-to-check promise that a given game is Lipschitz. The problem statement can be taken to apply in general to any class of games whose payoff functions are efficiently computable (e.g. circuit games [76]). Efficient computability of payoffs is also a sufficient condition to guarantee that solutions are easy to check, allowing us to check the validity of either kind of solution. In the special case of Lipschitz *polymatrix* games of interest here, they can be presented in a straightforward format for which we can check the λ-Lipschitz property.

A **PPAD**-completeness result consists of containment in **PPAD** as well as **PPAD**-hardness. In the context of *mixed-strategy* Nash equilibrium computation, the following property of a class of games is sufficient for containment in **PPAD**:

Given a mixed-strategy profile, it is possible to deterministically compute the payoffs with additive error $O(\epsilon)$, in time polynomial in the representation size of the game, and $1/\epsilon$.

This property is shared by most well-studied classes of multiplayer games, such as graphical games or polymatrix games. It does not hold for unrestricted Lipschitz games[1], but it of course holds for Lipschitz polymatrix games (these being a subset of the set of polymatrix games). In this chapter we obtain **PPAD** containment for *pure-strategy* approximate equilibria. The approach of [2] places the problem in some sense of "randomized **PPAD**" (Section 4.3). We establish containment in **PPAD** via a *deterministic* approach to selecting a pure approximate equilibrium (Lemma 4.4). Our approach requires λ to be a smaller-growing function of ϵ and n than that of [2], however there *is* overlap between the (ϵ, λ) régime where the problem is contained in (deterministic) **PPAD** and the one where the problem is hard for **PPAD**.

PPAD-hardness applies to the binary-action case where each player has two pure strategies $\{1, 2\}$. We reduce from the polymatrix games of [73], by considering an induced *population game* (as described in Section 3.2.1), which has the Lipschitz property.

4.1.1 A Comparison with the Previous Chapter

Chapter 3 showed an exponential query lower-bound for finding approximate pure Nash equilibria in Lipschitz games. To contrast that result, the games considered in this chapter require very few queries to learn completely.

Proposition 4.1. The deterministic profile query complexity of (m, ϵ, λ)-PureLPG is $O(n^2m^2)$.

Proof. In order to learn a polymatrix game completely, one must only learn the values of $\beta_{i,i',j,j'}$ (or, at least, consistent values). To fix the degrees of freedom one can query the payoff to $\mathbf{0}$, assign $u_i(\mathbf{0})$ as $\beta_{i,i+1,0,0}$ (or $\beta_{n,1,0,0}$ when $i = n$), and set the remaining $\beta_{i,i',0,0} = 0$. From here, the remaining values can each be obtained by a single query:

$$\beta_{i,i',j,j'} = u_i(\mathbf{0}_{-\{i,i'\}}, j, j') - \beta_{i,i',0,0}$$

where $u_i(\mathbf{0}_{-\{i,i'\}}, j, j')$ is the payoff to player i of the action profile in which player i plays action j, player i' plays action j', and all other players continue to play action 0.

[1] For Lipschitz games presented in terms of unrestricted circuits that compute payoffs from pure-strategy profiles, it does not hold unless $\mathbf{P} = \mathbf{RP}$. For games abstractly presented via a payoff query oracle, the property unconditionally does not hold.

In total, this algorithm requires one computation for each value of $\beta_{i,i',j,j'}$ where j and j' are both nonzero, and a single query for the others, for a total profile query complexity of

$$n^2(m^2-1)+1,$$

or $O(n^2m^2)$. □

In fact, Lipschitz polymatrix games are a strict subset of Lipschitz games which require only $O(n^2m^2)$ space to store (simply store each value $\beta_{i,i',j,j'}$). When combined with the main results of this chapter, the proposition above indicates the surprising phenomenon that, while this restrictive subclass of Lipschitz games can be very easily learned and stored, it is still likely impossible to find approximate pure equilibria in polynomial time.

Furthermore, as defined, a polymatrix game is λ-Lipschitz for any

$$\lambda \geq \max_{i,i' \in [n], j, j'_1, j'_2 \in [m]} \left| \beta_{i,i',j,j'_1} - \beta_{i,i',j,j'_2} \right|.$$

Clearly Proposition 4.1 also shows that the Lipschitz property can be checked by querying $O(n^2m^2)$ payoffs and running $O(n^2m^3)$ subtractions. Furthermore, if i plays j, i's payoff is a linear function of the indicator variables of the other player-strategy pairs. $\beta_{i,i',j,j'}$ represents a small contribution, possibly negative, that i' makes to i (when i plays j) by playing j'. To be valid, all total payoffs must lie in $[0,1]$, which can be checked from a game presented via the quantities $\beta_{i,i',j,j'}$ by performing the following additional checks for every $i \in [n], j \in [m]$:

$$\sum_{i' \neq i} \max_{j' \in [m]} \beta_{i,i',j,j'} \leq 1$$

$$\sum_{i' \neq i} \min_{j' \in [m]} \beta_{i,i',j,j'} \geq 0$$

There are $2nm$ such calculations, each of which requires $O(nm)$ operations, so an invalid game can be uncovered in polynomial time.

Remark 27. The study of Lipschitz polymatrix games was motivated by a desire to uncover a polynomial-time algorithm for finding equilibria in a heavily restricted subset of Lipschitz games. However, this clearly did not turn out to exist, once again due to the population game reduction.

4.2 PPAD-Completeness

The result we achieve in this chapter is as follows:

Theorem 4.1 (Main Result)**.** *For every constant $m \geq 2$, there exists some constant $\epsilon > 0$ such that, for all functions $\lambda(n) = \Theta(n^{-3/4})$, (m, ϵ, λ)-*PureLPG *is* **PPAD***-complete.*

In fact, for any constant $\alpha \in (\frac{2}{3}, 1)$, Theorem 4.1 holds for $\lambda(n) = \Theta(n^{-\alpha})$. In order to prove Theorem 4.1, we will need to show both containment in **PPAD** and **PPAD**-hardness under these settings of the parameters.

4.2.1 Containment in PPAD

Theorem 4.2. *For all functions $\epsilon(n), \lambda(n)$ satisfying*

$$\lambda(n) = \frac{1}{\text{poly}(n)}, \qquad \epsilon = \lambda(n)\omega(n^{2/3})$$

(m, ϵ, λ)-PureLPG $\in$ **PPAD**.

In particular, we prove this when $\epsilon(n) \geq 6\lambda(n)\sqrt[3]{n^2 m \log 3m}$.

Warmup

We will begin with the proof for $m = 2$. The proof of this theorem is broken down further into two smaller steps. First, we will exhibit sufficient settings of the parameters such that *mixed* approximate Nash equilibria of Lipschitz polymatrix games can be found in **PPAD**. Second, with the parameters set as in the statement of Theorem 4.2, we describe a derandomization technique for deriving a *pure* approximate equilibrium from the mixed one in polynomial time. More formally:

Lemma 4.3. *Whenever $\lambda(n) = \frac{1}{\text{poly}(n)}$, $(2, \frac{\lambda}{8}, \lambda)$-*MixedLPG $\in$ **PPAD**.

Sketch. As in the standard proof placing the problem of finding approximate Nash equilibria in **PPAD** (e.g. Theorem 3.1 of [24]), since ϵ is an inverse polynomial in n, the key requirement is the existence of an efficient deterministic algorithm answering mixed payoff queries. Because the payoffs to a player i in polymatrix games are linear in the actions of every other player, any mixed payoff query can be answered in time $O(n^2m^2)$ (or simply $O(n^2)$ since we are keeping the number of actions fixed). Thus $(2, \frac{\lambda}{8}, \lambda(n))$-MixedLPG $\in$ **PPAD**. □

Lemma 4.4 (Main Technical Lemma)**.** *For functions $\epsilon(n), \lambda(n)$ satisfying $\epsilon(n) \geq \lambda\sqrt[3]{70n^2}$, an ϵ-PNE of an n-player, binary-action, λ-Lipschitz polymatrix game G can be derived from an $\frac{\lambda}{8}$-ANE of G in polynomial time.*

Proof. Throughout this proof we implicitly use the assumption that payoffs from mixed strategy profiles can be computed in polynomial time. We will consider the (signed) discrepancy

$$d_i(\mathbf{p}) = u_i(2, \mathbf{p}_{-i}) - u_i(1, \mathbf{p}_{-i})$$

(d_i is a 2λ-Lipschitz function via the same argument as Remark 1).

Take some $\frac{\lambda}{8}$-ANE $\mathbf{p}^*$ of G. The reduction is performed in three steps:

(1) Constructing from $\mathbf{p}^*$ a well-supported Nash equilibrium $\mathbf{p}^{(0)}$. This step is required because we will use a player's discrepancy as a linear proxy for regret, and general approximate equilibria may allow players to allocate small probabilities to bad actions, thus decoupling the two measures (in contrast with exact equilibria, in which any mixed strategy by definition implies no discrepancy between the actions). Well-supported equilibria do not suffer from this same decoupling.

(2) Iteratively converting $\mathbf{p}^{(0)}$ into a pure action profile $\mathbf{a}$ that is "close" to an equilibrium. To achieve this step, we will bound the rate at which the sum of all players' discrepancies is allowed to increase, ensuring that few players experience large regret.

(3) Correcting $\mathbf{a}$ to obtain an approximate pure equilibrium $\mathbf{a}^*$. Because of the guarantees from the previous step, we can achieve this by allowing every player with high regret to change to their best response.

Step 1: Mixed equilibrium to WSNE

In this first step we construct WSNE $\mathbf{p}^{(0)}$ from $\mathbf{p}^*$. The idea here is to ensure that all players fall into one of the following two categories:

- Players with discrepancy that is small in absolute value
- Players playing pure strategies

In order to do this, consider any player i such that $|d_i(\mathbf{p}^*)| > \frac{\lambda}{2}\sqrt{n}$ (without loss of generality we can assume the discrepancy is negative, i.e. action 1 is significantly better). Then

$$\mathrm{reg}_i(\mathbf{p}^*) = -p_i^* d_i > p_i^* \frac{\lambda}{2}\sqrt{n}$$

so, since $\mathbf{p}^*$ is an approximate equilibrium,

$$p_i^* < \frac{1}{4\sqrt{n}}.$$

If every such player (at most n) changes to playing their pure best response, then every player's discrepancy will increase by at most $2n\lambda p_i^*$ (recall that discrepancy is 2λ-Lipschitz), so in this new profile, no player playing a mixed strategy can experience discrepancy greater than

$$\frac{\lambda}{2}\sqrt{n} + 2n\lambda\frac{1}{4\sqrt{n}} = \lambda\sqrt{n}$$

while no player playing a pure strategy can experience regret greater than

$$0 + 2n\lambda\frac{1}{4\sqrt{n}} = \frac{\lambda}{2}\sqrt{n}.$$

Thus this new profile is a $\lambda\sqrt{n}$-WSNE, and for any player i playing a mixed strategy, $d_i(\mathbf{p}^*) \leq \lambda\sqrt{n}$. Call this $\lambda\sqrt{n}$-WSNE $\mathbf{p}^{(0)}$.

Step 2: WSNE to pure profile

In this step, we iteratively define intermediate strategy profiles $\mathbf{p}^{(i)}$ and intermediate sets of players $S^{(i)}$ who have, at any point, experienced small discrepancy (defined below to ensure that players not in $S^{(i)}$ cannot have high regret in profile $\mathbf{p}^{(i)}$). Informally, for every player i, $\mathbf{p}^{(i)}$ is a strategy profile in which players $1, \ldots, i$ are playing pure strategies and players $i+1, \ldots, n$ may not be. Furthermore, the set $S^{(i)}$ is the set of players who, in at least one of the profiles $\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(i)}$ have experienced discrepancy at most $\lambda\sqrt{n}$. These will be iteratively computed to ultimately arrive at $\mathbf{a} = \mathbf{p}^{(n)}$: a pure action profile that is in some sense "close" to an approximate pure equilibrium.

Counterintutively, this will not always involve assigning each player to play their best response. Instead, we will instruct each player to play the pure strategy that allows us to bound the effect on the other players. Roughly, we identify below a measure of badness C that is quadratic in the probability

p assigned by a player to strategy 2 as opposed to strategy 1. In setting p to 0 or 1, we minimize the linear term in this measure, choosing 0 or 1 depending on the sign of the derivative with respect to p, and the measure increases by the relatively small second-order term.

More formally, take $\mathbf{p}^{(0)}$ as above and define

$$S^{(0)} = \{i' : |d_{i'}(\mathbf{p}^{(0)})| \leq \lambda\sqrt{n}\}$$

(i.e. a superset of the players playing mixed strategies in $\mathbf{p}^{(0)}$) while

$$S^{(i)} = S^{(i-1)} \cup \{i' : |d_{i'}(\mathbf{p}^{(i)})| \leq \lambda\sqrt{n}\}$$

(adding at each step the set of players playing pure strategies in $\mathbf{p}^{(0)}$ but experiencing small discrepancy for the first time at step i; note that, due to the Lipschitz property, no player's discrepancy can change sign without that player joining $S^{(i)}$, so any player not in $S^{(i)}$ must be playing their pure best response). Furthermore, define the sum of squared discrepancies cost function (which considers only the players in $S^{(i)}$):

$$C(\mathbf{p}^{(i)}) = \sum_{i' \in S^{(i)}} d_{i'}^2(\mathbf{p}^{(i)}).$$

Note that C only considers players in the set $S^{(i)}$, as it is meant to be a proxy for how far we are from equilibrium, and players in $[n] \setminus S^{(i)}$ have high discrepancies but are playing their best responses and hence should not contribute to this distance to equilibrium.

It is difficult to minimize C itself, and regardless C is already only a proxy for the true objective we must minimize to achieve an approximate pure equilibrium. However, since $d_{i'}$ is a multi-linear function (in fact, in the case of polymatrix games, linear), we can write

$$d_{i'}(\mathbf{p}^{(i)}) = c + \ell p_i$$

where c is some constant dependent only on $\mathbf{p}_{-i}^{(i)}$ and ℓ is a coefficient with absolute value at most 2λ (the Lipschitz parameter of $d_{i'}$). Squaring this, and keeping in mind that $\mathbf{p}_{-i}^{(i-1)} = \mathbf{p}_{-i}^{(i)}$, we can expand (for every $i' \in S^{(i-1)}$):

$$d_{i'}^2(\mathbf{p}^{(i)}) - d_{i'}^2(\mathbf{p}^{(i-1)}) = \alpha(p_i^{(i)} - p_i^{(i-1)}) + \ell^2((p_i^{(i)})^2 - (p_i^{(i-1)})^2)$$

for some value $\alpha_i = 2c\ell$ dependent only on $\mathbf{p}_{-i}^{(i)}$. Taking the sum over all players $i' \in S^{(i)}$, this yields

$$C(\mathbf{p}^{(i)}) - C(\mathbf{p}^{(i-1)}) = A(p_i^{(i)} - p_i^{(i-1)}) + \Lambda((p_i^{(i)})^2 - (p_i^{(i-1)})^2) + K_i$$

where

- A may be negative or non-negative.
- $\Lambda \leq 4\lambda^2 n$ (and obviously $(p_i^{(i)})^2 - (p_i^{(i-1)})^2 \leq 1$).
- K_i represents the one-time contribution of at most $\lambda^2 n$ each from any player in $S^{(i)} \setminus S^{(i-1)}$ whose discrepancy will be counted in this and all future costs. While we can't say anything about each individual K_i (since we don't know when exactly every player's discrepancy will dip to this level), we note that there are at most n such contributions of $\lambda^2 n$, i.e:

  $$\sum_{i \in [n]} K_i \leq \lambda^2 n^2$$

  (this bound is the reason we needed to start from a WSNE).

If $p_i^{(i-1)} \in \{0, 1\}$, then set $\mathbf{p}^{(i)} = \mathbf{p}^{(i-1)}$ (if a player is already playing a pure strategy, we can avoid adding to C at all on their turn). Otherwise, since we can efficiently compute A and decide whether it is negative or not, we can select $p_i^{(i)}$ such that $A(p_i^{(i)} - p_i^{(i-1)}) \leq 0$ (one such option for $p_i^{(i)} \in \{0, 1\}$ must exist). This bounds

$$C(\mathbf{p}^{(i)}) - C(\mathbf{p}^{(i-1)}) \leq 4\lambda^2 n + K_i$$

and a quick induction achieves

$$C(\mathbf{p}^{(n)}) \leq 4\lambda^2 n^2 + \sum_{i \in [n]} K_i \leq 4\lambda^2 n^2 + \lambda^2 n^2 = 5\lambda^2 n^2.$$

Furthermore, $\mathbf{a} = \mathbf{p}^{(n)}$ is a pure action profile.

Step 3: Pure profile to pure equilibrium

Here we distinguish between players in $S^{(n)}$ and players not in $S^{(n)}$.

The players not in $S^{(n)}$ must be playing their best response in $\mathbf{a}$. This is because they were playing their best response in $\mathbf{p}^{(0)}$, and their discrepancy never approached 0 or changed sign, so their best response must never have changed. In particular, therefore, none of these players suffer any regret.

We now consider the players in $S^{(n)}$. Note that we have bounded the sum of the squares of the discrepancies of the players in this set to be at most $5\lambda^2 n^2$. We can now also bound the number of players with regrets that are too high and fix them. To begin this final step, define $\delta = \lambda\sqrt[3]{20n^2}$ and consider the number of players $i \in S^{(n)}$ such that $d_i(\mathbf{a}) > \delta$. This will be at most $\frac{C(\mathbf{a})}{\delta^2}$. So have every player with *regret* at least δ in profile $\mathbf{a}$ (a subset of those with high *discrepancy*) simultaneously switch actions. Then, once more invoking Remark 1, the maximum regret any player (in $S^{(n)}$ or otherwise) can experience is

$$\delta + 2\lambda\frac{C(\mathbf{a})}{\delta^2} \leq \delta + \frac{10\lambda^3 n^2}{\delta^2} < \lambda\sqrt[3]{70n^2}.$$

In other words, the action profile $\mathbf{a}^*$ obtained after these players switch actions is a $\lambda\sqrt[3]{70n^2}$-PNE. This completes the proof of Lemma 4.4. □

Finally, because a violation of the Lipschitz property can be uncovered in a quick check before commencing the above algorithm simply by ensuring all the coefficients are smaller than λ (as described in Section 4.1.1), and [12] show that **PPAD** is closed under Turing reductions, the binary-action version of Theorem 4.2 clearly follows from combining the two Lemmas above.

Containment in PPAD for Additional Actions

We will now modify the proof of Theorem 4.2 to values of $m > 2$. Once again, the proof of this theorem is broken down further into two smaller steps:

Lemma 4.5 (Extension of Lemma 4.3)**.** *For any fixed $m \geq 2, \lambda(n) = \frac{1}{\text{poly}(n)}$,*
*$(m, \left(\frac{m-1}{m}\right)^2\lambda, \lambda)$-MixedLPG $\in$ **PPAD**.*

The proof of Lemma 4.5 is essentially the same as that of Lemma 4.3.

Lemma 4.6 (Extension of Lemma 4.4)**.** *For functions $\epsilon(n), \lambda(n)$ satisfying*

$$\epsilon(n) \geq 6\lambda(n)\sqrt[3]{n^2 m \log 3m}$$

an $\epsilon(n)$-PNE of an n-player, m-action, λ-Lipschitz polymatrix game G can be derived from an $\left(\frac{m-1}{m}\right)^2\lambda$-ANE of G in polynomial time.

Proof. We will use the following values in this proof. Let

$$\epsilon_0 = \left(\frac{m-1}{m}\right)^2 \lambda$$

$$\epsilon_1 = 2\sqrt{2n\lambda\epsilon_0}.$$

Then, in order to prove Lemma 4.6, we need to convert an ϵ_0-ANE to an ϵ-PNE. We will do so in the same three steps as in the proof of Lemma 4.4, but more care must be taken when generalizing the proof.

The primary generalization required is the transition from considering discrepancy to considering variance. Whereas in Lemma 4.4 there was a well-defined notion of discrepancy that could be expressed as quadratic in the probabilities assigned to the actions of a given player, such a concept does not easily translate to games with more actions. In order to rely on similar techniques, we will instead consider the variance of a subset of each player's actions. The exact subset to be considered will be described below, and throughout this proof will be referred to as the "relevant" set, as it will contain the set of actions that may become relevant in our analysis of the algorithm.

We will proceed along the same lines as the proof of Lemma 4.4 with the modified approach.

Step 1: Mixed equilibrium to WSNE

We begin with an ϵ_0-ANE $\mathbf{p}^*$. Consider any player i and assume that player i puts probability p on actions with regret greater than

$$\delta_0 = \sqrt{2(n-1)\lambda\epsilon_0}.$$

The regret of this player is at least $p\delta_0$, so it must be the case that $p \leq \epsilon_0/\delta_0$.

This being the case, if every player simultaneously moves the probability from such actions to their best response, then any given player's worst-case regret becomes

$$\delta_0 + 2(n-1)\lambda p \leq \delta_0 + \frac{2(n-1)\lambda\epsilon_0}{\delta_0} \leq 2\sqrt{2n\lambda\epsilon_0} = \epsilon_1.$$

Thus an ϵ_0-ANE can be converted easily to an ϵ_1-WSNE.

Step 2: WSNE to pure profile

This is, once again, the most involved step. Unlike Step 1 (which occurred simultaneously) and Step 3 (which will do the same), this step will take place for each player in turn. We will iteratively define

intermediate strategy profiles $\mathbf{p}^{(i')}$ in which players $1, \ldots, i'$ are playing pure strategies and players $i'+1, \ldots, n$ may not be. Ultimately, we will arrive at $\mathbf{a} = \mathbf{p}^{(n)}$: a pure action profile that is in some sense "close" to an approximate pure equilibrium.

Counterintuitively, this will not always involve instructing each player to play their best response when their turn arises. Instead, we will instruct each player to play the pure strategy that allows us to bound its effect on the other players. Roughly, we identify a measure of badness that is quadratic in the probability assigned by a player to each action. In selecting a pure action, we minimize the linear term in this measure depending on the direction of the gradient, and the measure increases by the relatively small second-order term. The quadratic measure we select will be the sum of the variances of subsets of the players' actions.

More specifically, we define the following concepts:

Notation	Definition
$S_i^{(i')}$	Player i's **relevant set** of actions (defined below) after step i'
$m_i^{(i')}$	$\left\lvert S_i^{(i')} \right\rvert$, i.e. the number of relevant actions player i has after step i'
$\mathbf{u}_i(S,t)$	the length-$\lvert S \rvert$ restriction of player i's payoff vector after step t to the actions in set S
$\mu_i(S,t)$	the average value of $\mathbf{u}_i(S,t)$, i.e. the payoff to player i after step t for playing the uniform mixed strategy over all actions in player set S
$\sigma_i^2(S,t)$	the variance of the values in $\mathbf{u}_i(S,t)$
$\mathrm{reg}_i(j,t)$	the regret player i experiences for playing action j after step t

Now, we will define the relevant set via the following recurrence relation. The base case will be:

$$S_i^{(0)} = \{j \mid \mathrm{reg}_i(j,0) \leq \epsilon_1\}.$$

The recurrence is:

Algorithm 2: Relevant Set Recurrence

$S_i^{(i'+1)} = S_i^{(i')}$

$\mu = \mu_i(S_i^{(i'+1)}, i'+1)$

while $(\max_{j \notin S_i^{(i'+1)}} u_i(j, i'+1) \geq \mu)$:

  $S_i^{(i'+1)} = S_i^{(i'+1)} \cup \{\arg\max_{j \notin S_i^{(i'+1)}} u_i(j, i'+1)\}$

  $\mu = \mu_i(S_i^{(i'+1)}, i'+1)$

In other words, $S_i^{(i'+1)}$ is the superset of $S_i^{(i')}$ obtained by repeatedly adding player i's highest-paying action to $S_i^{(i'+1)}$ and recalculating the value of $\mu_i(S_i^{(i'+1)}, i'+1)$ until doing so would decrease its value. We call $S_i^{(i')}$ the relevant set because it consists of the actions that, at any point during the algorithm's run, are worth consideration in our attempt to bound the total regret experienced by all players.

Step 2 of the algorithm will thus proceed as follows. During step i', player i' will first change their strategy to the pure strategy that has the least impact on the linear term of the sum of the variances of all other players' relevant sets. *Then* every player will recalculate their relevant sets using the new payoffs considering the changed strategy of player i'.

There are three values we need to bound here:

1. The sum of the variances of $S_i^{(0)}$, the players' initial relevant sets:

$$\sum_{i=1}^{n} \sigma_i^2 \left(S_i^{(0)}, 0\right).$$

2. How much larger the sum of the variances of $S_i^{(i'+1)}$ can be than the sum of the variances of $S_i^{(i')}$ (between steps), adding to a total over all n steps of:

$$\sum_{i=1}^{n} \sum_{i'=1}^{n} \sigma_i^2 \left(S_i^{(i')}, i'\right) - \sigma_i^2 \left(S_i^{(i'-1)}, i'\right).$$

3. How much the sum of the variances of $S_i^{(i')}$, can increase during step i', adding to a total over all n steps of:

$$\sum_{i=1}^{n} \sum_{i'=1}^{n} \sigma_i^2 \left(S_i^{(i'-1)}, i'\right) - \sigma_i^2 \left(S_i^{(i'-1)}, i'-1\right).$$

Note that the sum of these three values is

$$\sum_{i=1}^{n} \sigma_i^2 \left(S_i^{(n)}, n \right)$$

i.e. the sum of the variances of the relevant sets at the end of the entire Step 2.

We begin with quantity (1). The variance of any set of points within the interval $[a, b]$ is at most $(b-a)^2/4$ (see, for example, [9]), so quantity (1) is at most

$$n\frac{\epsilon_1^2}{4} = 2n^2\lambda\epsilon_0 = 2\left(\frac{n\lambda(m-1)}{m}\right)^2.$$

Next, we bound quantity (2). Quantity (2) considers no change in the strategy profile - all payoffs considered are based on strategy $\mathbf{p}^{(i')}$. Then, for every player, actions are added to the set $S_i^{(i')}$ one by one (each time increasing the average value of the payoffs) to form the set $S_i^{(i'+1)}$. The critical observation is that, in addition to providing a lower bound (the current average) on the payoff of this newly added action, we can also ensure an upper bound of $\mu_i \left(S_i^{(i')}, i' \right) + 2\lambda$, as the action must not have been relevant in the previous step. So specifically, consider a set of k points $\{x_1, \ldots, x_k\}$ with mean μ and variance σ^2 and consider adding a point x_{k+1} that is within an additive 2λ of μ. We calculate the change in variance below, keeping in mind that if we subtract μ from every point the variance remains the same:

$$\begin{aligned}
&\mathrm{Var}(x_1, \ldots, x_{k+1}) - \sigma^2 \\
&= \mathrm{Var}(x_1 - \mu, \ldots, x_{k+1} - \mu) - \sigma^2 \\
&= \left(\frac{1}{k+1} \sum_{j=1}^{k+1} (x_j - \mu)^2 \right) - \left(\frac{1}{k+1} \left(\sum_{j=1}^{k} (x_j - \mu) \right)^2 \right) - \sigma^2 \\
&= \left(\frac{1}{k+1} \left((x_{k+1} - \mu)^2 + \sum_{j=1}^{k} (x_j - \mu)^2 \right) \right) \\
&\quad - \left(\frac{1}{k+1} \left((x_{k+1} - \mu) + \sum_{j=1}^{k} (x_j - \mu) \right)^2 \right) - \sigma^2 \\
&= \frac{(x_{k+1} - \mu)^2}{k+1} + \frac{k}{k+1}\sigma^2 - \frac{(x_{k+1} - \mu)^2}{(k+1)^2} - \sigma^2 \\
&= \frac{1}{k+1} \left(\frac{k}{k+1} (x_{k+1} - \mu)^2 - \sigma^2 \right)
\end{aligned}$$

where we can go from step 4 to step 5 using the fact that

$$\sum_{j=1}^{k}(x_j - \mu) = 0.$$

So, since $|x_{k+1} - \mu| \leq 2\lambda$, the largest possible increase in variance is

$$\frac{4k\lambda^2}{(k+1)^2}$$

(and, in fact, adding a point will usually *decrease* the variance, but there's no need to get that specific because this bounded increase will serve our purpose for constant values of m). If we add this contribution for the maximum possible $m-1$ actions (the relevant set will never start empty) and all n players, we see

$$\sum_{i=1}^{n}\sum_{k=1}^{m-1}\frac{4k\lambda^2}{(k+1)^2} \leq n\sum_{k=1}^{m-1}\frac{4\lambda^2}{k} = 4n\lambda^2 H_{m-1} < 4n\lambda^2(\log(m-1)+1).$$

Finally, we need to bound quantity (3). Note that, given $\mathbf{u}_i\left(S_i^{(i'-1)}, i'-1\right)$, the quantity $\mathbf{u}_i\left(S_i^{(i'-1)}, i'\right)$ and its mean and variance are all functions of the strategy vector $\mathbf{p}_{i'}$ of player i'. In particular, we can write:

$$\mathbf{u}_i(S_i^{(i'-1)}, i') - \mathbf{1}\mu_i(S_i^{(i'-1)}, i') = \mathbf{c}_i^{(i')} + \mathbf{L}_i^{(i')}\mathbf{p}_{i'}^{(i')}$$

for some constant $m_i^{(i'-1)} \times 1$ vector $\mathbf{c}_i^{(i')}$ (which may depend on the known values $\mathbf{u}_i\left(S_i^{(i'-1)}, i'-1\right)$) and $m_i^{(i'-1)} \times m$ coefficient matrix $\mathbf{L}_i^{(i')}$ in which each element has magnitude no more than $\frac{m-1}{m}\lambda$. Propagating this expression:

$$\begin{aligned}\sigma_i^2(S_i^{(i'-1)}, i') &= \frac{1}{m_i^{(i'-1)}}\left|\left|\mathbf{u}_i(S_i^{(i'-1)}, i') - \mathbf{1}\mu_i(S_i^{(i'-1)}, i')\right|\right|_2^2 \\ &= \frac{1}{m_i^{(i'-1)}}\left(\left|\left|\mathbf{c}_i^{(i')}\right|\right|_2^2 + \left|\left|\mathbf{L}_i^{(i')}\mathbf{p}_{i'}^{(i')}\right|\right|_2^2 + 2\mathbf{c}_i^{(i')}\cdot\mathbf{L}_i^{(i')}\mathbf{p}_{i'}^{(i')}\right). \qquad (4.1)\end{aligned}$$

If we specifically consider the mixed term:

$$\begin{aligned} 2\mathbf{c}_i^{(i')} \cdot \mathbf{L}_i^{(i')} \mathbf{p}_{i'}^{(i')} &= 2\mathbf{c}_i^{(i)'\intercal} \left(\mathbf{L}_i^{(i')} \mathbf{p}_{i'}^{(i')} \right) \\ &= 2 \left(\mathbf{c}_i^{(i')\intercal} \mathbf{L}_i^{(i')} \right) \mathbf{p}_{i'}^{(i')} \\ &= \mathbf{b}_i^{(i')\intercal} \mathbf{p}_{i'}^{(i')} \end{aligned} \tag{4.2}$$

for some vector $\mathbf{b}_i^{(i')} \in \mathbb{R}^m$. So now we can consider the quantity we're really after. We focus on (for some $i' \in [n]$) the sum over all players $i \in [n]$ of the changes in $\sigma_i^2 \left(S_i^{(i'-1)}, \cdot \right)$ after step i':

$$\sum_{i=1}^{n} \sigma_i^2 \left(S_i^{(i'-1)}, i' \right) - \sigma_i^2 \left(S_i^{(i'-1)}, i' - 1 \right). \tag{4.3}$$

We want to minimize this value as best we can, as our goal is to end up with small variance among the set of relevant actions. While the variance is not a very good proxy for the regret (see Equation 4.5 below), it is a value that we will be able to successfully bound. So, substituting Equations 4.1 and 4.2 into Equation 4.3, we see that the change in this value across step i' is

$$\begin{aligned} &\sum_{i=1}^{n} \frac{1}{m_i^{(i'-1)}} \left(\left|\left| \mathbf{c}_i^{(i')} \right|\right|_2^2 + \left|\left| \mathbf{L}_i^{(i')} \mathbf{p}_{i'}^{(i')} \right|\right|_2^2 + \mathbf{b}_i^{(i')\intercal} \mathbf{p}_{i'}^{(i')} - \left|\left| \mathbf{c}_i^{(i')} \right|\right|_2^2 - \left|\left| \mathbf{L}_i^{(i')} \mathbf{p}_{i'}^{(i'-1)} \right|\right|_2^2 - \mathbf{b}_i^{(i')\intercal} \mathbf{p}_{i'}^{(i'-1)} \right) \\ &= \sum_{i=1}^{n} \frac{1}{m_i^{(i'-1)}} \left(\left|\left| \mathbf{L}_i^{(i')} \mathbf{p}_{i'}^{(i')} \right|\right|_2^2 + \mathbf{b}_i^{(i')\intercal} \mathbf{p}_{i'}^{(i')} - \left|\left| \mathbf{L}_i^{(i')} \mathbf{p}_{i'}^{(i'-1)} \right|\right|_2^2 - \mathbf{b}_i^{(i')\intercal} \mathbf{p}_{i'}^{(i'-1)} \right) \\ &= \sum_{i=1}^{n} \frac{1}{m_i^{(i'-1)}} \left(\left|\left| \mathbf{L}_i^{(i')} \mathbf{p}_{i'}^{(i')} \right|\right|_2^2 - \left|\left| \mathbf{L}_i^{(i')} \mathbf{p}_{i'}^{(i'-1)} \right|\right|_2^2 + \mathbf{b}_i^{(i')\intercal} \left(\mathbf{p}_{i'}^{(i')} - \mathbf{p}_{i'}^{(i'-1)} \right) \right) \\ &\leq \sum_{i=1}^{n} \frac{1}{m_i^{(i'-1)}} \left(\left(\frac{m-1}{m} \lambda \right)^2 + \mathbf{b}_i^{(i')\intercal} \left(\mathbf{p}_{i'}^{(i')} - \mathbf{p}_{i'}^{(i'-1)} \right) \right) \\ &= \mathbf{b}^{(i')\intercal} \left(\mathbf{p}_{i'}^{(i')} - \mathbf{p}_{i'}^{(i'-1)} \right) + \sum_{i=1}^{n} \frac{1}{m_i^{(i'-1)}} \left(\frac{m-1}{m} \lambda \right)^2 \end{aligned} \tag{4.4}$$

for some vector $\mathbf{b}^{(i')} \in \mathbb{R}^m$ (the sum of the vectors $\frac{1}{m_i^{(i'-1)}} \mathbf{b}_i^{(i')}$). So, ultimately, our algorithm aims to minimize this upper bound by ensuring that the first term is not positive. Recall that $\mathbf{p}_{i'}^{(i'-1)}$ is a (given) vector of non-negative real numbers with $\left|\left| \mathbf{p}_{i'}^{(i'-1)} \right|\right|_1 = 1$, while $\mathbf{p}_{i'}^{(i')}$ is a (desired) vector containing $m-1$ zeros and a single one. Consider the element b (at index j) of $\mathbf{b}^{(i')}$ with the smallest value *among those indices in the relevant set* $S_{i'}^{(i')}$ (note that player i''s entire support will be contained in $S_{i'}^{(i')}$). Set $\mathbf{b} = \mathbf{e}_j$ (the standard basis vector in dimension j). This ensures that the first term in

Equation 4.4 is not positive.

So, proceeding iteratively as described, this ensures

$$\sum_{i=1}^{n}\sum_{i'=1}^{n}\sigma_i^2\left(S_i^{(i'-1)},i'\right)-\sigma_i^2\left(S_i^{(i'-1)},i'-1\right)\leq\sum_{i=1}^{n}\sum_{i'=1}^{n}\frac{1}{m_i^{(i'-1)}}\left(\frac{m-1}{m}\lambda\right)^2\leq\left(\frac{m-1}{m}n\lambda\right)^2.$$

We can now bound:

- The maximum initial sum of variances of all players at $2\left(\frac{n\lambda(m-1)}{m}\right)^2$
- The maximum contribution to the sum of variances of all players from adding actions to the relevant set at $4n\lambda^2(\log(m-1)+1)$
- The maximum contribution to the sum of variances of all players from purifying the equilibrium at $\left(\frac{m-1}{m}n\lambda\right)^2$

Adding these together, the maximum sum of variances of actions in the relevant sets of players in $\mathbf{p}^{(n)}$ is

$$\begin{aligned}&\left(\frac{m-1}{m}n\lambda\right)^2+4n\lambda^2(\log(m-1)+1)+2\left(\frac{n\lambda(m-1)}{m}\right)^2\\&=3\left(\frac{m-1}{m}n\lambda\right)^2+4n\lambda^2\left(\log(m-1)+1\right))\\&<8n^2\lambda^2\log 3m\end{aligned}$$

for $m>2$.

Step 3: Pure profile to pure equilibrium

Finally, for any value of δ_1, if any player has regret at least δ_1, the variance of their relevant actions must be at least

$$\frac{\delta_1^2}{2m} \tag{4.5}$$

(there must be at least a pair of relevant actions that are a distance of δ_1 apart to achieve this regret, so the lowest-variance option is realized when all remaining actions yield regret $\delta_1/2$). So allow every player with regret greater than δ_1 to instead change to playing their best response (there will be at most

$$\frac{16n^2\lambda^2m\log 3m}{\delta_1^2}$$

such players). As each of these players can add at most 2λ to the regret of any other player (the payoff of the best response could increase by λ while that of the chosen action could decrease by λ), the maximum regret any player can experience is

$$\delta_1 + 2\lambda \frac{16n^2\lambda^2 m \log 3m}{\delta_1^2}.$$

Optimizing for δ_1, this value is at most

$$6\lambda \sqrt[3]{n^2 m \log 3m}.$$

□

This completes the proof of the general version of Theorem 4.2.

4.2.2 Hardness for PPAD

In Theorem 4.7 below, we only consider binary-action games (clearly the hardness results can be extended to games with more actions).

Theorem 4.7. *There exists some constant $\epsilon > 0$ such that $(2, \epsilon, \epsilon n^{-4/5})$-*PURELPG *is* **PPAD***-hard.*

Remark 28. While we show this Theorem for $\lambda = \epsilon n^{-4/5}$, the same proof holds for $\lambda = \epsilon n^{-\alpha}$ for any $\alpha \in (\frac{1}{2}, 1)$. Note that if $\alpha \leq \frac{1}{2}$ the pure equilibrium guarantee of [2] does not hold.

Proof of Theorem 4.7. By Theorem 2.48, there is a constant $\epsilon > 0$ such that it is **PPAD**-hard to find ϵ-ANEs of polymatrix games (we may assume $\frac{1}{\epsilon} \in \mathbb{N}$). We reduce the problem of computing an ϵ-ANE of polymatrix game G to finding an ϵ-PNE of Lipschitz polymatrix game G' as follows. Consider the induced population game $G' = g_G(L)$ for $L = n^4/\epsilon^5$. This game has $N = (n/\epsilon)^5$ players and is $\epsilon N^{-4/5}$-Lipschitz. Thus (for large enough n) it is guaranteed to have an ϵ-PNE. Furthermore, G' is still a polymatrix game, as the payoff to any player v_ℓ^i is simply the sum of the payoffs from $N - L$ games played with the players $v_\ell^{i'}$ for $i' \neq i$.

Now, because the payoffs of any pure action profile of G' can be derived from payoffs of mixed strategy profiles of G (which can be computed in polynomial time), this entire reduction occurs in polynomial time. Thus there is a constant $\epsilon > 0$ such that, for $\lambda(n) = \epsilon n^{-4/5}$, $(2, \epsilon, \lambda)$-PURELPG is **PPAD**-hard. □

Combining Theorems 4.2 and 4.7 yields Theorem 4.1. In fact, by scaling the payoffs in Theorem 4.1 as described in Remark 10, we obtain:

Corollary 4.8. *Take some constant* $\alpha \in (\frac{2}{3}, 1)$. *For any constant* $m \geq 2$, *non-increasing function* $\epsilon(n) = \frac{1}{\text{poly}(n)}$, *and* $\lambda(n) = \Theta(\epsilon n^{-\alpha})$, (m, ϵ, λ)-PURELPG *is* **PPAD**-*complete.*

4.3 Randomized PPAD

As noted above, (m, ϵ, λ)-PURELPG and the more general version relaxing the polymatrix constraint to any game with concisely-representable Lipschitz payoff functions seems to naturally belong to some randomized version of **PPAD**. However, unlike classes such as **P** or **BPP**, we have no real definition of a **PPAD** algorithm. As such, it's difficult to define randomized **PPAD** analogously to how the definition of **BPP** is often derived from the definition of **P**.

We certainly have other options to explore. In the coming sections we discuss a few such options and how they relate to each other. These definitions are based on the definition of **search-BPP** presented in [47]:

Definition 4.3.1 (**search-BPP**)**.** Take a search problem defined by relation R. If there exists a polynomial-time randomized algorithm A such that

$$\Pr[R(x, A(x)) = 1] \geq \frac{2}{3}$$

and a polynomial-time randomized algorithm B such that

$$R(x, y) = 0 \rightarrow \Pr[B(x, y) = 0] \geq \frac{2}{3}, \qquad \Pr[B(x, A(x)) = 1] \geq \frac{1}{2}$$

then $R \in$ **search-BPP** (where the probability is taken over the randomness of A and B).

4.3.1 Mapping Reduction Definition of RPPAD

This definition is the simplest, and is a natural extension to the standard **PPAD** class.

Definition 4.3.2 (**RPPAD**)**.** Take a total search problem defined by relation R. If there exists a polynomial-time computable randomized function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ and deterministic algorithm

A solving `EndOfTheLine` such that

$$\Pr[R(x, A(f(x))) = 1] \geq \frac{2}{3}$$

for any $x \in \{0,1\}^*$, as well as a polynomial-time randomized algorithm B such that

$$R(x, y) = 0 \to \Pr[B(x, y) = 0] \geq \frac{2}{3}, \qquad \Pr[B(x, A(f(x))) = 1] \geq \frac{1}{2}$$

then $R \in \mathbf{RPPAD}$ (where the probability is taken over the randomness of f and B).

4.3.2 Turing Reduction Definition of BPPPAD

The next definition we present of "randomized **PPAD**" is, at first glance, a superset of the mapping reduction definition above.

Definition 4.3.3 (**BPPPAD**)**.** We present here the definition of $\mathbf{BPPPAD} = \mathbf{search\text{-}BPP}^{\mathbf{PPAD}}$. Take a total search problem defined by relation R. If there exists a polynomial-time randomized algorithm A with oracle access to `EndOfTheLine` such that

$$\Pr[R(x, A(x)) = 1] \geq \frac{2}{3}$$

for every $x \in \{0,1\}^*$ and a polynomial-time randomized algorithm B such that

$$R(x, y) = 0 \to \Pr[B(x, y) = 0] \geq \frac{2}{3}, \qquad \Pr[B(x, A(x)) = 1] \geq \frac{1}{2}$$

then $R \in \mathbf{BPPPAD}$ (where the probability is taken over the randomness of A and B).

Remark 29. Note that both of the above definitions support amplification of the correctness probability, as

1. First, the correctness of decision algorithm B can be amplified.

2. With that completed, the algorithm can be run an appropriate number of times such that, with appropriately high probability, one of the runs outputs a correct answer that is accepted by B.

4.3.3 Equivalence of Definitions

In this section, we prove the following:

Theorem 4.9. RPPAD = BPPPAD.

This theorem is the randomized analogue of [12]. Below we present the proof and an example (note that we need only convert the the oracle to a function f, as the algorithm B can remain untouched).

Proof. Obviously, **RPPAD** $\subseteq$ **BPPPAD**, so what remains is to prove that **BPPPAD** $\subseteq$ **RPPAD**. In order to do so, we must take a relation $R \in$ **BPPPAD** solved by algorithm A and present an **RPPAD** algorithm B that mimics its behavior.

The essence of this proof consists of condensing the (polynomially-many) **PPAD** oracle calls into a single call that itself encodes much of the logic of algorithm A. In order to reduce to the standard (deterministic) version of this proof, we will also fix a random string r and show that

$$R(x, A(x, r)) = R(x, B(x, r))$$

for all $x \in \{0,1\}^*$.

So assume that, on input $x \in \{0,1\}^n$ and $r \in \{0,1\}^{\ell(n)}$ (for some polynomial ℓ), A makes $p(n)$ `EndOfTheLine` queries for some polynomial p (note that, because the `EndOfTheLine` queries still return deterministically, the entire sequence of queries is uniquely determined by a given value for r)[2]. The i^{th} query, denoted $A_i(x, r)$, consists of a successor circuit S_i and a predecessor circuit P_i, each of size at most $t(n)$ acting on inputs (and outputting strings) of size at most $q(n)$ for polynomials t, q. To successfully convert A to B, we need to construct circuits S and P (and an interpretation function g) satisfying the following constraints:

- $P(0) = 0$
- $S(0) \neq 0$
- $y \neq P(S(y)) \rightarrow g(y) = A(x, r)$
- $(y \neq 0 \wedge y \neq S(P(y)) \rightarrow g(y) = A(x, r)$

[2]We will assume there are always exactly $p(n)$ queries, as if there are fewer we can simply pad with trivial queries.

In other words, we should be able to use g to interpret the output of this single instance as a solution to R on instance x. Note that it suffices to describe polynomial-time algorithms for S and P, since $\mathbf{P} \subseteq \mathbf{P/poly}$.

So consider strings y of the form

$$a_1 b_1 \dots a_{p(n)} b_{p(n)}$$

where

$$a_1, \dots, a_{p(n)} \in \{0,1\}^{t(n)}, b_1, \dots, b_{p(n)} \in \{0,1\}^{q(n)}.$$

Each a_i will represent an instance $A_i(x,r)$ encoding functions S_i and P_i. Each b_i will represent a solution to the instance of a_i, or (in intermediate steps) the working space in which a solution is being computed.

We consider a string y to be well-formed if and only if it satisfies both of the following properties:

1. For any nonzero a_i, $A_i(x,r) = a_i$ when simulating $A(x,r)$ by providing b_j as the answer to oracle call $A_j(x,r)$ for all $j < i$ (because A can be simulated in polynomial-time when the oracle answers are provided, this too is a polynomial-time checkable property).

2. For any nonzero a_i and any $j < i$, b_j is a solution to query a_j (because `EndOfTheLine` $\in \mathbf{FNP}$ this is a polynomial-time checkable property).

For any ill-formed y (not satisfying both of the above), we will define $S(y) = P(y) = y$ to effectively remove these strings from consideration.

Now we define S and P on the remaining strings. Note that, while typically S is a "successor" circuit and P is a "predecessor", these roles do occasionally reverse. This is required to ensure that non-invertible strings of both functions are actually ultimately solutions to our entire query, and not just solutions to the first instance.

In order to define the functions formally, we present the following example. For simplicity, assume that $A(x,r)$ performs three oracle calls, all querying the same `EndOfTheLine` instance (we omit the a_i's as they should behave as described above and only include the b_i's):

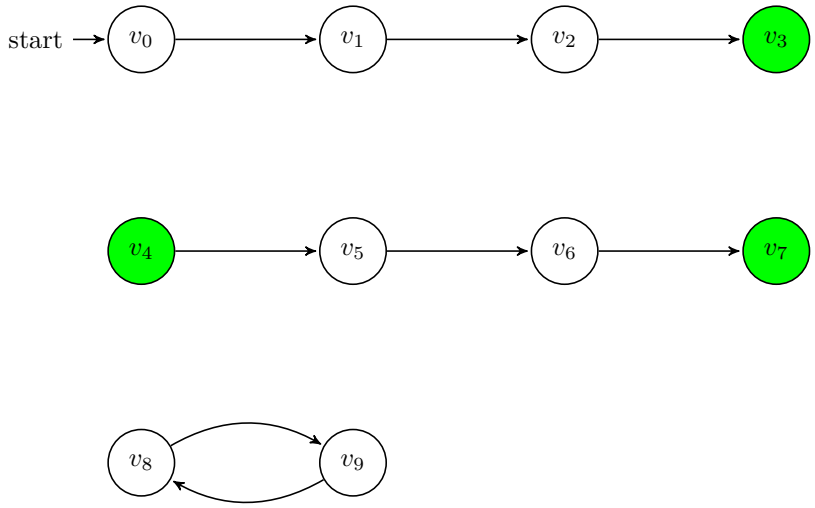


Figure 4.1: `EndOfTheLine` instance to be queried three times.

Below is the resulting graph from combining these three queries. Solutions are highlighted in green, blue, teal, and cyan and cases of concern highlighted in red (these cases are the reason that sometimes F and G must switch roles):

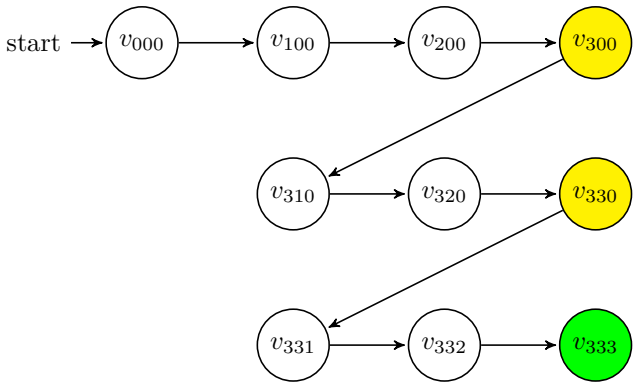


Figure 4.2: The standard `EndOfTheLine` solution.

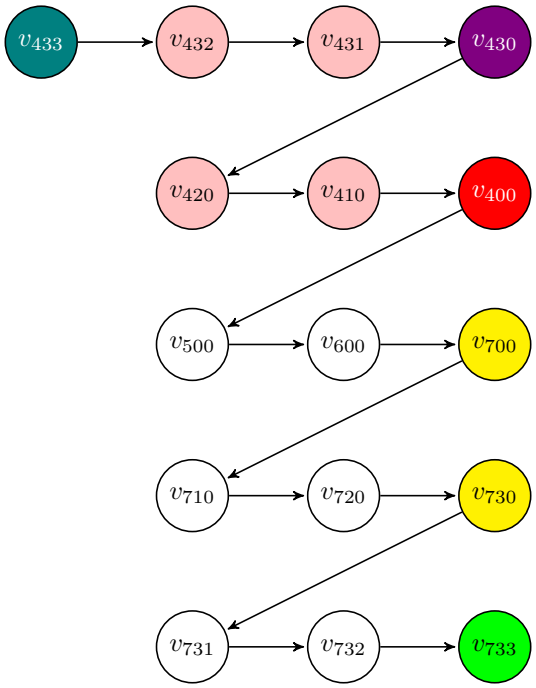


Figure 4.3: The `EndOfTheLine` solution returning an end of the alternate line for the first query.

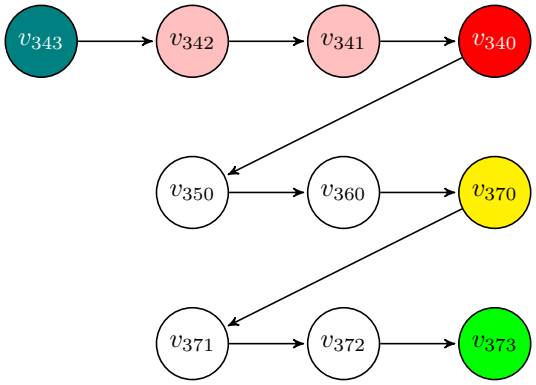


Figure 4.4: The `EndOfTheLine` solution returning an end of the alternate line for the second query.

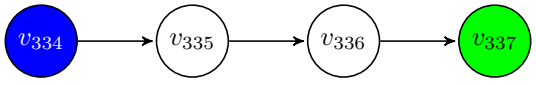


Figure 4.5: The `EndOfTheLine` solution returning an end of the alternate line for the third query.

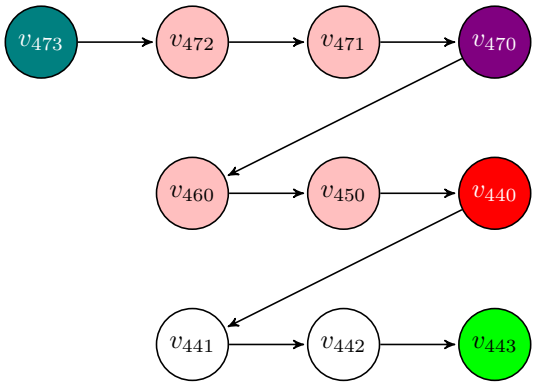


Figure 4.6: The `EndOfTheLine` solution returning the *start* of the alternate line for the first query and an end of the alternate line for the second query.

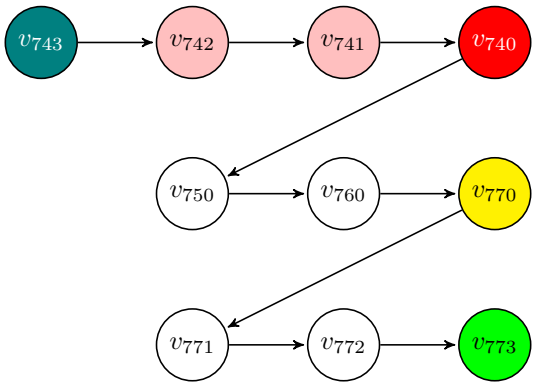


Figure 4.7: The `EndOfTheLine` solution returning the *end* of the alternate line for the first query and an end of the alternate line for the second query.

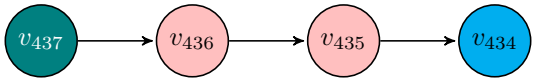


Figure 4.8: The `EndOfTheLine` solution returning the *start* of the alternate line for the first query and an end of the alternate line for the third query.

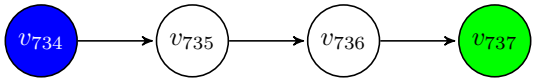


Figure 4.9: The `EndOfTheLine` solution returning the *end* of the alternate line for the first query and an end of the alternate line for the third query.

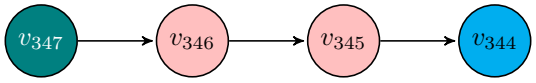


Figure 4.10: The `EndOfTheLine` solution returning the *start* of the alternate line for the second query and an end of the alternate line for the third query.

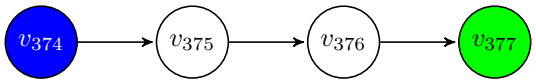


Figure 4.11: The `EndOfTheLine` solution returning the *end* of the alternate line for the second query and an end of the alternate line for the third query.

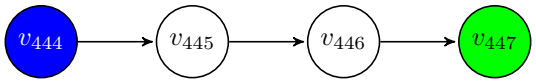


Figure 4.12: The `EndOfTheLine` solution returning the *start* of the alternate line for the first and second queries and an end of the alternate line for the third query.

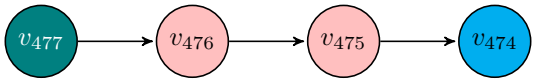


Figure 4.13: The `EndOfTheLine` solution returning the *start* of the alternate line for the first query, the *end* of the alternate line for the second query, and an end of the alternate line for the third query.

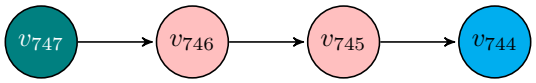


Figure 4.14: The `EndOfTheLine` solution returning the *end* of the alternate line for the first query, the *start* of the alternate line for the second query, and an end of the alternate line for the third query.

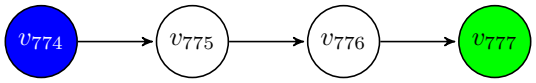


Figure 4.15: The `EndOfTheLine` solution returning the *end* of the alternate line for the first and second queries and an end of the alternate line for the third query.

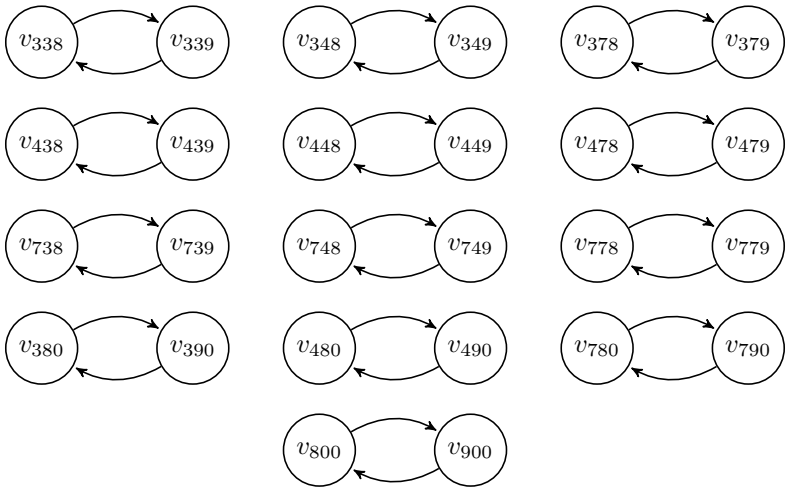


Figure 4.16: The remaining well-formed strings involving nodes on the loop.

Any other string is not well-formed and is assigned a self-loop.

Essentially, the above illustrates the need to ensure that the predecessor of a "4" in any query prior to the final one needs to exist and move smoothly into the next query. Consider any line in query i that does not contain "0": we will paste a forwards copy of query $i+1$ stemming from the end of the line, and a backwards copy of query $i+1$ stemming from the beginning of the line (for example, see rows 2 and 4 of Figure 4.3). Thus the red nodes are no longer solutions: they are instead the middle of paths for which each solution solves *all* queries. Note that there are $3^3 = 27$ possible solutions to this joint query: 3 for each index ("3", "4", and "7"). Furthermore, each well-defined triplet z is either a solution or admits a single consistent value for $S(z)$ and $P(z)$.

So, we define S and P below. Because we assume that the strings are well-formed (otherwise we have already defined their behaviors), this will be relatively simple. Take an input z to S. First, confirm this string is well-formed by performing the following verification (for simplicity we assume that 0 is not a well-formed `EndOfTheLine` instance, though this can be easily handled):

Algorithm 3: Check Well-Formed Strings

```
For i = 1 To p(n):
    Run A(x,r) until query i;
    If A_i(x,r) == a_i:
        If (a_i, b_i) ∈ EndOfTheLine:
            Continue;
        If i == p(n) Or a_{i+1} == 0:
            Return True;
        Return False;
    Else If a_i == 0:
        Return True;
    Return False;
```

In other words, ensure that the two properties for well-formed strings (described above) are satisfied, i.e.:

1. The strings a_i are exactly the queries that $A(x,r)$ makes (up to this point)

2. The strings b_i are valid solutions to the instances a_i (up to this point)

If the above is true, then take i to be the query we are currently working on (note that this can also be easily calculated from a well-formed string). The following algorithm is encoded as circuit S. The colors of specific lines correspond to the nodes at which they apply in the example above:

Algorithm 4: Definition of Circuit S

```
If i==1 Or an even number of {b_1,...,b_{i-1}} are beginnings of line:
    If b_i is end of line in a_i:
        If i == p(n):
            Return z;
        Return A_{i+1}(x,r);
    Return S_i(b_i);
If b_i is beginning of line in a_i:
    If i == p(n):
        Return z;
    Return A_{i+1}(x,r);
Return P_i(b_i);
```

- The green line describes when we have found an end of line solution to the final query.
- The yellow line describes when we have found an end of line solution to an intermediate query.
- The black return statement describes the standard case - we have not yet found a solution and simply use the successor function from instance a_i.
- The cyan line describes when we have found a *beginning* of line solution to the final query (with an odd number of *beginnings* of lines found prior).
- The red line describes when we have found a *beginning* of line solution to an intermediate query (it is at these nodes that the successor and predecessor circuits swap roles).
- The pink line describes when we have not yet found a solution and (because the successor and predecessor circuits are currently swapped) we use the predecessor function from the instance a_i.

The circuit P is defined very similarly:

Algorithm 5: Definition of Circuit P

```
If i==1 Or an even number of {b_1, ..., b_{i-1}} are beginnings of line:
    If b_i is beginning of line in a_i:
        If i == p(n):
            Return z;
        Return A_{i+1}(x, r);
    Return P_i(b_i);
If b_i is end of line in a_i:
    If i == p(n):
        Return z;
    Return A_{i+1}(x, r);
Return S_i(b_i);
```

In addition to the descriptions already associated with circuit S:

- The blue line describes when we have found a beginning of line solution to the final query.
- The teal line describes when we have found an *end* of line solution to the final query (with an odd number of *beginnings* of lines found prior).
- The violet line describes when we have found an *end* of line solution to an intermediate query.

Finally, g is simply defined by the remainder of algorithm $A(x, r)$ after the final query. Algorithm B simply queries the instance of `EndOfTheLine` with circuits S and P and applies g to the output.

Let's not lose sight of the fact that we are proving the equivalence of *randomized* mapping and Turing reductions. The key insight here is that the reduction itself depends on the random string. So, for a given r, the output of $B(x, r)$ is identical to the output of $A(x, r)$. This completes the proof. □

4.3.4 Pure Nash Equilibria

Theorem 4.10. *For a fixed $m \in \mathbb{N}$, consider the problem of finding ϵ-PNEs of n-player, m-action, λ-Lipschitz games presented in the form of payoff circuits (allowed to err if the game is not λ-Lipschitz) where*

$$\epsilon = O(\lambda\sqrt{n \log n}).$$

This problem is contained in **BPPPAD**.

This is proven in two steps:

Lemma 4.11. *For a fixed $m \in \mathbb{N}$, consider the problem of finding* mixed *approximate Nash equilibria of n-player, m-action games presented in the form of payoff circuits. This problem is contained in* **BPPPAD**.

Remark 30. Note that this problem is not covered by Theorem 2.15 as (a) the input variable is the number of players rather than the number of actions and (b) the input is provided as a circuit, not a complete list of payoffs.

Proof Sketch. Recall that, in order to successfully find a mixed Nash equilibrium in **PPAD**, we need efficient access to mixed strategy payoffs. More specifically, we need to successfully compute an $O(\epsilon)$-approximation of these payoffs in time polynomial in n, m, and $\frac{1}{\epsilon}$ (we assume ϵ is inverse polynomial so we need not worry about it). Corollary 2.3 tells us that we can successfully replace such a query with

$$\frac{64m^2}{\epsilon^2} \log\left(\frac{8n}{\eta}\right)$$

profile queries and be correct with probability at least $1 - \eta$. In particular, for any polynomial $p(n, m, \frac{1}{\epsilon})$, we can guarantee an ϵ-approximation for any given payoff with probability $2^{-p(n,m,\frac{1}{\epsilon})}$ making a polynomial number of profile queries. Since we only care about exponentially many possible mixed profiles, we can ensure all queries needed by the **PPAD** algorithm are ϵ-approximations with failure probability inverse polynomial in n, m, and $\frac{1}{\epsilon}$. By a similar but less difficult argument, we note that we can construct the required algorithm B confirming the equilibrium. □

From here, we simply need to show that a random realization of this mixed equilibrium is a pure equilibrium.

Proof Sketch of Theorem 4.10. This is similar to the work of [2], but starts with an approximate equilibrium rather than an exact one. It is very difficult to adapt the proof in this way, but much more feasible when starting from a WSNE. As such, we maintain Step 1 of the algorithm in the proof of Lemma 4.6. As such, we can start from an ϵ_1-WSNE $\mathbf{p}^*$ for

$$\epsilon_1 = \Theta(\lambda\sqrt{n}).$$

The rest of the proof follows that of [2]. By the same logic, there must exist a pure profile $\mathbf{a}^*$ in the support of $\mathbf{p}^*$ such that no deviation by any player to any action affects their payoff by more than $O(\lambda\sqrt{n \log n})$ (in fact, we can find such an $\mathbf{a}^*$ with constant probability from a random realization of $\mathbf{p}^*$ by modifying the constants under the radical). As such, for any player i and deviation d,

$$u_i(d, \mathbf{a}^*_{-i}) \leq u_i(d, \mathbf{p}^*_{-i}) + \Theta(\lambda\sqrt{n \log n}) \leq u_i(a^*_i, \mathbf{p}^*_{-i}) + \Theta(\lambda\sqrt{n \log n} + \lambda\sqrt{n}) \leq u_i(\mathbf{a}^*) + \Theta(\lambda\sqrt{n \log n})$$

where the first inequality comes from the fact that player i's deviation to d affects their payoff by at most $O(\lambda\sqrt{n \log n})$, the second from the fact that $\mathbf{p}^*$ is a $\Theta(\lambda\sqrt{n})$-WSNE, and the third from the fact that player i's deviation to a^*_i once again affects their payoff by at most $O(\lambda\sqrt{n \log n})$. As such, with constant probability, a random realization of our ϵ_1-WSNE is an $O(\lambda\sqrt{n \log n})$-PNE.

Finally, it is deterministic to confirm that this profile is an equilibrium, so the required decision algorithm B is easy to construct.

□

Chapter 5

Upper Bounds

5.1 Overview

In this chapter, we introduce algorithms for finding equilibria in Lipschitz games. In particular, we provide an improvement over Theorem 2.32, reproduced below.

Theorem 5.1 ([41], Theorem 2.32 Restated)**.** *Given an n-player, m-action, $\frac{1}{n}$-Lipschitz game G and any $\alpha, \eta, N > 0$, there is an uncoupled algorithm finding a $\frac{3}{4} + \frac{1}{2N} + \alpha$-approximate Nash equilibrium of G making*

$$O\left(\frac{m^2}{\alpha^3}\log\left(\frac{nN}{\eta}\right)\right)$$

queries.

In other words, we answer the question of whether or not $\frac{3}{4}$ is a barrier in the negative. Through a modification of the algorithm of [41] (a best-response dynamic in which we allow players to respond more rapidly the higher their regret) we achieve a slight improvement over $\frac{3}{4}$, opening the door to the possibility of future improvements.

5.2 Preliminaries

We consider the following concepts and notation unique to this chapter.

5.2.1 The Game Model

In this chapter, we consider specifically n-player, m-action $\frac{1}{n-1}$-Lipschitz games (we will make the distinction that we actually consider $\lambda = \frac{1}{n-1}$ instead of $\frac{1}{n}$ here because it makes certain calculations easier while remaining a slightly stronger result). We will need the following additional notation for this chapter, required as the algorithm we will introduce will take place over multiple discrete timesteps.

Definition 5.2.1. At time t,

- We denote by $\mathbf{p}(t)$ the strategy profile of all players at time t, represented as a collection of n probability distributions.

- Let $p_i(t)$ denote the specific mixed strategy of player i at time t, where we can interpret $p_i(t) \in [0,1]^m$ with $||p_i(t)||_1 = 1$.

- Finally, $\mathbf{p}_{-i}(t)$ represents the mixed strategies of players $[n] \setminus \{i\}$ at time t.

Furthermore, we abbreviate $\mathbf{u}_i(t) = \mathbf{u}_i(\mathbf{p}(t))$ to be the payoff vector to all players for playing $\mathbf{u}_i(\mathbf{p}(t))$.

We will additionally abbreviate $\mathrm{br}_i(t) = \mathrm{br}_i(\mathbf{p}_{-i}(t))$ and $\mathrm{reg}_i(t) = \mathrm{reg}_i(\mathbf{p}(t))$.

5.2.2 Block Equilibrium

[41] are able to obtain Theorem 5.1 via the following algorithm, the "Block Equilibrium" algorithm:

Algorithm 6: Block Equilibrium Computation `BU` (performed by each player i)

Input : Parameters $\eta > 0$, $N \in \mathbb{N}$.
Output: Mixed Strategy $\mathbf{p}_i$ in $0.75 + \frac{1}{N}$-approximate Nash equilibrium with probability $\geq 1 - \eta$.

Initialization
for $\ell \leftarrow 1$ **to** N **do**
  /* Partition the unit interval into discrete blocks of size $\frac{1}{N}$ and allocate each block B_ℓ to an arbitrary action $j \in [m]$. */
  $B_\ell \longleftarrow$ `Random(`m`)`;
end
for $j \leftarrow 1$ **to** m **do**
  /* Assign to $p_i(0)$ the mixed strategy corresponding to the blocks (initializing $\mathbf{p}_i(0)$ to an arbitrary N-uniform mixed strategy profile). */
  $p_{ij}(0) = \sum_{B_\ell = j} \frac{1}{N}$;
end

Update
for $t \leftarrow 1$ **to** N **do**
  $p_i(t) \leftarrow$ `BlockStep(`$p_i(t-1), B_1, \ldots, B_N$`)`; // Note that $B_1, \ldots, B_N$ are also modified.
end

return p_i;

Algorithm 7: The `BlockStep` algorithm

Input : Mixed strategy $p_i(t-1)$, blocks $B_1, \ldots B_N$ (passed by reference).
Output: Mixed strategy $p_i(t)$.

/* Reallocate the least recently assigned block of probability to the current best response (after the first N steps, this block will be the most out of date and has the greatest ability to contribute to the regret). Note that this is where the algorithm relies on randomness - the br function is approximated using the randomized query algorithm from Corollary 2.3. */
$B_{t \pmod N} \leftarrow \mathrm{br}_i\left(\mathbf{p}_{-i}(t-1)\right)$;
for $j \leftarrow 1$ **to** m **do**
  /* Recalculate $p_i(t)$. */
  $p_{ij}(t) = \sum_{B_\ell = j} \frac{1}{N}$;
end

return p_i;

At the conclusion of the outer loop of Algorithm 6, no player can suffer regret greater than $0.75 + \frac{1}{N}$ (with probability $\geq 1 - \eta$ due to the randomized payoff calculations approximating br in Algorithm 7). We begin our algorithm from this point.

5.3 Result Overview

The main result is stated in Theorem 5.2.

Theorem 5.2 (Main Theorem)**.** *For any $\eta > 0$, there exists an algorithm performing*

$$O\left(nm^3 \log\left(\frac{n^2 m}{\eta}\right)\right)$$

profile queries that achieves a 0.737*-approximate Nash equilibrium with probability at least* $1 - \eta$*.*

5.3.1 Outline

We will begin by making a number of simplifying assumptions, which we will then slowly relax in Section 5.5.7 until we achieve the proof of Theorem 5.2 as stated.

The algorithm that achieves this equilibrium divides the total available probability into discrete blocks, which are each assigned and reassigned to different actions to form a mixed strategy. This algorithm is described below in Algorithm 8, and should be performed by each player $i \in [n]$ (relying on Algorithm 7 as a subroutine). The three stages of the algorithm are structured as follows:

1. Perform the standard Block Equilibrium Computation in Section 5.2.2 to upper bound *each player's* regret at just above $\frac{3}{4}$.

2. Extend the standard Block Equilibrium Computation until the *average* regret across all players is just above $\frac{1}{2}$.

3. Perform the standard Block Equilibrium Computation for a very short final step, with each player reassigning probability at a *modified rate* proportional to their regret at the end of Stage Two.

Stage 1 is addressed above in Section 5.2.2. Stage 2 is addressed in Section 5.4, and Stage 3 is addressed in Section 5.5, which completes the proof of Theorem 5.2.

5.3.2 The Algorithm

The following algorithm is performed by each player i:

Algorithm 8: Extended Block Equilibrium Computation `EBU` (performed by each player i)

Input : Parameters $\eta > 0$.
Output: Mixed Strategy $\mathbf{p}_i$ in 0.737-approximate Nash equilibrium with probability $\geq 1 - \eta$.

Initialization
$N \longleftarrow 20571$, $T_1 \longleftarrow N$, $T_2 \longleftarrow 0.057N$;
for $\ell \leftarrow 1$ **to** N **do**
  /* Partition the unit interval into discrete blocks of size $\frac{1}{N}$ and allocate each block B_ℓ to an arbitrary action $j \in [m]$. */
  $B_\ell \longleftarrow$ `Random`(m);
end
for $j \leftarrow 1$ **to** m **do**
  /* Assign to $p_i(0)$ the mixed strategy corresponding to the blocks (initializing $\mathbf{p}_i(0)$ to an arbitrary N-uniform mixed strategy profile). */
  $p_{ij}(0) = \sum_{B_\ell = j} \frac{1}{N}$;
end

Stage One
/* Perform standard Block Equilibrium Computation from [41] employing Algorithm 7 as a subroutine. */
for $t \leftarrow 1$ **to** N **do**
  $p_i(t) \leftarrow$ `BlockStep`$(p_i(t-1), B_1, \ldots, B_N)$; // Note that $B_1, \ldots, B_N$ are also modified.
end

Stage Two
/* Continue reallocating blocks until the average regret over all players is at most $\frac{1}{2} + \frac{1}{2N}$. */
while $\sum_{i=1}^{n} \hat{U}_i(\mathbf{p}(t)) > \frac{n}{2} + \frac{n}{2N}$ // We prove in Theorem 5.3 that this takes at most T_1 steps **do**
  $t \leftarrow t + 1$;
  $p_i(t) \leftarrow$ `BlockStep`$(p_i(t-1), B_1, \ldots, B_N)$;
end
/* Determine the regret after Stage 2. */
$\epsilon_i \longleftarrow \mathrm{reg}_i(\mathbf{p}(t))$;
$N_i' \longleftarrow \frac{N}{2\epsilon_i}$;

Stage Three
/* Rebucket the probability into blocks of size $\frac{1}{N'}$ with B_1' containing the least recently assigned probability and $B_{N'}'$ containing the most recently assigned probability (note that $B_{N'}'$ is associated with less probability than the other blocks if N' is not an integer). Some blocks may be associated with multiple actions. */
$B_1', \ldots, B_{N'}' \longleftarrow$ `Rebucket`$(B_1, \ldots, B_N)$;
/* Reallocate probability at a rate proportional to each player's regret. */
for $_ \leftarrow 1$ **to** T **do**
  $t \leftarrow t + 1$;
  $p_i(t) \leftarrow$ `BlockStep`$(p_i(t-1), B_1', \ldots, B_{N'}')$;
end

return p_i;

Remark 31. It will suit us throughout the next few sections to use some of the constants N, T_1, T_2, η. The definitions of these constants can be found in the initialization phase or the input of Algorithm 8.

5.4 Stage 2

The most critical observation for Stage 2 is that it always terminates in finite time. In this section, we prove the following result:

Theorem 5.3. *Stage 2 of Algorithm 8 terminates in at most T_1 steps.*

In Section 5.4 we prove Theorem 5.3. For simplicity, we will re-index such that Stage 2 begins at time $t = 0$. Note that this means we are guaranteed for every player i that $r_i(t) \leq \frac{3}{4} + \frac{1}{2N}$ for the entirety of $t \in [T_1]$.

5.4.1 The Worst-Case Regret

In addition to the notation introduced in Section 5.2, we introduce the following concepts:

Definition 5.4.1. Assuming that a player i follows the `BU` algorithm, we can define the *best-response utility function* $v_i(t)$ describing the payoff to player i of playing their best response to $\mathbf{p}_{-i}(t)$ against

$$v_i(t) = \max_{j \in [m]} u_i(\mathbf{p}_{-i}(t), j).$$

Proposition 5.1. In any $\frac{1}{n-1}$-Lipschitz game, $v_i(t)$ must also be $\frac{1}{N}$-Lipschitz.

Proof. Say that j_{t-1} is player i's best response to $\mathbf{p}_{-i}(t-1)$ and j_t is player i's best response to $\mathbf{p}_{-i}(t)$. Then

$$v_i(t-1) = u_i(\mathbf{p}_{-i}(t-1), j_{t-1}) \geq u_i(\mathbf{p}_{-i}(t-1), j_t) \tag{5.1}$$

(since j_{t-1} was the best response). Furthermore, by the Lipschitz property, since $n-1$ players are each moving a $\frac{1}{N}$-sized chunk of probability,

$$v_i(t) = u_i(\mathbf{p}_{-i}(t), j_t) \in \left[u_i(\mathbf{p}_{-i}(t-1), j_{t-1}) - \frac{1}{N}, u_i(\mathbf{p}_{-i}(t-1), j_{t-1}) + \frac{1}{N}\right]$$

where the lower-bound indicates that (in the worst case) the payoff for playing j_{t-1} can drop by at most $\frac{1}{N}$ and j_{t-1} can remain the best response, and the upper-bound combines the Lipschitz property

with Equation 5.1. Replacing $u_i(\mathbf{p}_{-i}(t-1), j_{t-1})$ in these expressions with the equivalent $v_i(t-1)$ completes the proof of the Proposition. □

It is, of course, impossible to compute the *regret* from $v_i(t)$ without knowing the game itself - however, we can obtain a very simple upper bound (one that is even independent of the number of actions). This can be done by assuming that every action, as soon as probability is assigned to it, decreases in utility down to 0 as quickly as the Lipschitz requirement allows it.

Definition 5.4.2. Denote by $r_i^{(v)}(t)$ the worst-case regret suffered by a player i following the `BU` algorithm given best-response utility function $v_i(t)$.

Problems with "Worst-Case" Worst-Case Analysis

Section 5.4 will primarily analyze the *average* worst-case regret over the interval $[T_1]$ instead of the *worst-case* worst-case regret.

Analyzing the algorithm above is made difficult by the fact that for any player i, $v_i(t)$ can fluctuate heavily (subject to the Lipschitz constraint). An adversarial choice of $v_i(t)$ might sharply decrease at a given time to achieve a higher regret later on.

Remark 32. If at some time t' player i's regret $\mathrm{reg}_i(t') = \frac{3}{4} + \frac{1}{2N}$, $v_i(t)$ must decrease from 1 at time t' down close to $\frac{1}{2}$ before the regret can return again to $\frac{3}{4} + \frac{1}{2N}$.

In fact, a further complication arises:

Remark 33. Independently of all other players, given any time t' there is an adversarial choice of $v_i(t)$ such that $r_i(t) = \frac{3}{4} + \frac{1}{2N}$. In particular, there is no way to guarantee $r_i(T_1) < \frac{3}{4} + \frac{1}{2N}$.

Instead, we will need to consider the *average* value of $r_i(t)$ throughout the run of the `EBU` algorithm.

5.4.2 The Worst-Case Best Response

In particular, we will present a bound for the average value of $r_i(t)$ that is irrespective of the choice of function v:

Theorem 5.4. *Given any best-response utility function $v_i(t)$, the average value that $r_i^{(v)}(t)$ takes on the interval $t \in [T_1]$ is upper bounded by the average value that $r_i^{(1)}(t)$ takes on the same interval.*

We will show this by examining intermediate steps between an arbitrarily chosen $v_i(t)$ and 1.

Lemma 5.5. *Given a $\frac{1}{N}$-Lipschitz best-response utility function $v_i(t)$ for player i, take*

$$t_\mu := \arg\min_{t\in[T_1]} v_i(t)$$

to be the point at which $v_i(t)$ is minimal, and

$$\mu = v_i\left(t_\mu\right)$$

to be that minimal value.

For any $v_i \not\equiv 1$, the function $r_i^{(v')}(t)$ associated with best response utilities $v_i'(t)$ defined by

$$v_i'(t) = \begin{cases} v_i(t) & t \neq t_\mu \\ \min\left\{\mu + \frac{1}{N}, 1\right\} & t = t_\mu \end{cases}$$

exhibits an average value over the interval $t \in [T_1]$ at least as large *as the average value of $r_i^{(1)}(t)$ over the same interval.*

Proof. To begin, note that by assumption $\mu < 1$, and we are guaranteed that

- If $t_\mu \neq 0$, $\mu \in \left[v_i\left(t_\mu - 1\right) - \frac{1}{N}, v_i\left(t_\mu - 1\right)\right]$.
- If $t_\mu \neq T_1$, $\mu \in \left[v_i\left(t_\mu + 1\right) - \frac{1}{N}, v_i\left(t_\mu + 1\right)\right]$.

Thus $v_i'(t)$ is also $\frac{1}{N}$-Lipschitz.

The result of this perturbation of the best-response function is two-fold. First, the worst-case regret of player i at time t_μ will increase (since player i's strategy profile is unchanged but the regret increases with the best response, which remains unplayed). Then, the payoff to player i at time $t_\mu + 1$ (and for some time after) increases, as player i is now able to actually take advantage of this change in best-response utility.

First taking $t_\mu \leq T_1 - \mu N - 1$, we will need to investigate two cases: when $v_i(t)$ is far from equivalent to the constant function 1, and when it is close.

Case 1: Assume $\mu \leq 1 - \frac{1}{N}$. Player i's strategy profile remains unchanged at time t_μ, but their best response utility has been increased by $\frac{1}{N}$, so

$$r_i^{(v')}\left(t_\mu\right) = r_i^{(v)}\left(t_\mu\right) + \frac{1}{N}. \tag{5.2}$$

Now, at time $t = t_\mu + 1$, block $B_{t_\mu+1 \pmod N}$ is now assigned to the best-response at time t_μ: an action with utility $\frac{1}{N}$ higher than its original utility associated with v. Every other block remains unchanged. So, for t between $t_\mu + 1$ and $t_\mu + 1 + \lfloor \mu N \rfloor$, since $v_i'(t) = v_i(t)$ we have

$$r_i^{(v')}(t) = r_i^{(v)}(t) - \frac{1}{N^2} \tag{5.3}$$

(a single block of probability of size $\frac{1}{N}$ in player i's strategy profile pays $\frac{1}{N}$ better at every step than it did before, and everything else remains unchanged).

Finally, since payoffs are not guaranteed to be multiples of $\frac{1}{N}$, we may have one final time step in which $r_i^{(v')}(t) \neq r_i^{(v)}(t)$:

$$r_i^{(v')}\left(t_\mu + \lfloor \mu N \rfloor + 2\right) = r_i^{(v)}\left(t_\mu + \lfloor \mu N \rfloor + 2\right) - \frac{\mu - \frac{\lfloor \mu N \rfloor}{N}}{N}. \tag{5.4}$$

After time $t_\mu + \lfloor \mu N \rfloor + 2$, $B_{t_\mu+1 \pmod N}$ is assigned to an action with utility 0 in both the original and modified cases, so $r_i^{(v')}(t) = r_i^{(v)}(t)$. What we care about is the sign of

$$\begin{aligned}
\sum_{t=1}^{T_1} \left(r_i^{(v')}(t) - r_i^{(v)}(t) \right) &= \sum_{t=t_\mu}^{t_\mu + \lfloor \mu N \rfloor + 2} \left(r_i^{(v')}(t) - r_i^{(v)}(t) \right) \\
&= \frac{1}{N} - (1 + \lfloor \mu N \rfloor) \frac{1}{N^2} - \frac{\mu - \frac{\lfloor \mu N \rfloor}{N}}{N} \\
&= \frac{N - 1 - \lfloor \mu N \rfloor - \mu N + \lfloor \mu N \rfloor}{N^2} \\
&= \frac{N(1 - \mu) - 1}{N^2} \\
&\geq \frac{N \frac{1}{N} - 1}{N^2} \\
&= 0
\end{aligned}$$

where the $\geq$ comes from the assumption that $\mu \leq 1 - \frac{1}{N}$. So in this case,

$$\sum_{t=1}^{T_1} r_i^{(v')}(t) \geq \sum_{t=1}^{T_1} r_i^{(v)}(t)$$

and thus the average worst-case regret has not decreased.

Case 2: If $\mu > 1 - \frac{1}{N}$, then Equation 5.2 becomes

$$r_i^{(v')}(t_\mu) = r_i^{(v)}(t_\mu) + \frac{1-\mu}{N} \tag{5.5}$$

since the best response has only increased by $1-\mu$ instead of $\frac{1}{N}$ now. Similarly, noting that $\lfloor \mu N \rfloor = N - 1$ Equation 5.3 becomes

$$r_i^{(v')}(t) = r_i^{(v)}(t) - \frac{1-\mu}{N^2} \tag{5.6}$$

for $t = t_\mu + 1$ to $t = t_\mu + N$.

Finally, since we *are* guaranteed that $\frac{1}{N}$ divides 1, Equation 5.4 becomes

$$r_i^{(v')}(t_\mu + N + 1) = r_i^{(v)}(t_\mu + N + 1) \tag{5.7}$$

and can thus be ignored. So

$$\begin{aligned}\sum_{t=1}^{T_1}\left(r_i^{(v')}(t) - r_i^{(v)}(t)\right) &= \sum_{t=t_\mu}^{t_\mu+N}\left(r_i^{(v')}(t) - r_i^{(v)}(t)\right) \\ &= \frac{1-\mu}{N} - N\frac{1-\mu}{N^2} \\ &= 0.\end{aligned}$$

In this case also, the average worst-case regret has not increased.

Since in both cases above the average worst-case regret has not increased, the proof of the Lemma is almost complete. All that remains to note is that the increase in regret (Equations 5.2 and 5.5) always occurs at earlier time intervals than the decrease in regret (Equations 5.3, 5.6, and 5.4), so the claim also holds for $t_\mu > T_1 - \mu N - 1$. □

Theorem 5.4 follows as a corollary:

Proof of Theorem 5.4. Beginning with $v_i(t)$ selected arbitrarily, repeatedly applying Lemma 5.5 at each step we obtain a new valid $v_i(t)$. With each such application, $\sum_{t=0}^{T_1} r_i^{(v)}(t)$ does not decrease. After at most $(T_1+1)N$ applications of Lemma 5.5, we reach $v_i(t) \equiv 1$, and yet the average value of $r_i^{(1)}(t)$ over $t \in [T_1]$ is at least as large as it was originally, completing the proof of the theorem. □

5.4.3 Bounding the Average Worst-Case Regret

Now that we have proven Theorem 5.4, we obtain a bound on the average value of $r_i^{(v)}(t)$ by bounding the average value of $r_i^{(1)}(t)$.

Theorem 5.6. *Algorithm 8 ensures that the average regret of any given player i over the times in $[T_1]$ is at most*

$$\frac{1}{2} + \frac{1}{2N}. \tag{5.8}$$

Furthermore:

Corollary 5.7. *There exists a time $t_1 \in [T_1]$ at which the average regret of all players is at most (5.8).*

The corollary clearly follows from the Theorem - the *average of* the average regrets of all players over the interval $[T_1]$ is at most (5.8), so by switching the order of the sums there must be a time in $[T_1]$ when the average regret of all players is at most this value.

Proof of Theorem 5.6. At time $t = N + 1$, $r_i^{(1)}(t)$ reaches an equilibrium value of $\frac{1}{2} + \frac{1}{2N}$, at which it remains for the remainder of the interval $[T_1]$. This is because at time $t = 1$, probability $\frac{1}{N}$ is assigned to actions with utilities $0, \frac{1}{N}, \frac{2}{N}, \ldots, \frac{N-1}{N}$. The regrets achieved by each of these utilities are, respectively, $1, \frac{N-1}{N}, \frac{N-2}{N}, \ldots, \frac{1}{N}$. Thus

$$\begin{aligned} r_i^{(1)}(N) &= \sum_{\ell=1}^{N} \frac{1}{N} \frac{\ell}{N} \\ &= \frac{N+1}{2N} \\ &= \frac{1}{2} + \frac{1}{2N}. \end{aligned}$$

At every further time-step, the utilities associated with each block of probability decrease by $\frac{1}{N}$ except the lowest (0), which is reassigned to 1 and immediately decreases to $1 - \frac{1}{N}$. This is simply a permutation of the situation at time $t = N$, so we have reached an equilibrium. An application of Theorem 5.4 completes the proof. □

5.5 Stage 3

In this section, we consider Stage 3 to begin at time $t = 0$. This simplifies the notation considerably. Nonetheless, this section is extremely technical, and will include screenshots of Mathematica computations to avoid the need for the longer calculations.

5.5.1 Approximations

For the remainder of this section, we will approximate Stage Three of Algorithm 8 in the continuous limit, in which each block is infinitesimally small, and player i's probability is entirely reallocated over a time interval of length $\frac{N'}{N}$ (recall that in the discrete algorithm Stages 1 and 2 take N iterations and Stage 3 takes N'). Stage 3 is then performed for time 0.057 (as opposed to $0.057N$ time steps). We will additionally assume we have access to δ-distribution queries with $\delta = 0$. Finally, Lemmas 5.17 and 5.18 in Section 5.5.7 describe why these considerations are justified.

5.5.2 The Worst-Case Best-Response

Similar to Section 5.4.2, we are able to consider a best-response function $v_i(t)$ associated with the largest regret a player can suffer.

In this section, we show:

Theorem 5.8. *Given a lower-bound V for the initial condition $v_i(0)$, the worst-case best response for player i as a function of t is*

$$v_i(t) = \begin{cases} 1-(T_2-t) & & 0 \le T_2 \le 1-V \\ \begin{cases} V-t & 0 \le t \le \frac{V+T_2-1}{2} \\ 1-(T_2-t) & \frac{V+T_2-1}{2} < t \le T_2 \end{cases} & & 1-V < T_2 < 1+V \\ \begin{cases} V-t & 0 \le t \le V \\ 0 & V < t < T_2 - 1 \\ 1-(T_2-t) & T_2 - 1 \le t \le T_2 \end{cases} & & 1+V \le T_2 \end{cases}$$

This is the function that provides player i with the highest regret at time T_2.

The Initial Best-Response

This section bounds the value of $v_i(0)$:

Lemma 5.9. *For a player i suffering regret ϵ_0 at the beginning of Stage 3 (time $t = 0$), $v_i(0) \geq 2 - 2\sqrt{1-\epsilon_0}$.*

In other words, in order to suffer regret ϵ_0 at time $t = 0$, a player's best response utility must be large enough that the difference between it and the player's worst-case utility is sufficient. Note that this is well-defined because $\epsilon_0 \leq \frac{3}{4}$ at the end of Stage 2 (so $v_i(0)$ is guaranteed to lie within the range $[0, 1]$).

Proof. Once more, we note that for a given best response, the worst-case regret associated with that best response occurs when the payoff of an action immediately begins dropping with maximal speed as soon as probability is assigned to it. Combining this with a worst-case $v_i(t)$ for $t < 0$ (i.e. Stage 2), the probability reassigned at time t will, at time 0, be providing a utility of

$$\max\{0, v_i(t) + t)\} = \max\{0, v_i(0) + 2t\}.$$

In the continuous case, finding the associated worst-case regret $r_i(0)$ is simply equivalent to finding $v_i(0)$ minus the area of a triangle - it is the integral of a piecewise-linear function (see the right-hand graph in Figure 5.1). This gives a total regret of

$$\epsilon_0 \leq v_i(0) - \left(\frac{v_i(0)}{2}\right)^2.$$

The solution to this quadratic is

$$v_i(0) \geq 2 - 2\sqrt{1-\epsilon_0}$$

completing our proof. □

In the upcoming sections, we will use $v_i^{(\epsilon_0)}(0)$ in place of V (used in the statement of Theorem 5.8 to represent this specific bound.

Deriving Theorem 5.8

Below we present two Lemmas to help us prove that the function provided in Theorem 5.8 is the worst-case best-response utility function.

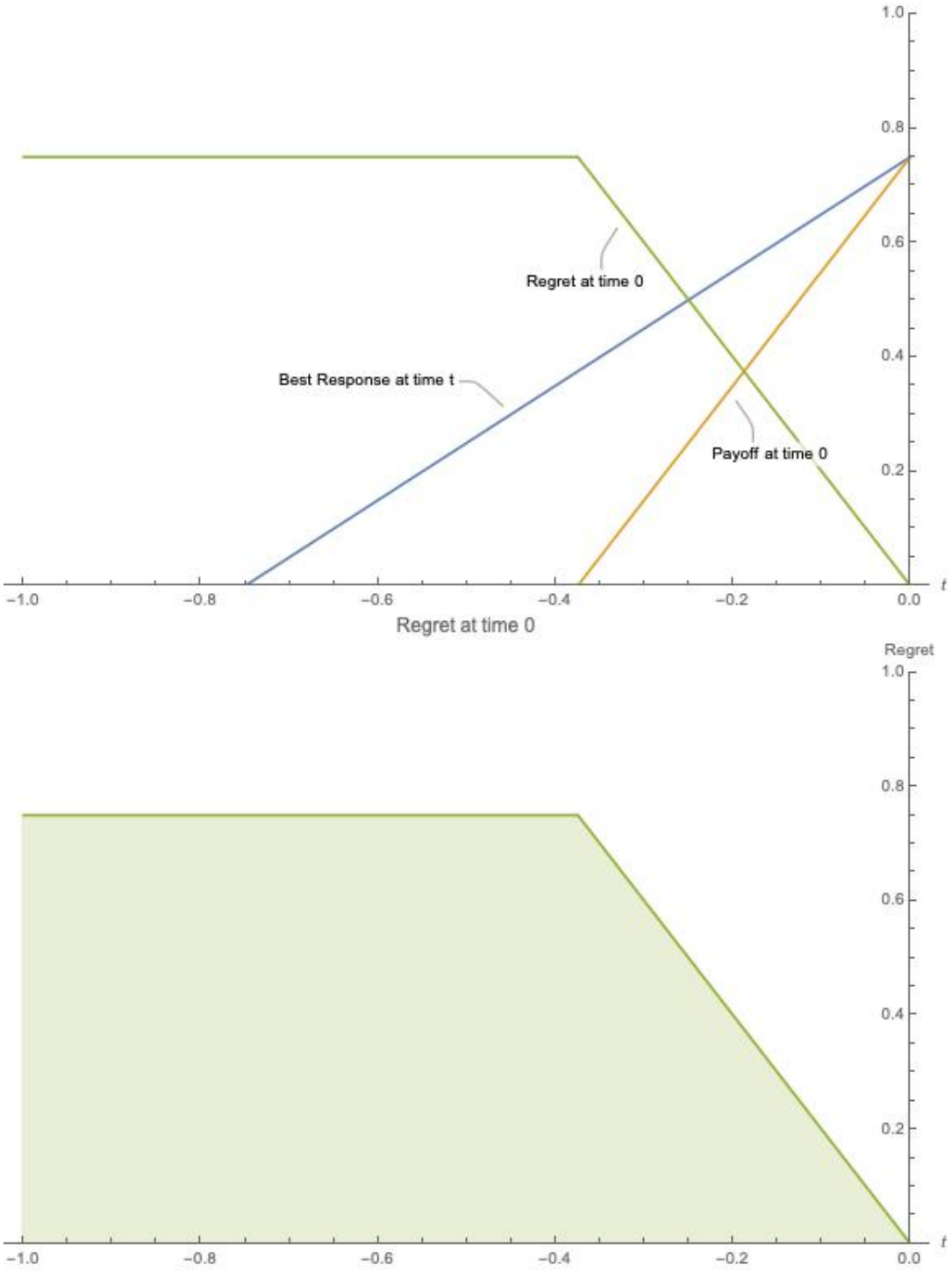


Figure 5.1: Best Response Dynamics. There will be a number of figures of this form to follow. The blue line describes the payoff of the best-response as time goes forward. The orange line shows, for a time t, what the payoff of the action played at time t has dropped to by the time we are considering (here time $t = 0$). Finally, the green line at time t shows the regret of the action played at time t by the time we are considering. The total regret is thus the area under the green curve.

First, we note that adding an additive constant to the payoff of every action played with positive probability does not change the regret.

Lemma 5.10. *Let $\mu = \max_{t\in[0,T_2]} v_i(t)$ for some valid best response function $v_i(t)$ with initial regret ϵ_0. If $\mu < 1$, then $v_i'(t) = v_i(t) + 1 - \mu$*

- *is a valid best response function.*
- *does not admit a lower worst-case regret than $v_i(t)$.*

Proof. Consider some mixed profile admitting regret ϵ_0 at time $t = 0$. If we consider raising the payoffs of all actions in the support by a value of $1 - \mu$, we maintain regret ϵ_0 at time $t = 0$. In fact, the only difference is that, given enough time, each action's payoff has the potential of falling by an additional $1 - \mu$. So we can easily achieve the same regret as before at time T_2, and most likely a worse one. □

The above Lemma essentially implies that the best-response utility should be 1 at some point on the interval $[T_2]$ in order to achieve the worst-case regret at time T_2. Furthermore:

Lemma 5.11. *Let s be the time on the interval $(0, T_2)$ at which $v_i(t)$ is*

- *maximal, but*
- *neither $\frac{dv_i}{dt}_-(t) = -1$ nor $\frac{dv_i}{dt}_+(t) = 1$ (i.e. one can decrease the value of v_i at t and maintain a valid $\frac{1}{N}$-Lipschitz best response function)*

(note that here we do not consider the endpoints). Then, maximally decreasing v_i at s while maintaining validity of v_i can only increase the worst-case regret at time T_2.

Proof. All else held the same, the only possible change to the regret at time T_2 is the contribution by the probability assigned at time s. In the worst-case, this probability is either assigned to an action with the same payoff at time T_2 (i.e. 0) or a slightly worse payoff (if it hasn't had time to drop fully to 0). Thus the regret at T_2 is either the same or slightly worse. □

While Lemmas 5.10 and 5.11 appear to be at odds with one another, putting them together actually allows us to prove Theorem 5.8. In particular, because Lemma 5.11 leaves the endpoints untouched, and Lemma 5.10 can only increase them, applying the two Lemmas as many times as possible will result in a piecewise-linear best-response utility function that will take on a maximum value of 1 and satisfy $\left|\frac{dv_i}{dt}_-(t)\right| = \left|\frac{dv_i}{dt}_+(t)\right| = 1$ at all points $t \in [T_2]$ for which these values are defined.

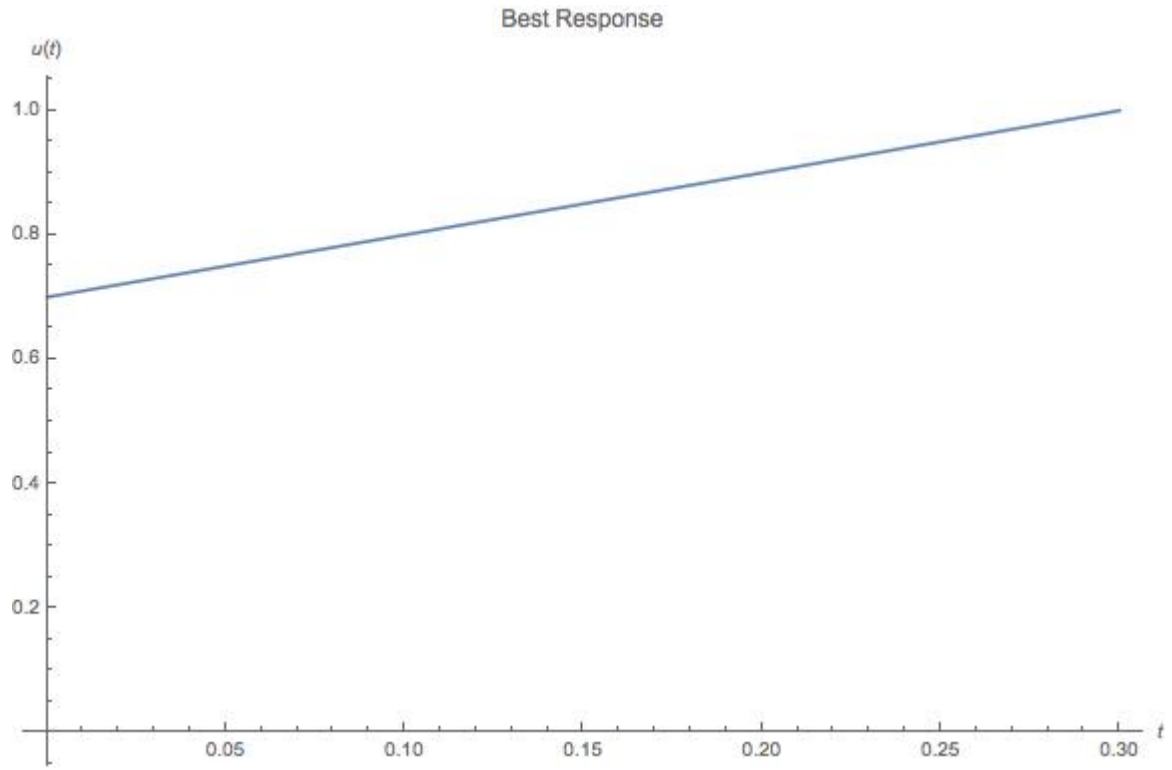


Figure 5.2: The Worst-Case Best Response for $0 \leq T' \leq 1 - v_i^{(\epsilon_0)}(0)$, where $v_i(0) > v_i^{(\epsilon_0)}(0)$ by the application of Lemma 5.10 (recall that $v_i^{(\epsilon_0)}(0)$ is merely a lower-bound for $v_i(0)$ rather than a true initial condition).

Remark 34. When considering this case, note that $v_i^{(\epsilon_0)}(0)$ does not actually appear in the formula. $T_2 < 1 - v_i^{(\epsilon_0)}(0)$ is simply a box to check - if we can check it, then we can proceed.

Proof of Theorem 5.8. We consider each of the cases in turn.

Case 1: $0 \leq T_2 < 1 - v_i^{(\epsilon_0)}(0)$

Because $T_2 \leq 1 - v_i^{(\epsilon_0)}(0)$, even if we set $v_i(0) = v_i^{(\epsilon_0)}(0)$ and define $v_i(t)$ to be the line with slope 1, then $v_i(T_2) < 1$. As such, we can apply Lemma 5.10 to increase every value in $v_i(t)$ to obtain

$$v_i(t) = 1 - (T_2 - t)$$

(see Figure 5.2).

Case 2: $1 - v_i^{(\epsilon_0)}(0) \leq T_2 < 1 + v_i^{(\epsilon_0)}(0)$

When this is the case, by the first inequality, the best response has time to drop first to lengthen its final rise along the lines of Lemma 5.11 (see Figure 5.3). This corresponds to the piecewise-linear

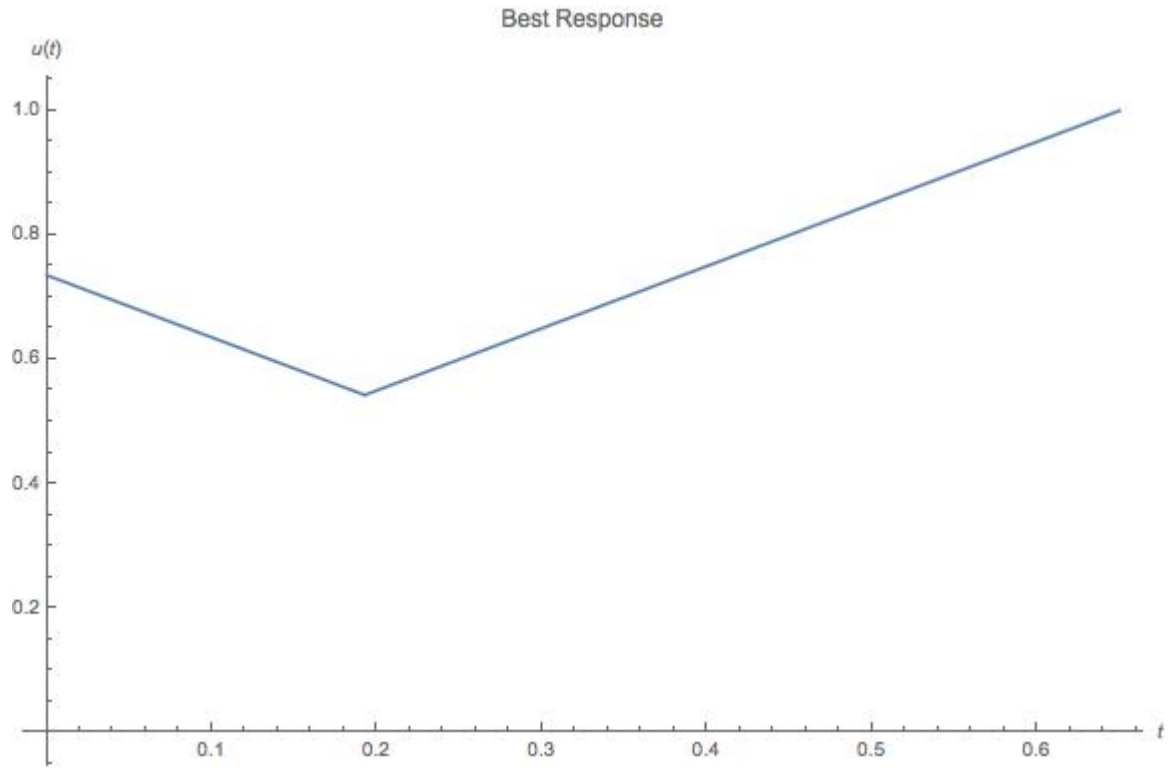


Figure 5.3: The Worst-Case Best Response for $1 - v_i^{(\epsilon_0)}(0) < T_2 < 1 + v_i^{(\epsilon_0)}(0)$

function

$$v_i(t) = \begin{cases} v_i^{(\epsilon_0)}(0) - t & 0 \leq t \leq \frac{v_i^{(\epsilon_0)}(0)+T_2-1}{2} \\ 1 - (T_2 - t) & \frac{v_i^{(\epsilon_0)}(0)+T_2-1}{2} < t \leq T_2 \end{cases}$$

Case 3: $1 + v_i^{(\epsilon_0)}(0) \leq T_2$

Although our choice of T_2 is small enough such that this case never actually comes into play (note that this would require $T_2 \geq 1$), it is important to specifically mention this case to point out that it does not occur above. In this case, we end up applying Lemma 5.11 enough times to result in a best response that decreases to 0 and remains there for a time before increasing all the way to 1 (see Figure 5.4). This always yields a regret of $\frac{3}{4}$, and corresponds to

$$v_i(t) = \begin{cases} v_i^{(\epsilon_0)}(0) - t & 0 \leq t \leq v_i^{(\epsilon_0)}(0) \\ 0 & v_i^{(\epsilon_0)}(0) < t < T_2 - 1 \\ 1 - (T_2 - t) & T_2 - 1 \leq t \leq T_2 \end{cases}$$

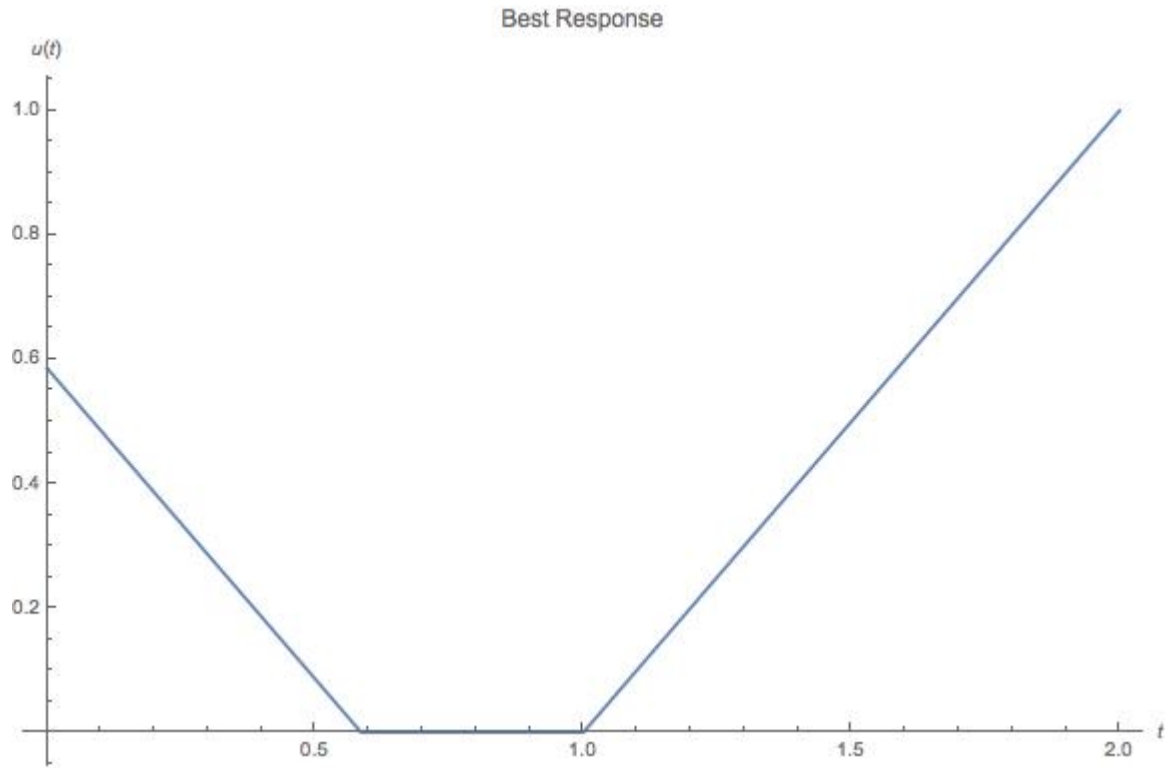


Figure 5.4: The Worst-Case Best Response for $1 + v_i^{(\epsilon_0)}(0) \leq T_2$

□

5.5.3 Bounding the Worst-Case Regret

Theorem 5.12. *For a player with regret ϵ_0 at time $t = 0$ and associated worst-case regret $v_i(t)$ as defined in Theorem 5.8, their regret at time $t = T_2$ can be upper bounded by*

$$r_i^{(v)}(T_2) \leq \epsilon_0 + 2T_2.$$

Furthermore, when $T_2 \leq 0.5$, we can additionally state

$$r_i^{(v)}(T_2) \leq \begin{cases} \frac{3}{4} + T_2 - T_2^2 - 2T_2\epsilon_0 + 2T_2^2\epsilon_0 & 0 \leq T_2 < 2\sqrt{1-\epsilon_0} - 1, 2 - 2\sqrt{1-\epsilon_0} \\ -\frac{9}{4} + \frac{T_2}{2} + \frac{T_2^2}{2} + 3\sqrt{1-\epsilon_0} + 2\epsilon_0 - 2T_2\epsilon_0 & 1 - 2\sqrt{1-\epsilon_0} \leq T_2 < 2 - 2\sqrt{1-\epsilon_0} \end{cases}$$

Note that, at a bare minimum, we can always ensure $r_i^{(v)}(T_2) \leq \epsilon_0 + 2T_2$, so the first case above is trivial. However, when $T_2 < v_i^{(\epsilon_0)}(0)$, this is not enough to achieve an improvement over the initial

block update algorithm. Fortunately, in this situation, we can improve over the trivial case. As a result of this calculation, we obtain:

Corollary 5.13 (Theorem 5.12)**.** *Maximizing $r_i^{(v)}(T_2)$ analytically, $T_2 = 0.057$ yields the minimal worst-case regret of any multiple of* 0.001*, maximal at $r_i(0.623) = 0.7368$.*

Remark 35. The point $(\epsilon_0, T_2) = (0.623, 0.057)$ lies on the boundary between the trivial and the non-trivial bounds. Note that both ϵ_0 is already quite large (since its max values is 0.75) and T_2 is extremely small. This is because we are performing a worse than worst-case analysis (primarily due to our simplifications along the lines of Lemma 5.10), and for most of the range of possible values of ϵ_0 the value $\epsilon_0 + 2T_2$ is already small enough. However, this is sufficient to show that $\frac{3}{4}$ is not a barrier (the main aim of this result).

The proof of Theorem 5.12 gets quite technical. In order to prove the Theorem, we will break up the worst-case regret for a given player i at time T_2 into three pieces whenever $T_2 < v_i^{(\epsilon_0)}(0)$. These three values will depend only on $v_i(t)$ which, in turn, depends only on ϵ_0, and will take our choice of T_2 as a parameter, so we will label them

$$r_i^{(v)}(T_2) = \rho_1(\epsilon_0) + \rho_2(\epsilon_0) + \rho_3(\epsilon_0)$$

where ρ_1 is the regret contribution from the probability reassigned in Stage 3, ρ_2 is the regret contribution from the probability reassigned before Stage 3 with positive payoff, and ρ_3 is the regret contribution from the probability reassigned before Stage 3 paying 0 at time T_2.

5.5.4 Calculating $\rho_1(\epsilon_0)$

We have defined ρ_1 as the contribution to the regret from probability reassigned during Stage 3 of the algorithm. This definition produces the following expression for ρ_1:

Lemma 5.14.

$$\rho_1(\epsilon_0) = \begin{cases} 2T_2^2\epsilon_0 & 0 \le T_2 \le \min\{1 - v_i^{(\epsilon_0)}(0), v_i^{(\epsilon_0)}(0)\} \\ 2\epsilon_0\left(-\frac{5}{4} - \frac{T_2}{2} + \frac{3T_2^2}{4} + \sqrt{1-\epsilon_0} + T_2\sqrt{1-\epsilon_0} + \epsilon_0\right) & 1 - v_i^{(\epsilon_0)}(0) < T_2 \le v_i^{(\epsilon_0)}(0) \end{cases}$$

and undefined elsewhere.

Proof.

Case 1: Consider $0 \le T_2 \le \min\{1 - v_i^{(\epsilon_0)}(0), v_i^{(\epsilon_0)}(0)\}$. Figure 5.5 shows the regret contribution at T_2 of the block reallocated at time t. If player i were reallocating probability at a rate of 1, $\rho_1(\epsilon_0)$ would simply be the area of this triangle with base of length T_2 and height $2T_2$ (because $T_2 \le v_i^{(\epsilon_0)}(0)$, we need not worry that we are treating any payoffs as negative). Instead, since the player is reallocating probability at a rate of $2\epsilon_0$, we will need to include a factor of $2\epsilon_0$ in our final calculation to obtain

$$\rho_1(\epsilon_0) = 2T_2^2\epsilon_0.$$

Case 2: Consider $1 - v_i^{(\epsilon_0)}(0) < T_2 \le v_i^{(\epsilon_0)}(0)$. Figure 5.6 shows the regret contribution at T_2 of the block reallocated at time t. In this case, the regret takes the form of a rectangle as $v_i^{(\epsilon_0)}(t)$ decreases and a triangle as $v_i^{(\epsilon_0)}(t)$ increases. Both have height $1 - v_i^{(\epsilon_0)}(0) + T_2$ (the payoff for each of these blocks decreases by T_2 from $v_i^{(\epsilon_0)}(0)$), and the rectangle has width $\frac{v_i^{(\epsilon_0)}(0)+T_2-1}{2}$ while the triangle's base is $\frac{T_2-v_i^{(\epsilon_0)}(0)+1}{2}$. Being sure to include the $2\epsilon_0$ in front, substituting in the necessary values yields

$$\rho_1(\epsilon_0) = 2\epsilon_0\left(-\frac{5}{4} - \frac{T_2}{2} + \frac{3T_2^2}{4} + \sqrt{1-\epsilon_0} + T_2\sqrt{1-\epsilon_0} + \epsilon_0\right).$$

This completes the proof of Lemma 5.14. □

5.5.5 Calculating $\rho_2(\epsilon_0)$

We have defined ρ_2 as the contribution to the regret from probability reassigned before Stage 3 of the algorithm that has not yet had the opportunity to reach payoff 0.

Lemma 5.15. *Assuming $T_2 \le 0.5$,*

$$\rho_2(\epsilon_0) = \begin{cases} \frac{1}{4} - T_2^2 & 0 \le T_2 \le \min\{1 - v_i^{(\epsilon_0)}(0), v_i^{(\epsilon_0)}(0)\} \\ -1 + \frac{T_2}{2} - \frac{T_2^2}{4} + \sqrt{1-\epsilon_0} - T_2\sqrt{1-\epsilon_0} + \epsilon_0 & 1 - v_i^{(\epsilon_0)}(0) < T_2 \le v_i^{(\epsilon_0)}(0) \end{cases}$$

and undefined elsewhere.

Proof.

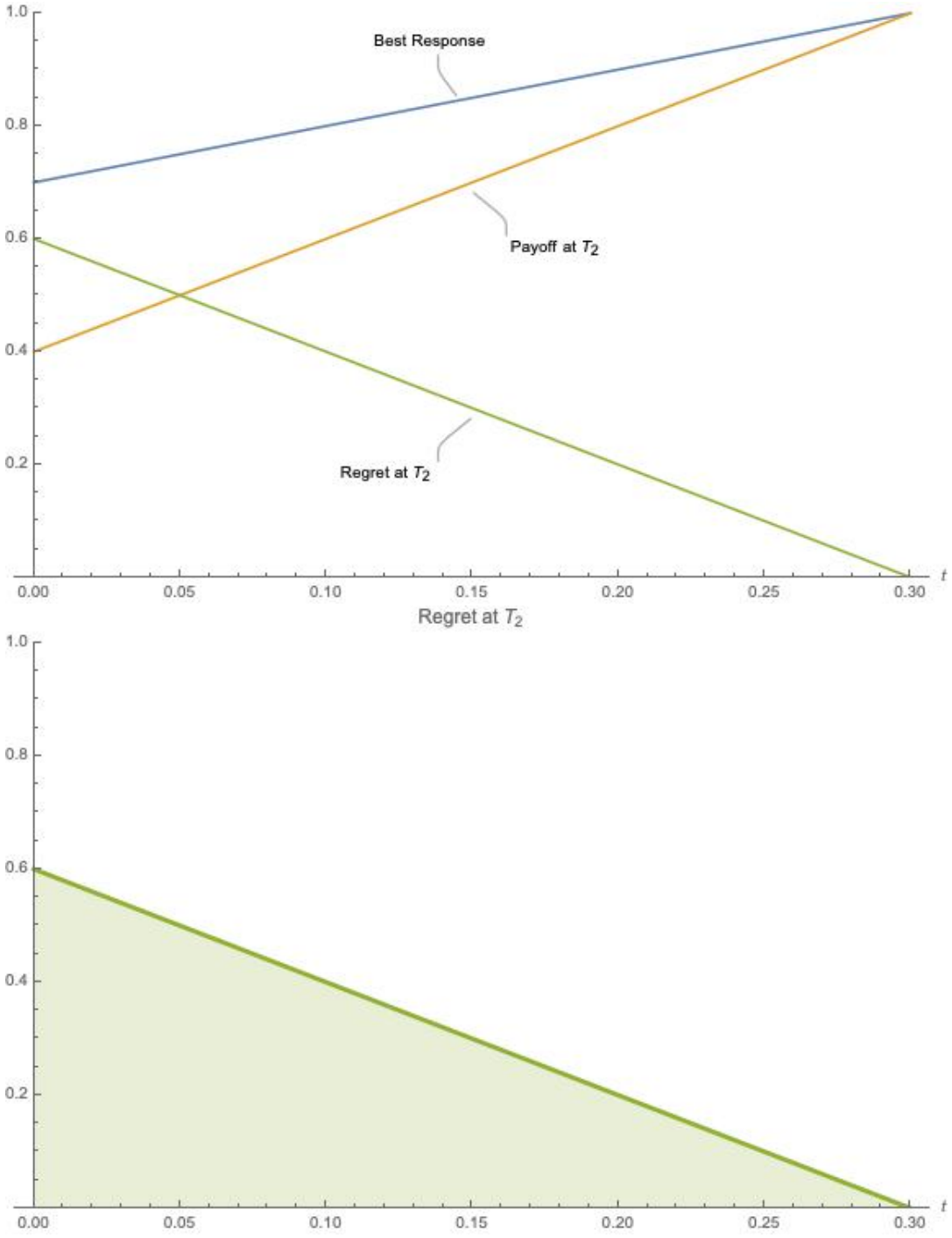


Figure 5.5: $\rho_1(\epsilon_0)$ Dynamics for $0 \leq T_2 \leq \min\{1 - v_i^{(\epsilon_0)}(0), v_i^{(\epsilon_0)}(0)\}$

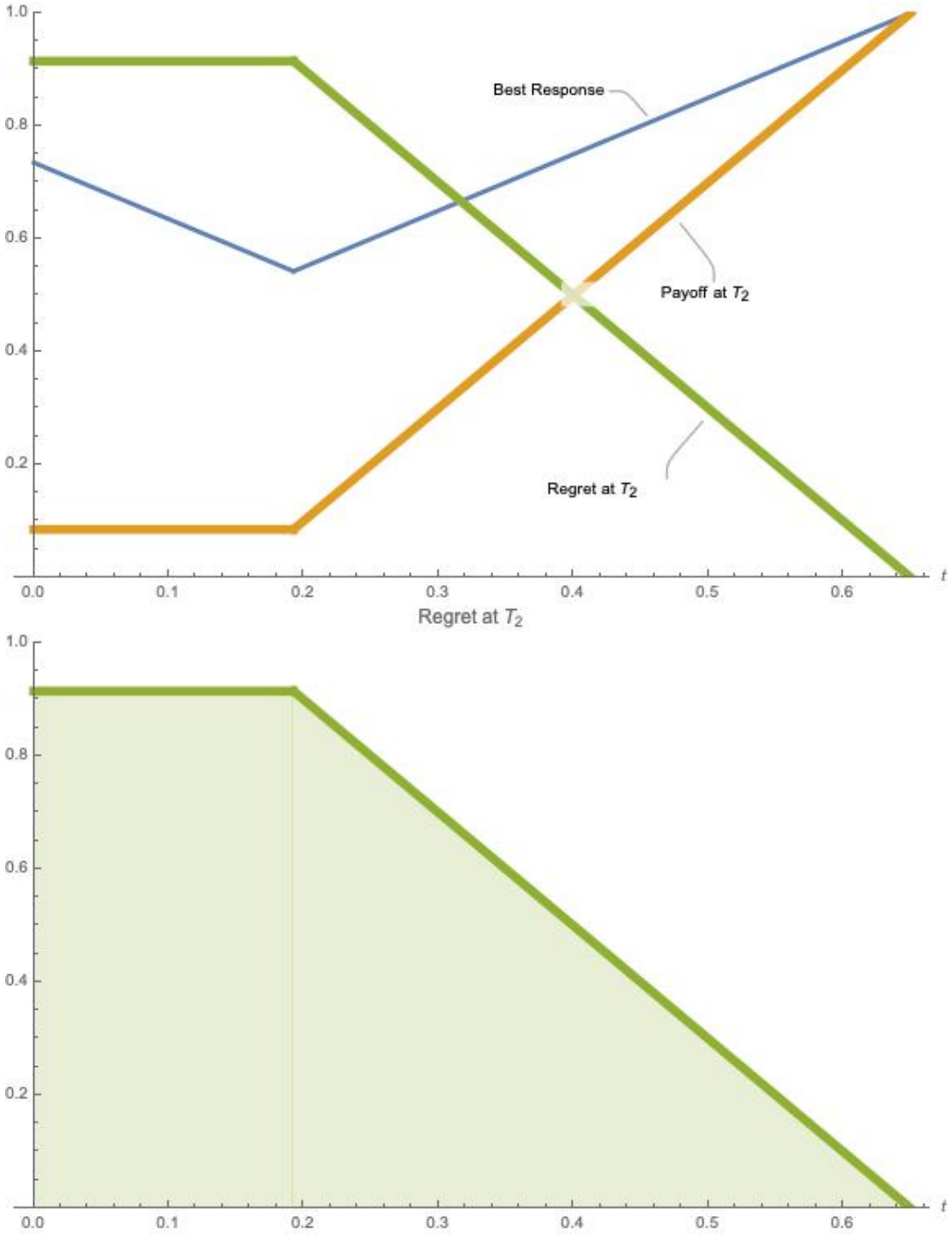


Figure 5.6: $\rho_1(\epsilon_0)$ Dynamics for $1 - v_i^{(\epsilon_0)}(0) < T_2 \leq v_i^{(\epsilon_0)}(0)$

Case 1: Consider $0 \leq T_2 \leq \min\{1 - v_i^{(\epsilon_0)}(0), v_i^{(\epsilon_0)}(0)\}$. Figure 5.7 shows the regret contribution at T_2 of the blocks that were reallocated shortly before $t = 0$. This shape is derived from the fact that some of the probability still could not have reached 0 over this time interval given the initial regret. This is a trapezoid with $h_1 = v_i^{(\epsilon_0)}(T_2) = 1$ (i.e. the regret of an action with payoff 0) and $h_2 = 2T_2$ (i.e. the regret of a block that was assigned at $t = 0$ and dropped for time T_2). Since the slope of the green line is -2 (payoffs drop at rate 1 and best response rises at rate 1), we can calculate the base of this trapezoid to be $\frac{1-2T_2}{2}$. By assumption, this is nonnegative, and thus we need not make any extra assumptions to state

$$\rho_2(\epsilon_0) = \frac{1 + 2T_2}{2} \frac{1 - 2T_2}{2} = \frac{1}{4} - T_2^2.$$

Case 2: Consider $1 - v_i^{(\epsilon_0)}(0) < T_2 \leq v_i^{(\epsilon_0)}(0)$. Figure 5.8 shows the regret contribution at T_2 of the blocks that were reallocated shortly before $t = 0$ (note that the lines are thicker for $t \geq 0$ to account for the higher density of blocks assigned to those values). This trapezoid is somewhat simpler than the above case: $h_1 = 1$, and $h_2 = 1 - (v_i^{(\epsilon_0)}(0) - T_2)$. This gives the same base length as before, yielding

$$\rho_2(\epsilon_0) = \left(1 - \frac{v_i^{(\epsilon_0)}(0) - T_2}{2}\right) \frac{v_i^{(\epsilon_0)}(0) - T_2}{2}.$$

This completes the proof of Lemma 5.15. □

5.5.6 Computing $\rho_3(\epsilon_0)$

Finally, we have defined ρ_3 as the contribution to the regret from the probability assigned to actions paying 0 to player i at time T_2.

Lemma 5.16. *Assuming $T' \leq 0.5$,*

$$r_3(\epsilon) = \begin{cases} 1 - 2T_2\epsilon_0 - \frac{1-2T_2}{2} & 0 \leq T' \leq \min\{1 - v_i^{(\epsilon_0)}(0), v_i^{(\epsilon_0)}(0)\} \\ 1 - 2T_2\epsilon_0 - \frac{v_i^{(\epsilon_0)}(0)-T_2}{2} & 1 - v_i^{(\epsilon_0)}(0) < T' \leq v_i^{(\epsilon_0)}(0) \end{cases}$$

The algebraic and geometric derivations for $r_3(\epsilon)$ are more or less identical.

Proof.

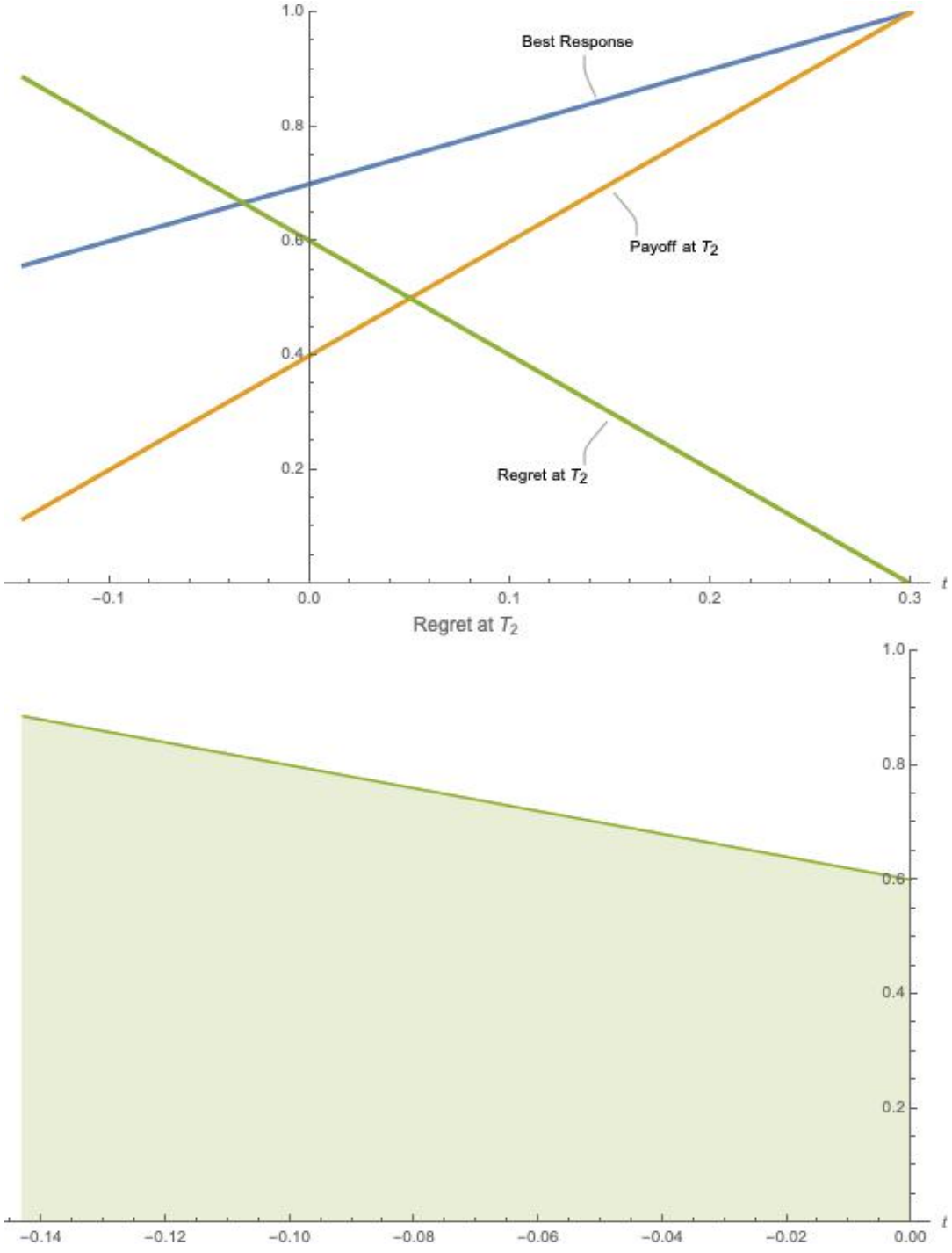


Figure 5.7: $\rho_2(\epsilon_0)$ Dynamics for $0 \leq T_2 \leq \min\{1 - v_i^{(\epsilon_0)}(0), v_i^{(\epsilon_0)}(0)\}$

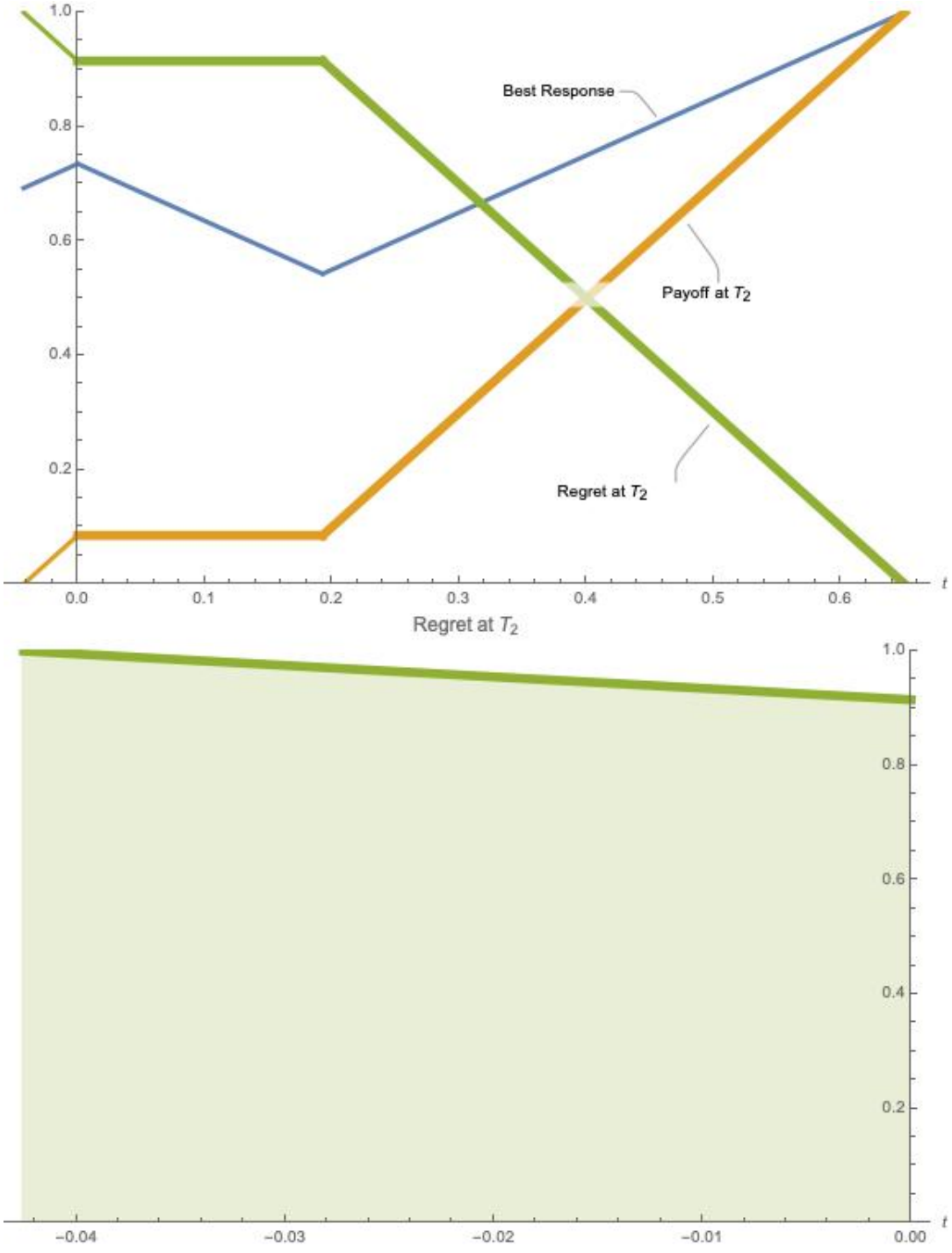


Figure 5.8: $\rho_2(\epsilon_0)$ Dynamics for $1 - v_i^{(\epsilon_0)}(0) < T_2 \leq v_i^{(\epsilon_0)}(0)$

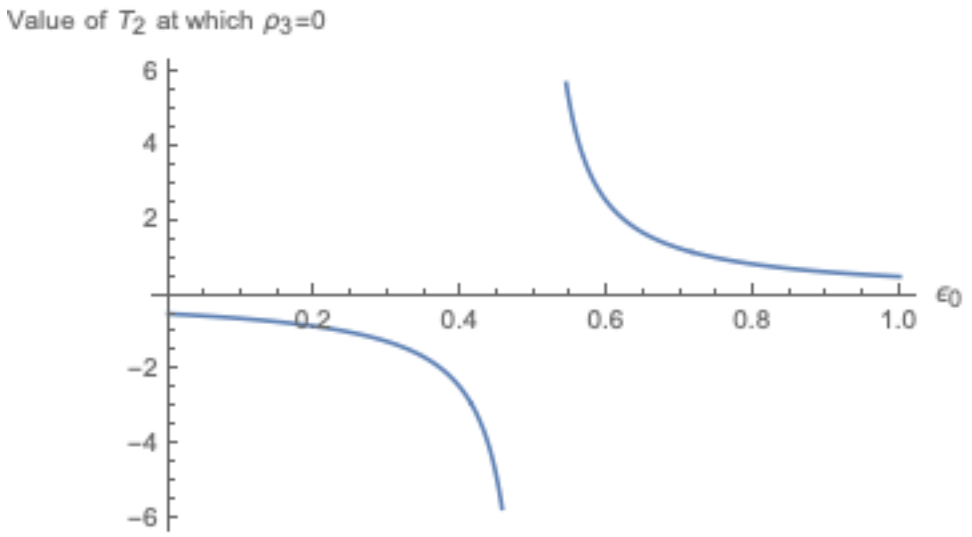


Figure 5.9: In order for ρ_3 to be nonnegative in Case 1 as required, T_2 must lie below this curve (for $\epsilon_0 > \frac{1}{2}$). Since $T_2 \leq 0.5$ by assumption, this is guaranteed to be the case.

Case 1: This first case is easy. Consider $0 \leq T_2 \leq \min\{1 - v_i^{(\epsilon_0)}(0), v_i^{(\epsilon_0)}(0)\}$. The amount of probability paying 0 to player i is the amount unconsidered in ρ_1 and ρ_2. In this first case, that value is

$$1 - 2T_2\epsilon_0 - \frac{1 - 2T_2}{2}.$$

Because all of this probability is experiencing regret 1 (Figure 5.10),

$$\rho_3(\epsilon_0) = 1 - 2T_2\epsilon_0 - \frac{1 - 2T_2}{2}.$$

We need to ensure this is nonnegative, which holds whenever

$$\begin{cases} \epsilon_0 < \frac{1}{2}, T_2 \geq \frac{1}{-2+4\epsilon_0} \\ \epsilon_0 = \frac{1}{2} \\ \epsilon_0 > \frac{1}{2}, T_2 \leq \frac{1}{-2+4\epsilon_0} \end{cases}$$

(depicted in Figure 5.9). Under the assumption that $T_2 \leq 0.5$, this always holds.

Case 2: In this case, we actually have to consider the value of $v_i^{(\epsilon_0)}(0)$. The amount of probability

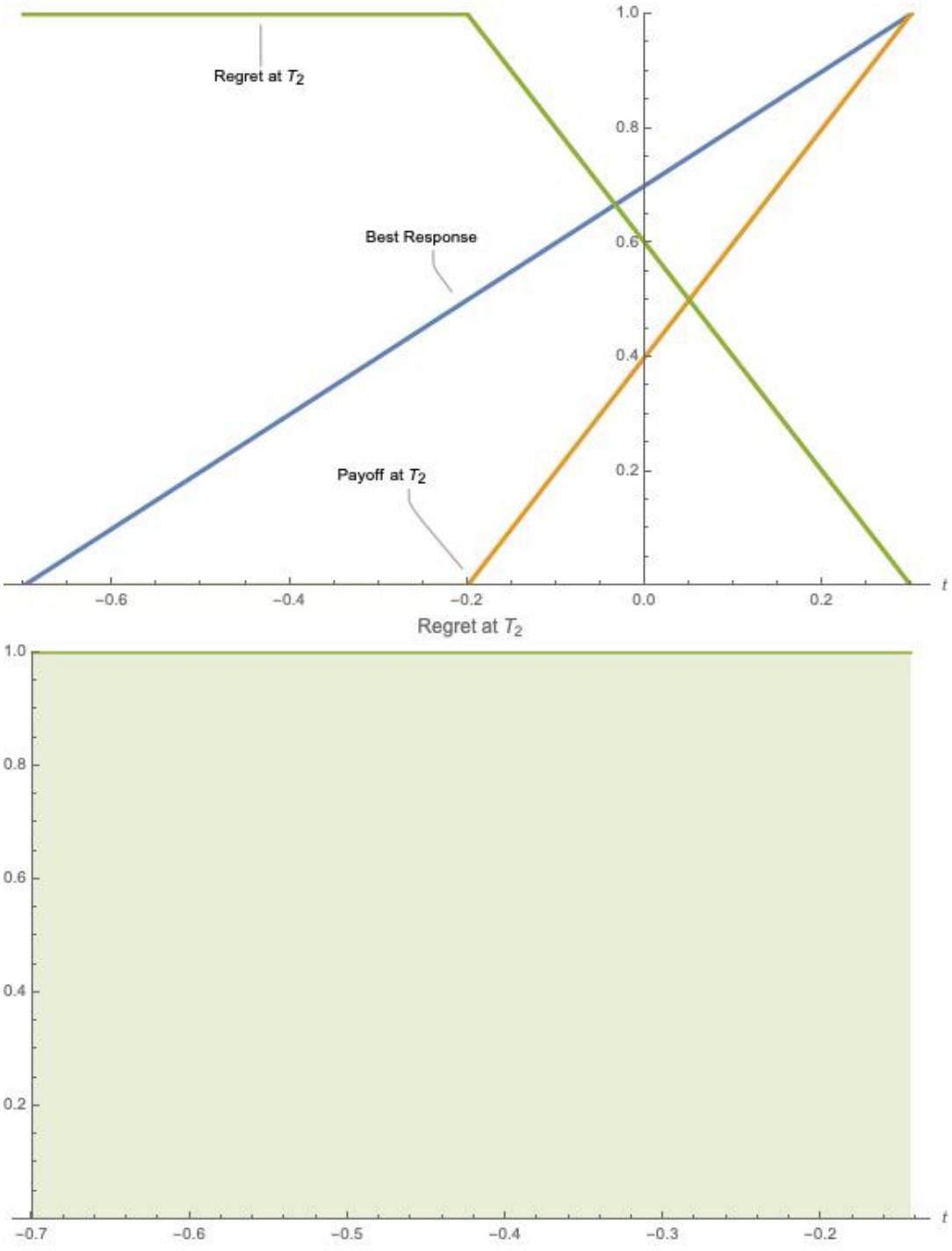


Figure 5.10: $\rho_3(\epsilon_0)$ Dynamics for $0 \leq T_2 \leq \min\{1 - v_i^{(\epsilon_0)}(0), v_i^{(\epsilon_0)}(0)\}$

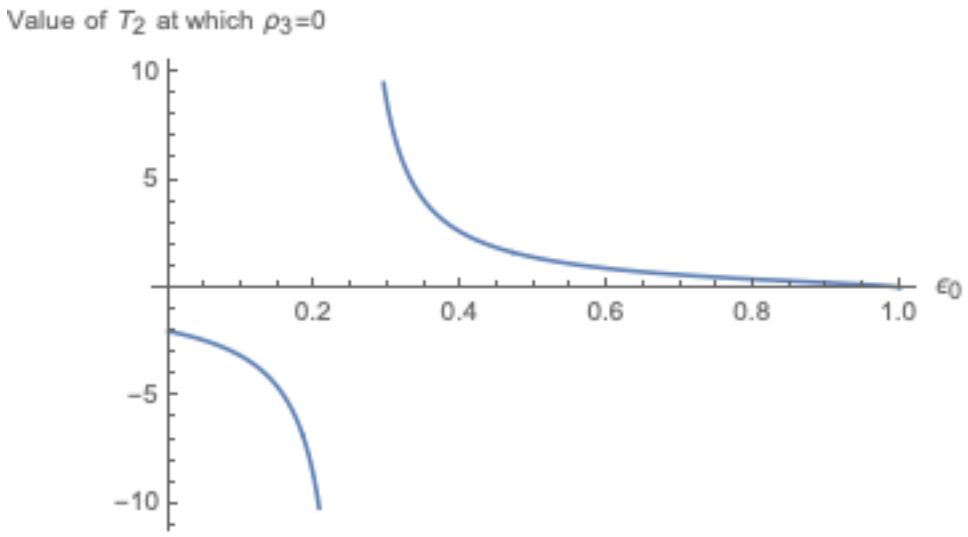


Figure 5.11: In order for ρ_3 to be nonnegative in Case 2 as required, T_2 must lie below this curve (for $\epsilon_0 \in (\frac{1}{4}, \frac{3}{4}]$). Since $T_2 \leq 0.5$ by assumption, this is guaranteed to be the case.

contributing to ρ_1 is still $2T_2\epsilon_0$ and the amount contributing to ρ_2 is now

$$\frac{v_i^{(\epsilon_0)}(0) - T_2}{2}$$

yielding

$$\rho_3(\epsilon_0) = 1 - 2T_2\epsilon_0 - \frac{v_i^{(\epsilon_0)}(0) - T_2}{2}$$

(the area shown in Figure 5.12). It's a bit trickier here to ensure that ρ_3 is nonnegative, which holds whenever

$$\begin{cases} \epsilon_0 < \frac{1}{4}, T_2 \geq -2\sqrt{-\frac{-1+\epsilon_0}{(-1+4\epsilon_0)^2}} \\ \epsilon_0 = \frac{1}{4} \\ \epsilon_0 > \frac{1}{4}, T_2 \leq 2\sqrt{-\frac{-1+\epsilon_0}{(-1+4\epsilon_0)^2}} \end{cases}$$

depicted in Figure 5.11. At first glance this is not necessarily guaranteed. However, recall that $\epsilon_0 \leq \frac{3}{4}$. As such, the restriction $T_2 \leq 0.5$ is, once again, sufficient.

This completes the proof of Lemma 5.16 □

Because ρ_3 is nonnegative, we are guaranteed that ρ_1 and ρ_2 are also nonnegative, and Theorem 5.12 holds.

Proof of Corollary 5.7. We can plot $r_i^{(v)}(T_2)$ (as a function of ϵ_0 and T_2) by its contours (see Fig-

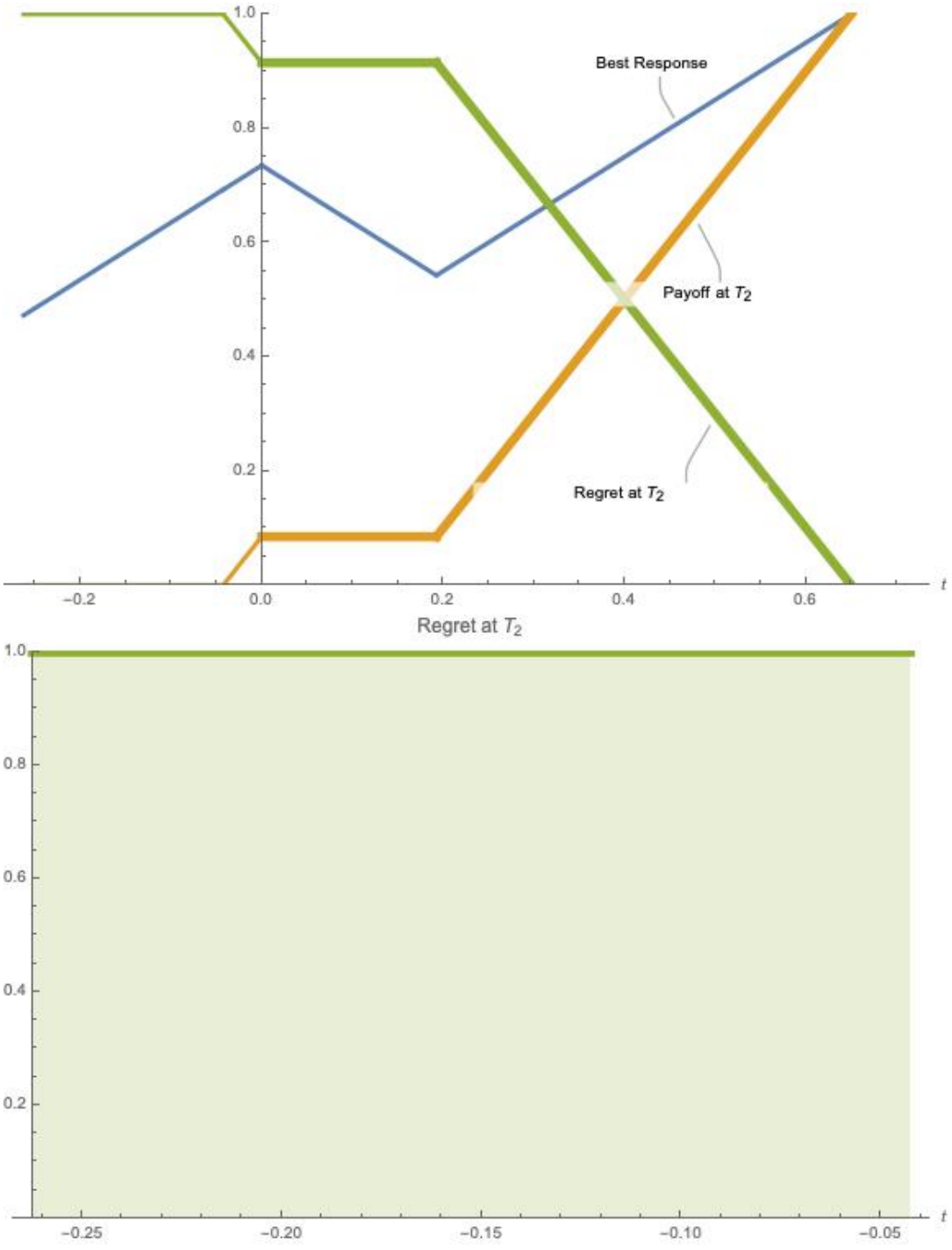


Figure 5.12: $\rho_3(\epsilon_0)$ Dynamics for $1 - v_i^{(\epsilon_0)}(0) < T' \leq v_i^{(\epsilon_0)}(0)$

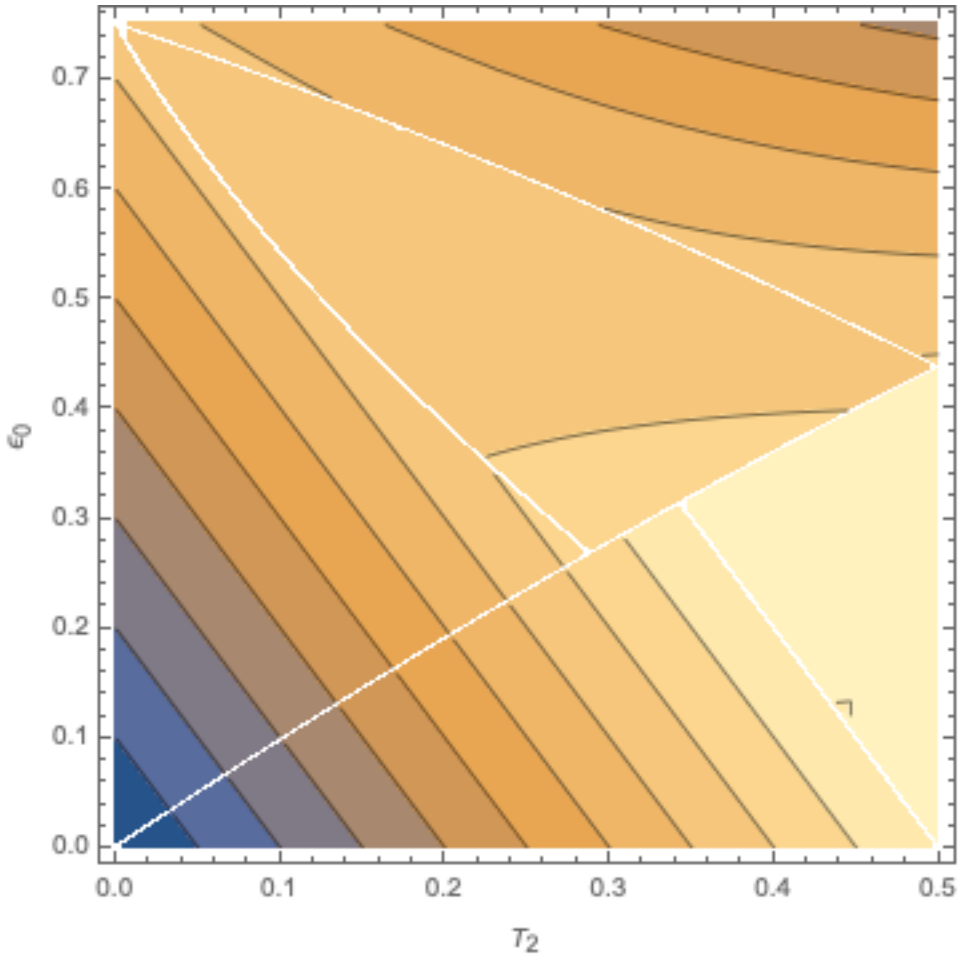


Figure 5.13: A contour plot of $r_i^{(v)}(T_2)$ from 0 (blue) to 1 (yellow) with T_2 on the x-axis and ϵ_0 on the y-axis.

ures 5.13 and 5.14). In order to obtain the value presented in Corollary 5.7, we run a numerical optimization algorithm to bound

$$\min_{T_2 \in [0,.5]} \max_{\epsilon_0 \in [0,.75]} r_i^{(v)}(T_2).$$

Note that the inner optimization is analytical (this only involves computing piecewise polynomial derivatives), while the outer is numerical (it is difficult to optimize analytically over two-dimensional piecewise constraints). Solving the inner optimization yields the graph in Figure 5.15 as a function of T_2. Viewing the orthogonal graph, Figure 5.16 shows the value of $r_i^{(v)}(T_2)$ as a function of ϵ_0 when $T_2 = 0.057$. Notice that, as expected, when $T_2 = 0.057$, for $\epsilon_0 < 0.623$, we use $r_i^{(v)}(T_2) = \epsilon_0 + 2T_2$. It's only for the small range of larger ϵ_0 that the piecewise-defined regret comes in, and allows us to beat a regret of 0.75. □

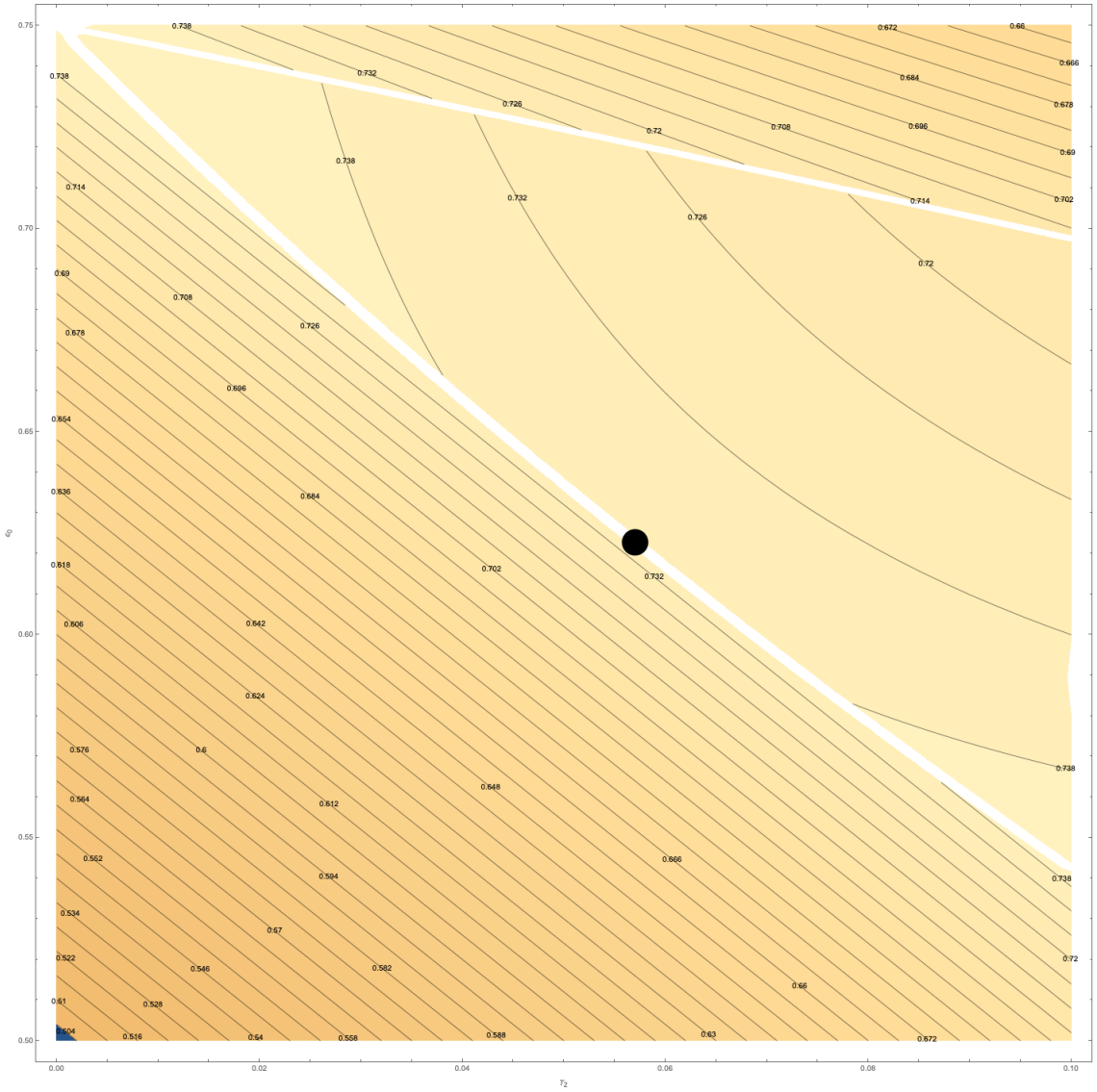


Figure 5.14: Figure 5.13 zoomed in on the relevant area with a point denoting the best choice of T_2 for the adversarial choice of ϵ_0.

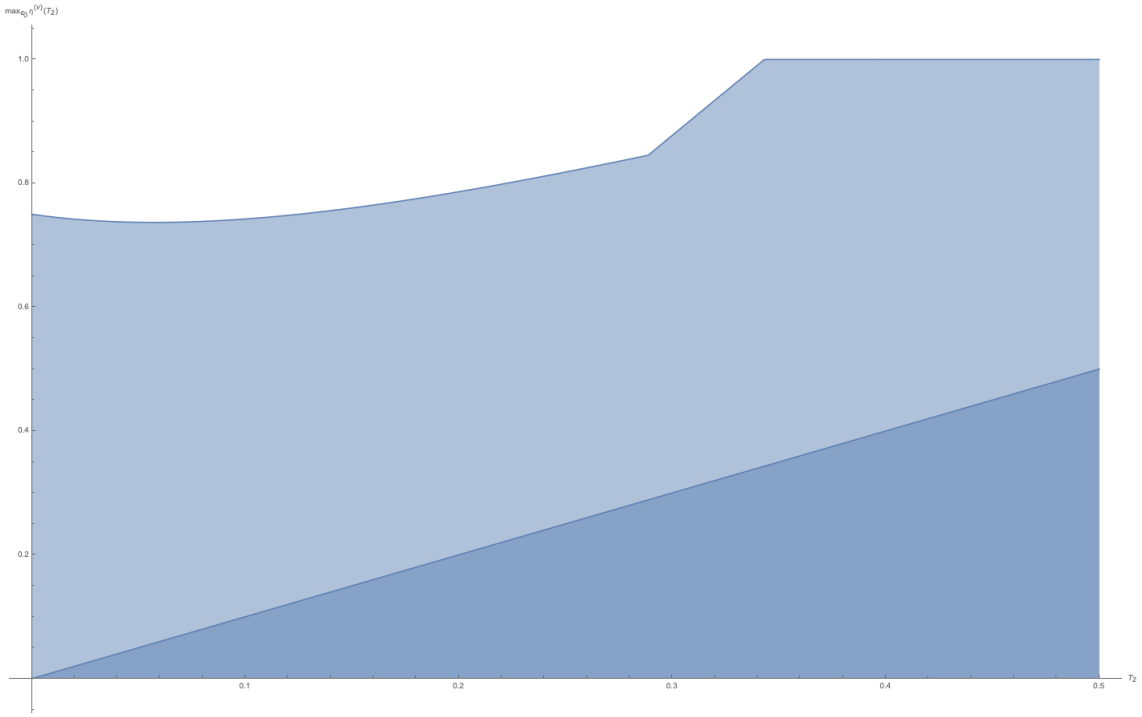


Figure 5.15: The graph of $\max_{\epsilon_0 \in [0,.75]} r_i^{(v)}(T_2)$ as a function of T_2.

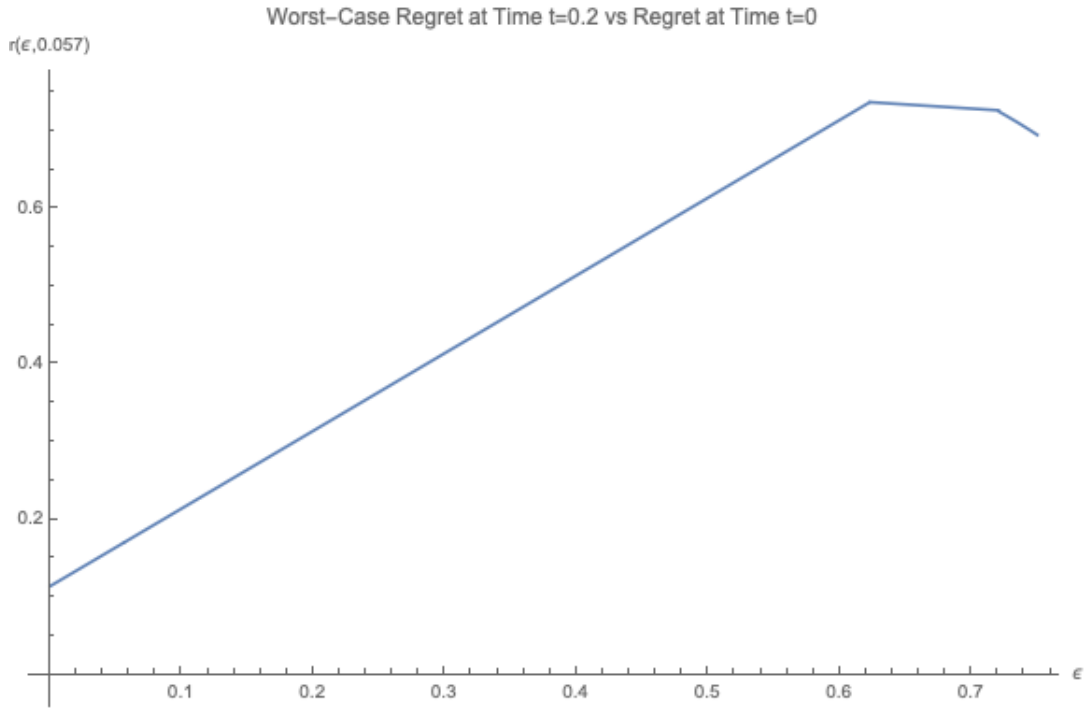


Figure 5.16: The graph of $r_i^{(v)}(T_2)$ as a function of ϵ_0 when $T_2 = 0.057$ is selected.

5.5.7 Justifying the Approximations

Lemma 5.17. *Approximating Stage Three of Algorithm 8 in the continuous limit can underestimate the regret by at most an additive $\frac{1}{N}$. More specifically, the worst-case regret from the discrete adversarial best-response function is at most $\frac{2}{N}$ larger than the worst-case regret from the continuous adversarial best-response function.*

Proof Sketch. As Figure 5.17 shows (with regret displayed in green), the discrete regret is never more than a distance of $\frac{2}{N}$ from the continuous. Including this additive factor when calculating the regret by integrating over the green curves completes the proof. □

If our queries were all exact, this would be almost the end of the story. However, since we are (essentially) taking δ-distribution queries, things are not quite so simple. The risk of incorrect oracle queries appears only when miscalculating the best response and when exiting Stage 2 (the latter is handled below). If the best response is still correct, then it doesn't matter that the queries were inaccurate. Otherwise, only a single step is affected, and this step provides an extra regret for the reassigned probability of at most δ (since it was assumed to fall from there anyway). If this happens every single step, then every reassigned block can give extra additive regret δ.

Lemma 5.18. *Approximating the result of Algorithm 8 with access to a completely accurate oracle underestimates the worst-case regret by at most an additive factor of 2δ.*

Proof Sketch. As can be seen within each of the three sections of Algorithm 8, the oracle is used only to determine the best response of player i at time t. So, if the oracle errs by at most an additive factor of δ, the probability is assigned instead to an action yielding utility less than 2δ below that of the best response. Because throughout we assume all utilities fall at a maximal rate as soon as probability is given to them, no calculated regret can be more than 2δ from the true regret. Since this possible error is carefully handled in the termination condition of Stage 2 in order not to underestimate the average regret, the output of Algorithm 8 with a flawless oracle at most underestimates the regret by 2δ. □

The final approximation we have made is assuming that, at the end of Stage 2, the average regret over all players is at most $\frac{1}{2}$. Theorem 5.6 does not quite guarantee this (we are leaving out an additive $\frac{1}{2N}$, plus the additional possible δ from the innacurate queries as mentioned above). However, the only change this introduces is that $v_i(t)$ is not actually $\frac{1}{N}$-Lipschitz in Stage 3, but is instead $\left(\frac{1+2\delta}{N} + \frac{1}{N^2}\right)$-Lipschitz (since the average rate of reassigning probability is now slightly greater

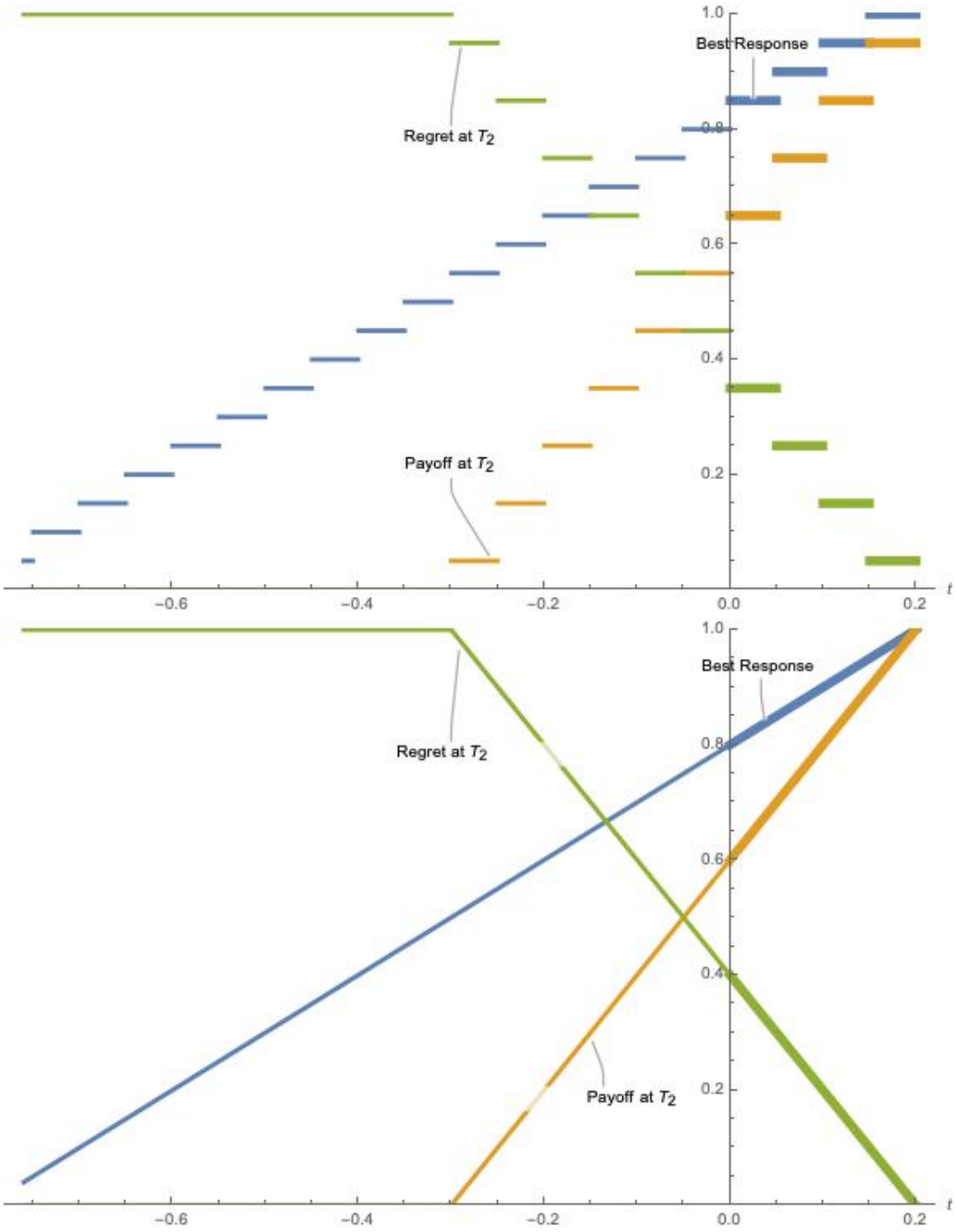


Figure 5.17: Best Response, Payoff, and Regret Dynamics in the Discrete Case (left) and the Continuous Approximation (right)

than 1 in the continuous case, or $\frac{1}{N}$ in the discrete case). As such, the regret of a given player can wind up $\frac{2\delta}{N} + \frac{T_2}{N^2}$ larger than the continuous case we considered.

So:

Proof of Theorem 5.2. Plugging in our parameter definitions,

$$0.7368 + 2\delta + \frac{2 + 2\delta}{N} + \frac{T_2}{N^2} = 0.7368 + 0.0001 + 0.000097 + 0.0000027 = 0.737.$$

Now let's assume that we can directly make δ-distribution queries. Every step of the algorithm requires each player (all n of them) to determine their best response and current payoff (from which they can determine their regret) which requires m additional queries for each player. There are at most $N + T_1 + T_2 = 2.057N$ steps of the algorithm, so we make

$$2.057N(nm + 1)$$

δ-distribution queries, translating to

$$O\left(nm^3 \log\left(\frac{n^2 m}{\eta}\right)\right)$$

profile queries (via Corollary 2.3), completing the proof (we require that each simulated query be correct with probability at least

$$1 - \frac{\eta}{2.057N(nm + 1)}$$

to ensure error probability at most η by a union bound). □

Remark 36. Note that, unfortunately, this does not hold for best-response queries, as we need to know when to break out of Stage 2.

Chapter 6

Deterministic Lower Bounds for General Games

6.1 Overview

We finish this work with a chapter on more general results that are not explicitly focused on Lipschitz games, but were instead derived as intermediate steps to the study of Lipschitz games above and remain of independent interest. More specifically, we present:

- A deterministic dichotomy for finding approximate Nash equilibria of general n-player, m-action games requiring exponentially-many queries to achieve better regret than naively playing the uniform mixed strategy.
- An edge-isoperimetric inequality that applies to a generalization of the hypercube (i.e. graphs that represent games with more than two actions) bounding the possible number of edges with both endpoints in any subset of the m-clique extended to n-dimensions.

6.2 A Nash Lower Bound

In this section, we provide lower bounds on the *deterministic* query complexity of finding approximate Nash equilibria in order to motivate the above consideration of explicitly randomized algorithms.

As a reminder, Hart and Nisan claim the following result:

Theorem 6.1 ([53]). *For any $\epsilon < \frac{1}{2}$, the deterministic profile query complexity of finding ϵ-ACEs of n-player games is $2^{\Omega(n)}$.*

In their work, [53] reduce from the following problem:

Definition 6.2.1 (`ApproximateSink`). Given as input a directional labelling of all edges of the n-dimensional hypercube, the `ApproximateSink` problem is to find a vertex with in-degree at least $\frac{n}{4}$.

While the proof in [53] utilizes a reduction from `ApproximateSink` to provide a lower bound for correlated equilibria, we employ a more streamlined approach when it comes to Nash equilibria, presenting an explicit family of "hard" games that allows us to uncover the optimal value of ϵ as a function of the number of actions:

Theorem 6.2 (Nash Equilibrium Lower Bound). *Given some $m \in \mathbb{N}$, for any $\epsilon < \frac{m-1}{m}$, the deterministic profile query complexity of finding ϵ-ANEs of n-player, m-action games is $2^{\Omega(n)}$.*

Furthermore, this value of ϵ cannot be improved:

Proposition 6.1. Given some $n, m \in \mathbb{N}$, for any $\epsilon \geq \frac{m-1}{m}$, an ϵ-ANE of an n-player, m-action game can be found making no profile queries.

The upper bound of Proposition 6.1 can be met if every player plays the uniform mixed strategy over their actions (thus indicating that playing randomly is the best one can do in a general game without making an exponential number of profile queries). Finally, we can apply Lemma 3.1 to scale Theorem 6.2 and obtain a Lipschitz version of this result:

Corollary 6.3. *Given some $m \in \mathbb{N}, \lambda \in (0, 1]$, for any $\epsilon < \frac{m-1}{m}\lambda$, the deterministic profile query complexity of finding ϵ-ANEs of n-player, m-action, λ-Lipschitz games is $2^{\Omega(n)}$.*

In order to prove these results, we introduce a family of games $\{G_{k,m}\}$. For any $k, m \in \mathbb{N}$, $G_{k,m}$ is a $2k$-player, m-action generalization of k Matching Pennies games in which every odd player $2i-1$ wants to match the even player $2i$ and every even player $2i$ wants to mismatch with the odd player $2i-1$.

Definition 6.2.2 (Generalized Matching Pennies). Define $G_{1,2}$ to be the generalized Matching Pennies game, as described in Figure 6.1(a). Define the generalization $G_{k,m}$ to be the $2k$-player m-action game such that, for any $i \in [k]$, player $2i-1$ has a payoff 1 for matching player $2i$ and 0 otherwise (and vice versa for player $2i$) ignoring all other players.

		Player 2	
		1	2
Player 1	1	$1, 0$	$0, 1$
	2	$0, 1$	$1, 0$

(a) The payoff matrix of $G_{1,2}$, the Matching Pennies game.

		Player 2		
		1	2	3
Player 1	1	$1, 0$	$0, 1$	$0, 1$
	2	$0, 1$	$1, 0$	$0, 1$
	3	$0, 1$	$0, 1$	$1, 0$

(b) The payoff matrix of $G_{1,3}$, the generalized Matching Pennies game.

Figure 6.1: The payoff matrices of $G_{1,2}$ and $G_{1,3}$.

The critical property of the generalized Matching Pennies game is that we can bound the probability that any given action profile is played in any ϵ-ANE of $G_{k,m}$. If too much probability is jointly placed on matching actions, player 2 will have high regret. Conversely, if too much probability is jointly placed on mismatched actions, player 1 will have high regret.

Lemma 6.4. *For any $k, m \in \mathbb{N}, \alpha > 0$, take $\epsilon = \frac{m-1}{m} - \alpha$. In any ϵ-ANE $\mathbf{p}^*$ of $G_{k,m}$, every pair of players $(2i-1, 2i)$ plays any pair of actions (j_{2i-1}, j_{2i}) with probability less than ρ where*

$$\rho = \frac{(2-\alpha)m - 1}{2m}.$$

This phenomenon can be seen in Figure 6.2.

Proof. Define $n = 2k$ to be the number of players in $G_{k,m}$, and consider some ϵ-ANE $\mathbf{p}^*$. Now WLOG consider players 1 and 2 and assume, for the sake of contradiction, that there exist some actions j_1, j_2 such that $p_{1,j_1} p_{2,j_2} \geq \rho$. We will need to consider the two cases $j_1 = j_2$ and $j_1 \neq j_2$.

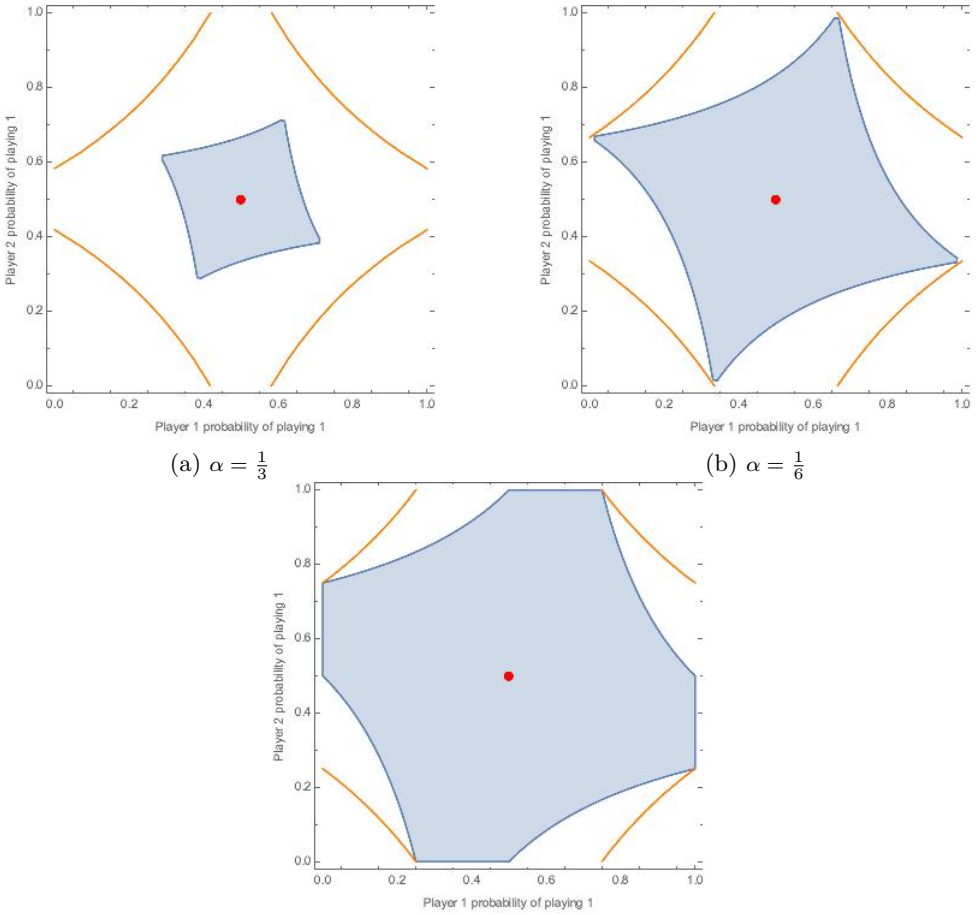
Case 1: Matching Actions

In this case, WLOG assume $j_1 = j_2 = 1$. We show that player 2 can improve her payoff by more than ϵ. Under $\mathbf{p}^*$, with probability $> \rho$, any random realization will yield player 2 a payoff of 0. In other words,

$$u_2(\mathbf{p}^*) < 1 - \rho.$$

Furthermore, we are guaranteed that

$$\sum_{j=2}^{m} p_{1,j} < 1 - \rho$$



(a) $\alpha = \frac{1}{3}$ (b) $\alpha = \frac{1}{6}$

(c) $\alpha \to 0$

Figure 6.2: The region of possible values for $(\mathbf{p}_1^*, \mathbf{p}_2^*)$ in any $\left(\frac{1}{2} - \alpha\right)$-approximate Nash equilibrium of $G_{k,2}$. The only exact Nash equilibrium is shown by the red point, and the corresponding values of ρ are displayed as the orange lines.

so there must exist some action (WLOG action 2) for which

$$p_{1,2} < \frac{1-\rho}{m-1}.$$

As such,

$$u_2(2, \mathbf{p}_{-2}^*) - u_2(\mathbf{p}^*) > 1 - \frac{1-\rho}{m-1},$$

so

$$\mathrm{reg}_2(\mathbf{p}^*) > \underbrace{\left(1 - \frac{1-\rho}{m-1}\right)}_{u_2(2,\mathbf{p}_{-2}^*)} - \underbrace{(1-\rho)}_{u_2(\mathbf{p}^*)} = \frac{\rho m - 1}{m-1} \geq \frac{m-1}{m} - \alpha$$

for all $m \geq 2$. This contradicts our assumption of an ϵ-ANE.

Case 2: Mismatched Actions

In this case, WLOG assume $j_1 = 1, j_2 = 2$. The situation is simpler:

$$u_1(\mathbf{p}^*) < 1 - \rho$$

and

$$u_1(2, \mathbf{p}^*_{-1}) > \rho,$$

so

$$\mathrm{reg}_1(\mathbf{p}^*) > \frac{m-1}{m} - \alpha.$$

This too contradicts our assumption of an ϵ-ANE, and thus completes the proof of Lemma 6.4. □

We can now prove Theorem 6.2. The general idea is that, should an efficient algorithm exist, because any equilibrium of $G_{k,m}$ must have large support by Lemma 6.4, there is significant probability assigned to action profiles that are not queried by the algorithm. We show there is a game that the algorithm cannot distinguish from $G_{k,m}$ that shares no approximate equilibria with $G_{k,m}$.

Proof of Theorem 6.2. Consider any $\alpha > 0$ and let $\epsilon = \frac{m-1}{m} - \alpha$. Taking ρ as in the statement of Lemma 6.4, assume there exists some deterministic algorithm A that takes an n-player, m-action game G as input and finds an ϵ-ANE of G querying the payoffs of $q < \frac{\alpha}{2}\rho^{-\frac{n}{2}}$ action profiles. Fix some $k \in \mathbb{N}$ and consider input $G_{k,m}$ as defined in Definition 6.2.2. Then $\mathbf{p}^* = A(G_{k,m})$ is an ϵ-ANE of $G_{k,m}$. Note that, for some j,

$$p_{1,j} \leq \frac{1}{m}$$

(WLOG assume $j = 1$).

Now define the perturbation $G'_{k,m}$ of $G_{k,m}$ with payoffs defined to be equal to $G_{k,m}$ for every action profile queried by A, 1 for every remaining action profile in which player 1 plays action 1 (chosen because it is assigned low probability by $\mathbf{p}^*$ by assumption), and 0 otherwise. Note that, by definition, A cannot distinguish between $G_{k,m}$ and $G'_{k,m}$, so $A(G'_{k,m}) = \mathbf{p}^*$.

The quantity we need to bound is

$$\mathrm{reg}_i(\mathbf{p}^*) \geq u'_i(1, \mathbf{p}^*_{-1}) - u'_i(\mathbf{p}^*).$$

We must bound the components of this expression as follows:

Claim 2. $u_1'(1, \mathbf{p}_{-1}^*) > (1 - q\rho^{\frac{n}{2}})$

Claim 3. $u_1'(\mathbf{p}^*) < \left(\frac{1}{m} + q\rho^{\frac{n}{2}}\right)$

Using the claim and once again recalling the assumption that $q < \frac{\alpha}{2}\rho^{-\frac{n}{2}}$, we see

$$\mathrm{reg}_1'(\mathbf{p}^*) > \left(1 - \frac{1}{m} - 2q\rho^{\frac{n}{2}}\right) = \frac{m-1}{m} - \alpha = \epsilon.$$

So $\mathbf{p}^*$ cannot actually be an ϵ-ANE of $G_{k,m}$, as we desired. We finish by proving these two claims.

Define the sets $Q_i^{(j)}$ to consist of the pure action profiles queried by A in which player i plays action j (as a basic bound, note that none of these sets can contain more than q elements). The perturbation $G'_{k,m}$ is the game with payoffs that agree with $G_{k,m}$ on all queried action profiles of A, and otherwise provide player 1 a payoff of 1 for playing action 1 and a payoff of 0 otherwise. Because A cannot distinguish between $G_{k,m}$ and $G'_{k,m}$, $\mathbf{p}^* = A(G'_{k,m})$.

Proof of Claim 2. We first separate the payoffs into those which were queried by A and those which were not (i.e. those that differ between $G_{k,m}$ and $G'_{k,m}$ and those that do not):

$$u_1'(1, \mathbf{p}_{-1}^*) = \sum_{\mathbf{a} \in Q_1^{(1)}} \mathbf{p}^*(\mathbf{a}) u_1(1, \mathbf{a}_{-1}) + \sum_{\mathbf{a} \notin Q_1^{(1)}} \mathbf{p}^*(\mathbf{a}).$$

Unfortunately, the best lower bound we can claim for the first sum is 0. However, the second sum is more friendly.

$$u_1'(1, \mathbf{p}_{-1}^*) \geq \sum_{\mathbf{a} \notin Q_1^{(1)}} \mathbf{p}^*(\mathbf{a}) = \left(1 - \sum_{\mathbf{a} \in Q_1^{(1)}} \mathbf{p}^*(\mathbf{a})\right).$$

Combining our assumption on the size of $Q_1^{(1)}$ with the result of Lemma 6.4 (which provides an upper bound on the value of $\mathbf{p}^*(\mathbf{a})$), this yields

$$u_1'(1, \mathbf{p}_{-1}^*) > \left(1 - q\rho^{\frac{n}{2}}\right),$$

proving the Claim. ◁

Proof of Claim 3. It is slightly more involved to obtain an upper bound on the value of $u_1'(\mathbf{p}^*)$.

We first note the following.

$$u_1'(\mathbf{p}^*) = \sum_{\substack{\mathbf{a}\in[m]^n \\ a_1=1}} \mathbf{p}^*(\mathbf{a})u_1'(\mathbf{a}^*) + \sum_{\substack{\mathbf{a}\in[m]^n \\ a_1\neq 1}} \mathbf{p}^*(\mathbf{a})u_1'(\mathbf{a}^*).$$

By assumption, the first expression is upper bounded at $\frac{1}{m}$. Our focus, therefore, must lie on the second expression. This can be broken up further, by the Law of Total probability:

$$\sum_{\substack{\mathbf{a}\in[m]^n \\ a_1\neq 1}} \mathbf{p}^*(\mathbf{a})u_1'(\mathbf{a}^*) = \sum_{\substack{\mathbf{a}\in[m]^n \\ a_1\neq 1 \\ \mathbf{a}\in Q_1^{(a_1)}}} \mathbf{p}^*(\mathbf{a})u_1'(\mathbf{a}^*) + \sum_{\substack{\mathbf{a}\in[m]^n \\ a_1\neq 1 \\ \mathbf{a}\notin Q_1^{(a_1)}}} \mathbf{p}^*(\mathbf{a})u_1'(\mathbf{a}^*).$$

Since $u_1'(\mathbf{p}^*) = 0$ whenever $\mathbf{a} \notin Q_1^{(a_1)}$, we need only consider the case when $\mathbf{a} \in Q_1^{(a_1)}$. While we cannot bound the payoff we can, fortunately, bound the probability

$$\Pr_{\mathbf{a}\sim\mathbf{p}^*}(a_1 \neq 1, \mathbf{a} \in Q_1^{(a_1)}) \leq q\rho^{\frac{n}{2}}.$$

This bounds the entire payoff at

$$u_1'(\mathbf{p}^*) \leq \left(\frac{1}{m} + q\rho^{\frac{n}{2}}\right)$$

(note that we can do even better, since we are double-counting some, but this is sufficient). ◁

Since the two claims hold, this completes the proof of Theorem 6.2. □

6.3 A Discussion of Correlated Equilibria

Of course, when considering correlated equilibria as in [53], Lemma 6.4 no longer holds, so the proof of Theorem 6.2 breaks down. While we present no explicit improvement over the correlated lower bound presented by [53] at this time, there are still a few results of independent interest reproduced below.

6.3.1 The Hyper-m-Clique

In this section, we will be working with a generalization of the hypercube, which we will call the n-dimensional hyper-m-clique.

Definition 6.3.1 (Hyper-m-clique)**.** Define the n-dimensional hyper-m-clique $H_{n,m}$ to be the graph

over vertex set $[m]^n$ such that there is an edge between every pair of vertices that differ by Hamming distance exactly 1. Note that the n-dimensional hypercube is $H_{n,2}$. See Figure 6.3 for a visualization of $H_{2,3}$.

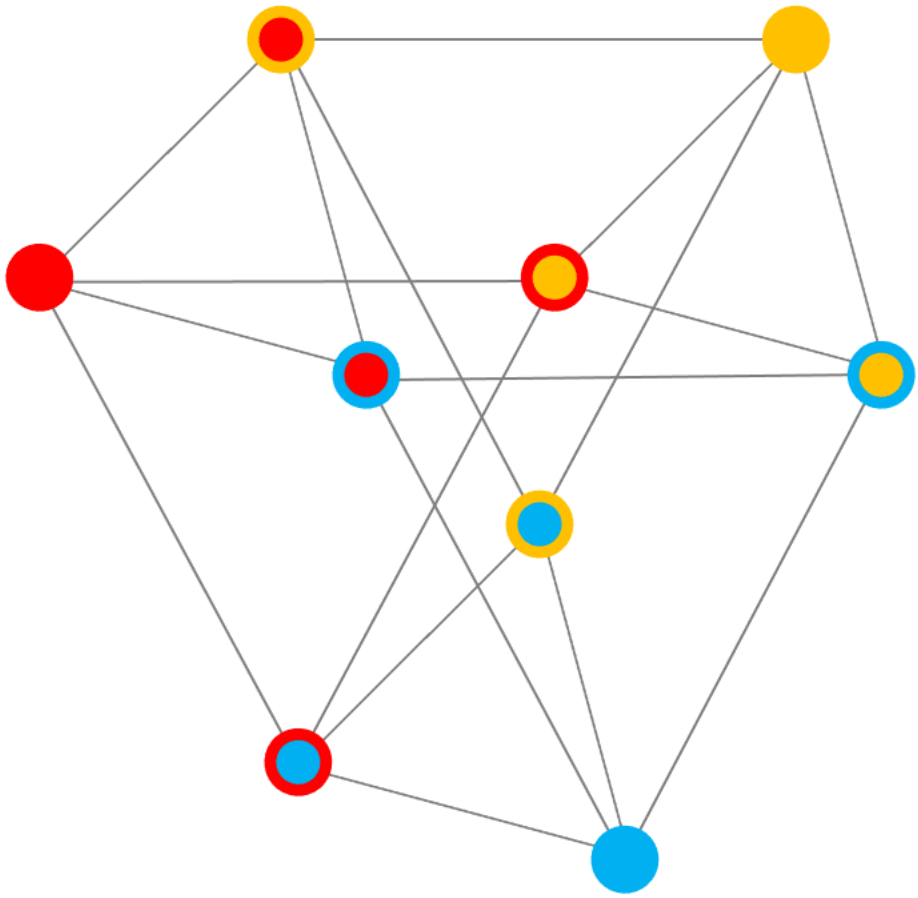


Figure 6.3: The 2-dimensional hyper-3-gon $H_{2,3}$, color-coded for clarity. The fill color represents the first dimension and the border color represents the second, with edges between points of the same fill but different border or points of the same border but different fill.

Furthermore, for notational convenience, for a vertex $v \in V(H_{n,m})$ denote as $v^{(i,j)}$ the vertex with entries equal to $v_{i'}$ at all indices i' not equal to i and equal to j at index i.

Remark 37. Note that $H_{n,m}$ has m^n vertices, each of which is an endpoint of $n(m-1)$ edges.

The edge-isoperimetric inequalities on the hypercube are well-known, and are heavily utilized in [53]:

Theorem 6.5 ([10])**.** *For any (non-empty) subset $S \subseteq V(H_{n,2})$ of the vertices of the hypercube of*

size k, there are at most

$$\frac{k \log k}{2}$$

edges in $E(H_{n,2})$ with both endpoints in S.

Here we will extend the edge-isoperimetric inequality from $H_{n,2}$ to $H_{n,m}$:

Theorem 6.6 (Generalized Edge-Isoperimetric Inequality)**.** *For any subset $S \subseteq V(H_{n,m})$ of size k, there are at most*

$$\frac{1}{2}k(m-1)\log k$$

edges in $E(H_{n,m})$ with both endpoints in S.

The proof of this Theorem follows along the lines of [51], though provides important new insights to extend the proof to $H_{n,m}$.

In order to show this, we will need the following identity.

Lemma 6.7. *Taking $\gamma_{i,i'} = k_{i'}/k_i$ for $k_1, \ldots, k_m \in \mathbb{N}$,*

$$\prod_{i=1}^{m}\left(1 + \sum_{i' \neq i} \gamma_{i,i'}\right) \geq m^m.$$

Proof. Note that

$$\prod_{i=1}^{m}\left(1 + \sum_{i' \neq i} \gamma_{i,i'}\right) = \frac{(\sum_{i=1}^{m} k_i)^m}{\prod_{i=1}^{m} k_i}.$$

From here, we take the gradient and end up with the system:

$$mk_i = \sum_{j=1}^{m} k_j$$

for every $i \in [m]$. In other words, at the minimum, $k_1 = \ldots = k_m$, so

$$\prod_{i=1}^{m}\left(1 + \sum_{i' \neq i} \gamma_{i,i'}\right) \geq \prod_{i=1}^{m} m = m^m.$$

□

Additionally, define the average degree of a graph to be the average over all vertices in the graph of the number of edges adjacent to that vertex (note that the product of the average degree with the

number of vertices will result in twice the number of edges, as each edge is adjacent to two vertices). Then, in order to prove Theorem 6.6, we require:

Lemma 6.8. *For any (non-empty) subgraph Z of $H_{n,m}$ with average degree $\bar{d}$,*

$$|V(Z)| \geq 2^{\bar{d}/(m-1)}.$$

Proof. We show this by induction on n. The base case clearly holds - $H_{1,m}$ is a clique on m vertices, so if $|V(Z)| = k$, then there are exactly $\frac{k(k-1)}{2}$ vertices, so $\bar{d} = k-1$. Since $k \leq m$, clearly either $k = 1$ and $\bar{d} = 0$, or $k \geq 2 \geq 2^{(k-1)/(m-1)}$.

Now, for the inductive hypothesis, assume that $|V(Z')| \geq 2^{\bar{d}/(m-1)}$ for any non-empty subgraph Z' of $H_{n-1,m}$, and consider some non-empty subgraph Z of $H_{n,m}$ of size k. We can partition the vertices of Z into sets $Z_1, \ldots, Z_m$ of size $k_1, \ldots, k_m$ based on the final index, each of which is a subset of the vertices of $H_{n-1,m}$ (WLOG assume $k_1 \leq \ldots \leq k_m$). Define s_{i_1,i_2} to be the number of edges of Z with one endpoint in Z_{i_1} and the other in Z_{i_2}. For each vertex in Z_{i_1}, there can be at most one edge crossing into Z_{i_2}, so $s_{i_1,i_2} \leq \min\{k_{i_1}, k_{i_2}\}$.

The inductive hypothesis gives us, for any non-empty S_i,

$$(m-1)\log k_i \geq \frac{1}{k_i}\sum_{v \in Z_i} d_{Z_i}(v)$$

(note that the right-hand side is the average degree of Z_i). Multiplying through:

$$(m-1)k_i \log k_i \geq \sum_{v \in Z_i} d_{Z_i}(v) = \sum_{v \in Z_i} d_Z(v) - \sum_{i' \neq i} s_{i,i'}.$$

Note that this even holds if $k_i = 0$, so we need not rely on the assumption that S_i is non-empty. Taking a sum over the partition:

$$(m-1)\sum_{i=1}^{m} k_i \log k_i + \sum_{i=1}^{m}\sum_{i' \neq i} s_{i,i'} \geq \sum_{v \in Z} d_Z(v).$$

Finally, substituting in our bound on $s_{i,i'}$, we obtain

$$(m-1)\sum_{i=1}^{m} k_i \log k_i + 2\sum_{i=1}^{m}(m-i)k_i \geq \sum_{v \in Z} d_Z(v). \tag{6.1}$$

So consider the value $k \log_{2^{1/(m-1)}} k$. We note

$$\begin{aligned} k \log_{2^{1/(m-1)}} k &= \left(\sum_{i=1}^{m} k_i\right) \log_{2^{1/(m-1)}} \left(\sum_{i=1}^{m} k_i\right) \\ &\geq \sum_{i=1}^{m} k_i \log_{2^{1/(m-1)}} \left(k_i \left(1 + \sum_{i' \neq i} \gamma_{i,i'}\right)\right) \\ &= (m-1) \sum_{i=1}^{m} k_i \log k_i + \sum_{i=1}^{m} k_i \log_{2^{1/(m-1)}} \left(1 + \sum_{i' \neq i} \gamma_{i,i'}\right) \end{aligned}$$

where $\gamma_{i,i'} = \frac{k_{i'}}{k_i}$. Now, consider the final two terms in the second sum:

$$\begin{aligned} & k_{m-1} \log_{2^{1/(m-1)}} \left(1 + \sum_{i' \neq m-1} \gamma_{m-1,i'}\right) + k_m \log_{2^{1/(m-1)}} \left(1 + \sum_{i' \neq m} \gamma_{m,i'}\right) \\ &\geq k_{m-1} \left(\log_{2^{1/(m-1)}} \left(1 + \sum_{i' \neq m-1} \gamma_{m-1,i'}\right) + \log_{2^{1/(m-1)}} \left(1 + \sum_{i' \neq m} \gamma_{m,i'}\right)\right) \\ &= k_{m-1} \left(\log_{2^{1/(m-1)}} \left(1 + \sum_{i' \neq m-1} \gamma_{m-1,i'}\right) \left(1 + \sum_{i' \neq m} \gamma_{m,i'}\right)\right) \\ &= 2k_{m-1} + k_{m-1} \left(\log_{2^{1/(m-1)}} \frac{1}{2^{\frac{2}{m-1}}} \left(1 + \sum_{i' \neq m-1} \gamma_{m-1,i'}\right) \left(1 + \sum_{i' \neq m} \gamma_{m,i'}\right)\right). \end{aligned}$$

Adding in the third term, we have to take care that the logarithm is nonnegative. However, this follows along the lines of Lemma 6.7.

$$\begin{aligned} & \sum_{i=m-2}^{m} k_i \log_{2^{1/(m-1)}} \left(1 + \sum_{i' \neq i} \gamma_{i,i'}\right) \\ &\geq 2k_{m-1} + k_{m-2} \left(\log_{2^{1/(m-1)}} \frac{1}{2^{\frac{2}{m-1}}} \left(1 + \sum_{i' \neq m-2} \gamma_{m-2,i'}\right) \left(1 + \sum_{i' \neq m-1} \gamma_{m-1,i'}\right) \left(1 + \sum_{i' \neq m} \gamma_{m,i'}\right)\right) \\ &= 2k_{m-1} + 4k_{m-2} + k_{m-2} \log_{2^{1/(m-1)}} \frac{1}{2^{\frac{2}{m-1}} 2^{\frac{4}{m-1}}} \left(1 + \sum_{i' \neq m-2} \gamma_{m-2,i'}\right) \left(1 + \sum_{i' \neq m-1} \gamma_{m-1,i'}\right) \left(1 + \sum_{i' \neq m} \gamma_{m,i'}\right). \end{aligned}$$

Continuing along these lines:

$$\sum_{i=1}^{m} k_i \log_{2^{1/(m-1)}} \left(1 + \sum_{i' \neq i} \gamma_{i,i'}\right) \geq 2 \sum_{i=1}^{m} (m-i) k_i + k_1 \log_{2^{1/(m-1)}} \frac{\prod_{i=1}^{m} \left(1 + \sum_{i' \neq i} \gamma_{i,i'}\right)}{\prod_{i=1}^{m} 2^{\frac{2i}{m-1}}}$$

We now need to bound the value of the logarithm. We use Lemma 6.7 above and the arithmetic series

formula to note that

$$\log_{2^{1/(m-1)}} \frac{\prod_{i=1}^{m} \left(1 + \sum_{i' \neq i} \gamma_{i,i'}\right)}{\prod_{i=1}^{m} 2^{\frac{2i}{m-1}}} \geq \log_{2^{1/(m-1)}} \frac{m^m}{2^m} = m \log_{2^{1/(m-1)}} \frac{m}{2} \geq 0.$$

Putting all this together yields

$$\begin{aligned} & k \log_{2^{1/(m-1)}} k \\ & \geq (m-1) \sum_{i=1}^{m} k_i \log k_i + 2 \sum_{i=1}^{m} (m-i) k_i \end{aligned}$$

which, by Equation 6.1, in turn tells us that

$$\log_{2^{1/(m-1)}} k \geq \bar{d}$$

so

$$k \geq 2^{\bar{d}/(m-1)}$$

and the induction is complete. □

Theorem 6.6 follows easily from Lemma 6.8: since the average degree of the subgraph defined by S is at most $\log_{2^{1/(m-1)}} k$, (ensuring not to double count every edge) we are left with at most $\frac{1}{2} k(m-1) \log k$ edges with both endpoints in S.

6.3.2 Correlated (and Coarse Correlated) Equilibria

As noted above, the proof of Theorem 6.2 does not hold for correlated equilibria. It breaks down in Lemma 6.4: if every pair of players $(2i-1, 2i)$ plays a given combination of actions with probability at most ρ, then all we can say is $\mathbf{X}^*(\mathbf{a}) < \rho$ instead of $\rho^{\frac{n}{2}}$.

In fact, this issue can be expressed in a more motivated way. Our proof of Theorem 6.2 actually shows the stronger result:

Theorem 6.9 (Nash Equilibrium Decision Lower Bound)**.** *Given some $m \in \mathbb{N}$, for any $\epsilon < \frac{m-1}{m}$, the deterministic profile query complexity of* verifying *ϵ-ANEs of n-player, m-action games is $2^{\Omega(n)}$.*

In other words, given a strategy profile $\mathbf{p}$, an exponential number of queries are required to determine whether $\mathbf{p}$ is a Nash equilibrium. We prove above that this in fact applies to every single

equilibrium of $G_{k,m}$ (since it shares no approximate equilibria with its perturbations, any algorithm needs to distinguish between the two games in order to find or verify equilibria), which in turn implies hardness for the search problem. In contrast, consider the correlated strategy profile in which two values $r_o, r_e \in [m]$ are chosen uniformly at random, and every odd player plays action r_o while every even player plays action r_e. It may be difficult to find this equilibrium, but it's easy to verify that it is, in fact, a correlated equilibrium - an algorithm would need only query the polynomially-many possible deviations.

Coarse Correlated Equilibria with Subexponential Support Size

In fact, any correlated equilibrium with subexponential support can be verified making a subexponential number of queries. This provides a significant obstacle to extending the above to correlated equilibria. Instead, we can generalize the proof technique of [53]. The result does not, however, extend directly, and is unable to break the 1/2 barrier. However, the generalized technique is still included below as a potential step towards approaching a result along the lines of Theorem 6.2.

Theorem 6.10 (Coarse Correlated Equilibrium Lower Bound)**.** *Given some $m \in \mathbb{N}$, for any $\epsilon < \frac{m-1}{2m}$, the deterministic profile query complexity of finding ϵ-CCEs of n-player, m-action games with support size $2^{o(n)}$ is $2^{\Omega(n)}$.*

This section will proceed similarly to [53], but we include all the proofs below for completeness. We will begin by considering the m−`ApproximateSink` problem - the extension of `ApproximateSink` to hyper-m-cubes:

Definition 6.3.2 (m−`ApproximateSink`)**.** Given some $m \in \mathbb{N}$, the m−`ApproximateSink` problem is defined as follows: given a labelling of the edges of $H_{n,m}$, find a vertex with in-degree at least $\frac{n(m-1)}{2m}$ (i.e. at least a $\frac{1}{2m}$ fraction of the edges are incoming). A query of this problem is specified by a point $\mathbf{a} \in [m]^n$ and returns the vector of $n(m-1)$ values $R(\mathbf{a}, \mathbf{a}^{(i,j)}) \in \{-1, 1\}$ for each $i \in [n], j \in [m] \setminus \{a_i\}$ describing the direction of the edge between $\mathbf{a}$ and the neighbor in which index i is replaced by j (-1 if the edge is entering $\mathbf{a}$, and 1 if the edge is leaving).

Obviously a solution to m−`ApproximateSink` always exists by a simple counting argument. However, even as m gets large (and the required fraction of incoming edges gets small), this problem requires $2^{\Omega(n)}$ queries to solve deterministically:

Theorem 6.11 (m−`ApproximateSink` Lower Bound)**.** *Given some $m \in \mathbb{N}$, the deterministic query complexity of m−`ApproximateSink` is $2^{\Omega(n)}$.*

Remark 38. In fact, this holds for the version of m−`ApproximateSink` requiring a vertex with in-degree at least $\frac{n(m-1)}{cm}$ for any constant $c \geq 2$ (or even any c a function of m).

There are a number of steps required to prove this result. First, we prove the problem cannot be solved by a subclass of algorithms that [53] named "Polite Algorithms".

Definition 6.3.3 (Polite Algorithm)**.** An algorithm solving m−`ApproximateSink` is "polite" if, whenever a vertex v is queried, at least $\frac{n(m-1)}{2m}$ of its neighbors have not been queried.

With this definition in mind, we can directly extend the following Lemma to m−`ApproximateSink`:

Lemma 6.12. *Given some $m \in \mathbb{N}$, no deterministic polite algorithm can solve m−`ApproximateSink`.*

Proof. This proof follows directly that of [53]: for any vertex v, because at least $\frac{n(m-1)}{2m}$ of its neighbors have not been queried, if they are adversarially all set to leave v then v cannot be a solution to the problem. □

We can continue following the proof technique of [53].

Definition 6.3.4 (Closure)**.** For any set $V \subseteq [m]^n$ of vertices in $H_{n,m}$, define its closure $V^* \subseteq [m]^n$ to be the smallest set containing V and every point with more than $\frac{n(m-1)}{4m}$ of its neighbors in V^*.

Lemma 6.13. *If $|V| < 2^{n/(4m)-1}$, then $|V^*| \leq 2|V|$.*

Proof. As in [53], we proceed by contradiction. Assume $|V^*| > 2|V|$, and let U be the set obtained by starting with V and repeatedly adding elements with more than $\frac{n(m-1)}{4m}$ neighbors in U until $|U| = 2|V|$. Define the value

$$e(U) = |\{(u, i, j) \mid u \in U, u^{(i,j)} \in U\}|$$

(i.e. twice the total number of edges of $H_{n,m}$ for which both endpoints lie in U).

We can easily lower bound this value. Every time we add a vertex it must add at least $\frac{n(m-1)}{4m}$ edges to U, so (because each edge is counted twice in $e(U)$) we must have

$$e(U) \geq |U \setminus V| \frac{n(m-1)}{2m} = |V| \frac{n(m-1)}{2m}.$$

Now, for an upper bound. We will rely on the result of Theorem 6.6 to note that $e(U) \leq |U|(m-1)\log|U| = 2|V|(m-1)(\log|V|+1)$. Since the lower bound must lie below the upper bound, we see

$$\frac{n(m-1)}{2m}|V| \leq 2|V|(m-1)(\log|V|+1).$$

Simplifying, this gives us

$$\frac{n}{4m} - 1 \leq \log|V|$$

which contradicts the assumption on the size of V. □

Finally, we complete the proof of Theorem 6.11 with:

Lemma 6.14. *Every algorithm that makes at most $T \leq 2^{n/(4m)-1}$ queries can be simulated by a polite algorithm that makes at most $2T$ queries.*

This can be proven identically to [53], taking care to include the extra factor of $(m-1)$ out of the edge-isoperimetric inequality. This final step completes the proof of Theorem 6.11.

Finally, we can prove Theorem 6.10 via a query-efficient reduction from m−`ApproximateSink`.

Theorem 6.10. This is proven via reduction to m−`ApproximateSink`. Assume for contradiction that an ϵ-CCE can be found with cost $2^{o(n)}$. Consider any labelling of the edges of $H_{n,m}$, and construct the n-player, m-action game defined by the payoff function:

$$u_i(\mathbf{a}) = \frac{1}{2} - \frac{\sum_{j=1}^{m} R(\mathbf{a}, \mathbf{a}^{(i,j)})}{2(m-1)}.$$

Then, in any ϵ-CCE $\mathbf{X}^*$ of this game,

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \mathrm{reg}_i^{(j)}(\mathbf{X}^*) \leq nm\epsilon.$$

Expanding the LHS:

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{m} \mathrm{reg}_i^{(j)}(\mathbf{X}^*) &= \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{\mathbf{a}\in[m]^n} \mathbf{X}^*(\mathbf{a})\left(u_i(j,\mathbf{a}_{-i}) - u_i(\mathbf{a})\right) \\ &= \sum_{\mathbf{a}\in[m]^n} \mathbf{X}^*(\mathbf{a})\left(\sum_{i=1}^{n}\sum_{j=1}^{m} u_i(j,\mathbf{a}_{-i}) - \sum_{i=1}^{n}\sum_{j=1}^{m} u_i(\mathbf{a})\right).\end{aligned}$$

By the definition of our game, for a given player $i \in [n]$,

$$\sum_{j=1}^{m} u_i(j, \mathbf{a}_{-i}) = \frac{m}{2}$$

so we can continue with

$$\begin{aligned}
\sum_{i=1}^{n}\sum_{j=1}^{m} \mathrm{reg}_i^{(j)}(\mathbf{X}^*) &= \sum_{\mathbf{a}\in[m]^n} \mathbf{X}^*(\mathbf{a}) \left(\frac{nm}{2} - \sum_{i=1}^{n}\sum_{j=1}^{m} \left(\frac{1}{2} - \frac{\sum_{j'=1}^{m} R(\mathbf{a}, \mathbf{a}^{(i,j')})}{2(m-1)} \right) \right) \\
&= \sum_{\mathbf{a}\in[m]^n} \mathbf{X}^*(\mathbf{a}) \left(\frac{nm}{2} - \frac{nm}{2} + \frac{1}{2(m-1)} \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{j'=1}^{m} R(\mathbf{a}, \mathbf{a}^{(i,j')}) \right) \\
&= \frac{m}{2(m-1)} \sum_{\mathbf{a}\in[m]^n} \mathbf{X}^*(\mathbf{a}) \sum_{i=1}^{n}\sum_{j'=1}^{m} R(\mathbf{a}, \mathbf{a}^{(i,j')}).
\end{aligned}$$

Putting these together:

$$\sum_{\mathbf{a}\in[m]^n} \mathbf{X}^*(\mathbf{a}) \sum_{i=1}^{n}\sum_{j'=1}^{m} R(\mathbf{a}, \mathbf{a}^{(i,j')}) \leq 2n(m-1)\epsilon.$$

In particular, this means that there must exist a specific $\mathbf{a}$ in the support of $\mathbf{X}^*$ at which

$$\sum_{i=1}^{n}\sum_{j'=1}^{m} R(\mathbf{a}, \mathbf{a}^{(i,j')}) \leq 2n(m-1)\epsilon.$$

So, taking

$$\epsilon = \frac{m-1}{2m},$$

we see

$$\sum_{i=1}^{n}\sum_{j'=1}^{m} R(\mathbf{a}, \mathbf{a}^{(i,j')}) \leq \frac{n(m-1)^2}{m},$$

so the in-degree at $\mathbf{a}$ must be at least $\frac{n(m-1)}{2m}$.

So, if the support of this ϵ-CCE $\mathbf{X}^*$ is at most $2^{o(n)}$ and we can find $\mathbf{X}^*$ in at most $2^{o(n)}$ queries, we can then query every profile in the support (which has subexponential size by assumption) to find $\mathbf{a}$ in at most another $2^{o(n)}$ queries, solving m−`ApproximateSink` in at most $2^{o(n)}$ queries, contradicting Theorem 6.11 and completing the proof. □

Clearly the value of 2 in the denominator of the payoff functions presents a barrier against a

lower bound above $\epsilon = \frac{1}{2}$ for a proof along these lines. Should a tighter reduction exist this technique provides hope for closing the gap between the known $\frac{1}{2}$ lower bound and the trivial $\frac{m-1}{m}$ upper bound.

Coarse Correlated Equilibria with Arbitrary Support Size

Unfortunately, Theorem 6.10 leaves open the query complexity of finding approximate coarse correlated equilibria with exponential support size. As such, we show that we can always convert a coarse correlated equilibrium to one with small support making few additional queries, albeit at significant expense to the approximation factor. Continuing along the lines of [53]:

Lemma 6.15. *In any ϵ-CCE $\mathbf{X}^*$ of an n-player, m-action game G in which all payoffs are at most α, any algorithm that queries a set Q of action profiles can place no more than $\frac{m}{m-1}(\alpha+\epsilon)$ probability on profiles outside of the set*

$$Q' = \{\mathbf{a} \mid \exists j (j, \mathbf{a}_{-1}) \in Q\}.$$

Proof. Assume for the sake of contradiction that this is not the case, and WLOG assume that, in this equilibrium $\mathbf{X}^*$, at most a $\frac{1}{m}$ of the probability assigned outside of Q' is placed on player 1 playing action 1 (so

$$\sum_{\mathbf{a} \notin Q', a_1 \neq 1} \mathbf{X}^*(\mathbf{a}) > \alpha + \epsilon$$

under these assumptions).

Now consider the two games G' in which $u_1(\mathbf{a}) = 0$ for all $\mathbf{a} \notin Q'$, and G'' in which

$$u_1(\mathbf{a}) = \begin{cases} 1 & a_1 = 1 \\ 0 & a_1 \neq 1 \end{cases}$$

for all $\mathbf{a} \notin Q'$.

Clearly,

$$\left| \mathrm{reg}_1^{\prime(\phi)}(\mathbf{X}^*) - \mathrm{reg}_1^{\prime\prime(\phi)}(\mathbf{X}^*) \right| = \sum_{\mathbf{a} \notin Q', a_1 \neq 1} \mathbf{X}^*(\mathbf{a}) > \alpha + \epsilon \tag{6.2}$$

for the constant deviation function $\phi(j) = 1$. However, note that $u_1'(\mathbf{a}) \leq \alpha$ for all $\mathbf{a}$, so

$$|\mathrm{reg}_1'(\mathbf{X}^*)| \leq \alpha.$$

Combining this with Equation 6.2 yields

$$\mathrm{reg}_1''(\mathbf{X}^*) > \epsilon.$$

This contradicts our assumption (since it is very possible that $G = G''$) and thus proves the Lemma.

□

Lemma 6.16. *Any algorithm computing an ϵ-CCE making at most T queries admits an algorithm computing an ϵ'-CCE with cost $2T$ where*

$$\epsilon' = \frac{2\sqrt{\epsilon(m-1)^3 m} + \epsilon m((3-\epsilon)m - 3)}{((1-\epsilon)m - 1)^2}.$$

Proof. Take any game G. We will begin by scaling the payoffs along the lines of Lemma 3.1 to obtain a game G' with no payoffs greater than α. Make T queries to find an ϵ-CCE of G', which is also an ϵ/α-CCE $\mathbf{X}'^*$ of G. Defining Q and Q' as in the proof of Lemma 6.15, consider the correlated profile $\mathbf{X}^*$ that places no probability on action profiles $\mathbf{a} \notin Q'$ and instead scales up the probability assigned to profiles $\mathbf{a} \in Q'$ by a factor of

$$\beta = \frac{1}{\Pr_{\mathbf{a} \in \mathbf{X}'^*}(\mathbf{a} \in Q')}.$$

Since Lemma 6.15 guarantees that

$$\Pr_{\mathbf{a} \in \mathbf{X}'^*}(\mathbf{a} \notin Q') \leq \frac{m}{m-1}(\alpha + \epsilon),$$

we can guarantee that

$$\beta \leq \frac{1}{1 - \frac{m}{m-1}(\alpha + \epsilon)}.$$

For any constant deviation function $\phi \in [m]$ and any player $i \in [n]$, we can bound the regret:

$$\begin{aligned}\sum_{\mathbf{a}\in[m]^n} \mathbf{X}^*(\mathbf{a})\left(u_i(\phi(a_i),\mathbf{a}_{-i}) - u_i(\mathbf{a})\right) &= \beta \sum_{\mathbf{a}\in Q'} \mathbf{X}'^*(\mathbf{a})\left(u_i(\phi(a_i),\mathbf{a}_{-i}) - u_i(\mathbf{a})\right) \\ &\leq \beta \sum_{\mathbf{a}\in[m]^n} \mathbf{X}'^*(\mathbf{a})\left(u_i(\phi(a_i),\mathbf{a}_{-i}) - u_i(\mathbf{a})\right) \\ &\quad + \beta \sum_{\mathbf{a}\notin Q'} \mathbf{X}'^*(\mathbf{a})\left|u_i(\phi(a_i),\mathbf{a}_{-i}) - u_i(\mathbf{a})\right| \\ &\leq \frac{1}{1-\frac{m}{m-1}(\alpha+\epsilon)}\left(\frac{\epsilon}{\alpha} + \frac{m}{m-1}(\epsilon+\alpha)\right).\end{aligned}$$

Finding the value of α that minimizes this (confirming that it is in the range $(0,1)$) completes the proof. □

The following Corollary follows directly from applying Lemma 6.16 to Theorem 6.10 (with arbitrary value of c as noted in Remark 38):

Corollary 6.17. *Given some $m \in \mathbb{N}$, the deterministic* query complexity *of finding an ϵ-CCE of an n-player, m-action game is $2^{\Omega(n)}$ for*

$$\epsilon < \frac{5m(m-1) - 2\sqrt{6m^2(m-1)^2}}{3m^2}.$$

However, this result is fairly weak and leaves a lot of room for improvement, as these values of ϵ form a sequence (albeit increasing) approaching

$$\frac{1}{3}(5 - 2\sqrt{6}) \approx 0.034.$$

Chapter 7

Conclusion

The preceding chapters take a deep dive into the complexity of Lipschitz games. We analyze both the query and computational complexity of finding pure Nash equilibria of Lipschitz games. On the query complexity side, we are able to achieve a lower bound close to that of finding mixed Nash equilibria of general games, introducing only a factor of $\frac{\lambda}{\epsilon}$ to the exponent. On the computational complexity side, for $\epsilon = o(n\lambda)$, we once again show a **PPAD**-hardness lower bound. In both instances, the Lipschitz restriction appears unable to aid much in finding Nash equilibria.

We introduce a few interesting concepts that call for further study - among them Multi-Lipschitz games and the complexity class **BPPPAD** = **RPPAD**.

We discuss and improve upper bounds for finding equilibria of unrestricted Lipschitz games, and break the $\epsilon = \frac{3}{4}$ barrier suggested by the results of [41].

Finally, we analyze deterministic considerations in general games and how they relate to Lipschitz games. In the process, we extended the edge-isoperimetric inequalities to a generalization of the hypercube.

Of course, there is always more to study. We conclude with a handful of problems that remain unsolved by our work above.

7.1 Open Questions

There are certainly some unanswered questions introduced by the studies in the preceding chapters. We present some of them below.

Open Question 1. Can a better lower bound be obtained on the query complexity of finding ϵ-PNEs of games in $\mathcal{G}(n, m, \lambda)$ for $\epsilon = \Theta(n\lambda)$?

Theorem 2.31 begins to address this question, but what if $\epsilon < \frac{1}{8}$, or $m > 2$? For $m > 2$ we have *some* idea, but:

Open Question 2. Can we improve the upper bound achieved by the extended block equilibrium computation algorithm presented in Theorem 5.2?

Similar questions can be asked pertaining to the computational complexity results obtained above.

Open Question 3. What is the computational complexity of computing an ϵ-PNE of an n-player, m-action, λ-Lipschitz polymatrix game where

$$\epsilon = O(\lambda\sqrt{n \log n})$$

(and m is held constant)?

In other words, can we close the gap between the results of Theorem 4.2 (the proof of **PPAD** containment when $\epsilon = \Omega(\lambda\sqrt[3]{n^2})$) and Theorem 2.29 (the existence of an equilibrium when $\epsilon = \Omega(\lambda\sqrt{n \log n})$? Showing that such an equilibrium cannot be found in **PPAD** would imply that $\mathbf{P} \neq \mathbf{BPP}$, so we do not expect such a claim to hold.

Furthermore, even the existence claim leaves room for potential improvement. While Theorem 2.29 conclusively shows that the existence guarantee of $\epsilon = \Theta(n\sqrt{n \log n})$ is tight in *general* games, [28] show existence for $\epsilon = \Omega(\lambda)$ in Lipschitz *anonymous* games.

Open Question 4. For what values of ϵ are there guaranteed to exist ϵ-PNEs of n-player, m-action, λ-Lipschitz polymatrix games?

Once we begin considering computational complexity, the question we consider becomes more of a promise problem in order to remain total (we assume that the input circuit provides Lipschitz payoffs and allow the algorithm to err if the promise is not satisfied). This, of course, begs the question:

Open Question 5. Are there **PPAD**/**RPPAD** algorithms that output an ϵ-PNE *or* a violation of the Lipschitz property (for appropriate values of (ϵ, λ))?

One property of Lipschitz games upon which we rely heavily throughout this work is the ability to maintain *upper bounds* as λ and ϵ are scaled *up* together, and *lower bounds* as λ and ϵ are scaled *down*

together. This underlies many of the results we present. However, as the opposite is not inherently true, we are unable to generalize certain statements. Two important questions arise from this in particular:

Open Question 6. Can we always find an $\epsilon = O(n\lambda)$ such that ϵ-ANEs of n-player, binary-action, λ-Lipschitz games can be found making $O(\log n)$ queries (i.e. can we scale Theorem 2.31 down)?

Open Question 7. Can we always find a $\lambda\frac{(m-1)}{m}$-ANE of an n-player, m-action, λ-Lipschitz game in a subexponential number of queries (i.e. does the dichotomy between Theorem 6.2 and Proposition 6.1 scale down like the upper bound does in obtaining Corollary 6.3)?

Expanding from Lipschitz considerations, two additional obvious open questions spring from the results of Chapter 6:

Open Question 8. Can we develop a similarly strong dichotomy for the *randomized* query complexity of finding ϵ-ANEs of n-player, m-action games?

Of course the recent work of [54] raising ϵ_0 to nearly $\frac{1}{3}$ represents significant progress on this question.

Open Question 9. Can we develop similarly strong lower bounds on the deterministic query complexity of finding ϵ-CCEs of n-player, m-action games?

[53] make progress towards this, but there still appears to remain much room for improvement.

7.2 Final Thoughts

The above work has implications of both a theoretical and applied nature. Theoretically, we have advanced the study of Lipschitz games, providing both new upper bounds and lower bounds to problems of previous interest.

Practically, Lipschitz games, and games that can be approximated by Lipschitz or Multi-Lipschitz games, arise in many real-world situations. We have made significant progress towards showing the regimes in which these games may be practical to solve and, perhaps more importantly, the scenarios in which they are not.

Bibliography

[1] Althöfer, I.: On sparse approximations to randomized strategies and convex combinations. Linear Algebra and its Applications **199**, 339 – 355 (1994). https://doi.org/https://doi.org/10.1016/0024-3795(94)90357-3, http://www.sciencedirect.com/science/article/pii/0024379594903573, special Issue Honoring Ingram Olkin

[2] Azrieli, Y., Shmaya, E.: Lipschitz games. Math. Oper. Res. **38**(2), 350–357 (2013). https://doi.org/10.1287/moor.1120.0557, https://doi.org/10.1287/moor.1120.0557

[3] Babichenko, Y.: Best-reply dynamics in large binary-choice anonymous games. Games Econ. Behav. **81**, 130–144 (2013). https://doi.org/10.1016/j.geb.2013.04.007, https://doi.org/10.1016/j.geb.2013.04.007

[4] Babichenko, Y.: Small support equilibria in large games. CoRR **abs/1305.2432** (2013), https://arxiv.org/abs/1305.2432

[5] Babichenko, Y.: Query complexity of approximate nash equilibria. J. ACM **63**(4), 36:1–36:24 (2016). https://doi.org/10.1145/2908734, https://doi.org/10.1145/2908734

[6] Babichenko, Y.: Fast convergence of best-reply dynamics in aggregative games. Math. Oper. Res. **43**(1), 333–346 (2018). https://doi.org/10.1287/moor.2017.0868

[7] Babichenko, Y.: Informational bounds on equilibria (a survey). SIGecom Exch. **17**(2), 25–45 (2019). https://doi.org/10.1145/3381329.3381333, https://doi.org/10.1145/3381329.3381333

[8] Babichenko, Y., Barman, S., Peretz, R.: Simple approximate equilibria in large games. In: Babaioff, M., Conitzer, V., Easley, D.A. (eds.) ACM Conference on Economics and

Computation, EC '14, Stanford , CA, USA, June 8-12, 2014. pp. 753–770. ACM (2014). https://doi.org/10.1145/2600057.2602873, https://doi.org/10.1145/2600057.2602873

[9] Bertsekas, D.P., Tsitsiklis, J.N.: Introduction to Probability Vol. 1. Athena Scientific (2002)

[10] Bollobás, B.: Combinatorics: set systems, hypergraphs, families of vectors, and combinatorial probability. Cambridge University Press (1986)

[11] Buresh-Oppenheim, J., Morioka, T.: Relativized NP search problems and propositional proof systems. In: 19th Annual IEEE Conference on Computational Complexity (CCC 2004), 21-24 June 2004, Amherst, MA, USA. pp. 54–67. IEEE Computer Society (2004). https://doi.org/10.1109/CCC.2004.1313795, https://doi.org/10.1109/CCC.2004.1313795

[12] Buss, S.R., Johnson, A.S.: Propositional proofs and reductions between NP search problems. Ann. Pure Appl. Log. **163**(9), 1163–1182 (2012)

[13] Cai, Y., Candogan, O., Daskalakis, C., Papadimitriou, C.H.: Zero-sum polymatrix games: A generalization of minmax. Math. Oper. Res. **41**(2), 648–655 (2016). https://doi.org/10.1287/moor.2015.0745, https://doi.org/10.1287/moor.2015.0745

[14] Carmona, G.: Purification of bayesian–nash equilibria in large games with compact type and action spaces. Journal of Mathematical Economics **44**(12), 1302–1311 (2008)

[15] Carmona, G., Podczeck, K.: On the existence of pure-strategy equilibria in large games. Journal of Economic Theory **144**(3), 1300–1319 (2009). https://doi.org/https://doi.org/10.1016/j.jet.2008.11.009, https://www.sciencedirect.com/science/article/pii/S0022053108001695

[16] Chen, B., Ankenman, J.: The Mathematics of Poker. ConJelCo LLC (2006), https://books.google.com/books?id=bgNGPQAACAAJ

[17] Chen, X., Cheng, Y., Tang, B.: Well-supported vs. approximate nash equilibria: Query complexity of large games. In: Papadimitriou, C.H. (ed.) 8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA. LIPIcs, vol. 67, pp. 57:1–57:9. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). https://doi.org/10.4230/LIPIcs.ITCS.2017.57, https://doi.org/10.4230/LIPIcs.ITCS.2017.57

[18] Chen, X., Deng, X.: 3-nash is ppad-complete. Electronic Colloquium on Computational Complexity (ECCC) **134** (2005), http://eccc.hpi-web.de/eccc-reports/2005/TR05-134/index.html

[19] Chen, X., Deng, X.: Settling the complexity of two-player nash equilibrium. In: 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings. pp. 261–272. IEEE Computer Society (2006). https://doi.org/10.1109/FOCS.2006.69, https://doi.org/10.1109/FOCS.2006.69

[20] Chen, X., Deng, X., Teng, S.: Computing nash equilibria: Approximation and smoothed complexity. In: 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings. pp. 603–612. IEEE Computer Society (2006). https://doi.org/10.1109/FOCS.2006.20, https://doi.org/10.1109/FOCS.2006.20

[21] Chen, X., Deng, X., Teng, S.: Settling the complexity of computing two-player Nash equilibria. J. ACM **56**(3), 14:1–14:57 (2009). https://doi.org/10.1145/1516512.1516516

[22] Conitzer, V., Sandholm, T.: Complexity results about nash equilibria. In: Gottlob, G., Walsh, T. (eds.) IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003. pp. 765–771. Morgan Kaufmann (2003), http://ijcai.org/Proceedings/03/Papers/111.pdf

[23] Conitzer, V., Sandholm, T.: Communication complexity as a lower bound for learning in games. In: Brodley, C.E. (ed.) Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004. ACM International Conference Proceeding Series, vol. 69. ACM (2004). https://doi.org/10.1145/1015330.1015351, https://doi.org/10.1145/1015330.1015351

[24] Daskalakis, C., Goldberg, P.W., Papadimitriou, C.H.: The complexity of computing a nash equilibrium. Commun. ACM **52**(2), 89–97 (2009). https://doi.org/10.1145/1461928.1461951, https://doi.org/10.1145/1461928.1461951

[25] Daskalakis, C., Goldberg, P.W., Papadimitriou, C.H.: The complexity of computing a Nash equilibrium. Commun. ACM **52**(2), 89–97 (2009). https://doi.org/10.1145/1461928.1461951

[26] Daskalakis, C., Mehta, A., Papadimitriou, C.H.: A note on approximate nash equilibria. Theor. Comput. Sci. **410**(17), 1581–1588 (2009). https://doi.org/10.1016/j.tcs.2008.12.031, https://doi.org/10.1016/j.tcs.2008.12.031

[27] Daskalakis, C., Papadimitriou, C.H.: Continuous local search. In: Randall, D. (ed.) Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011. pp. 790–804. SIAM (2011). https://doi.org/10.1137/1.9781611973082.62, https://doi.org/10.1137/1.9781611973082.62

[28] Daskalakis, C., Papadimitriou, C.H.: Approximate nash equilibria in anonymous games. J. Econ. Theory **156**, 207–245 (2015). https://doi.org/10.1016/j.jet.2014.02.002, https://doi.org/10.1016/j.jet.2014.02.002

[29] Daskalakis, K., Papadimitriou, C.H.: Three-player games are hard. Electronic Colloquium on Computational Complexity (ECCC) **139** (2005), http://eccc.hpi-web.de/eccc-reports/2005/TR05-139/index.html

[30] Deb, J., Kalai, E.: Stability in large bayesian games with heterogeneous players. J. Econ. Theory **157**, 1041–1055 (2015). https://doi.org/10.1016/j.jet.2015.02.001, https://doi.org/10.1016/j.jet.2015.02.001

[31] Deligkas, A., Fearnley, J., Savani, R., Spirakis, P.G.: Computing approximate nash equilibria in polymatrix games. Algorithmica **77**(2), 487–514 (2017). https://doi.org/10.1007/s00453-015-0078-7, https://doi.org/10.1007/s00453-015-0078-7

[32] Erdem, A., Pelillo, M.: Graph transduction as a noncooperative game. Neural Comput. **24**(3), 700–723 (2012). https://doi.org/10.1162/NECO_a_00233, https://doi.org/10.1162/NECO_a_00233

[33] Etessami, K., Yannakakis, M.: On the complexity of nash equilibria and other fixed points. SIAM J. Comput. **39**(6), 2531–2597 (2010). https://doi.org/10.1137/080720826, https://doi.org/10.1137/080720826

[34] Fabrikant, A., Papadimitriou, C.H., Talwar, K.: The complexity of pure nash equilibria. In: Babai, L. (ed.) Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004. pp. 604–612. ACM (2004). https://doi.org/10.1145/1007352.1007445, https://doi.org/10.1145/1007352.1007445

[35] Fearnley, J., Gairing, M., Goldberg, P.W., Savani, R.: Learning equilibria of games via payoff queries. J. Mach. Learn. Res. **16**, 1305–1344 (2015), http://dl.acm.org/citation.cfm?id=2886792

[36] Fearnley, J., Goldberg, P.W., Hollender, A., Savani, R.: The complexity of gradient descent: CLS = PPAD ∩ PLS. In: Khuller, S., Williams, V.V. (eds.) STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021. pp. 46–59. ACM (2021). https://doi.org/10.1145/3406325.3451052, https://doi.org/10.1145/3406325.3451052

[37] Fearnley, J., Savani, R.: Finding approximate nash equilibria of bimatrix games via payoff queries. ACM Trans. Economics and Comput. **4**(4), 25:1–25:19 (2016). https://doi.org/10.1145/2956579, https://doi.org/10.1145/2956579

[38] Feder, T., Nazerzadeh, H., Saberi, A.: Approximating nash equilibria using small-support strategies. In: MacKie-Mason, J.K., Parkes, D.C., Resnick, P. (eds.) Proceedings 8th ACM Conference on Electronic Commerce (EC-2007), San Diego, California, USA, June 11-15, 2007. pp. 352–354. ACM (2007). https://doi.org/10.1145/1250910.1250961, https://doi.org/10.1145/1250910.1250961

[39] Filos-Ratsikas, A., Giannakopoulos, Y., Hollender, A., Lazos, P., Poças, D.: On the complexity of equilibrium computation in first-price auctions. In: EC '21: The 22nd ACM Conference on Economics and Computation. pp. 454–476. ACM (2021)

[40] Goldberg, P.W., Cossío, F.J.M.: Learning convex partitions and computing game-theoretic equilibria from best response queries. In: Christodoulou, G., Harks, T. (eds.) Web and Internet Economics - 14th International Conference, WINE 2018, Oxford, UK, December 15-17, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11316, pp. 168–187. Springer (2018). https://doi.org/10.1007/978-3-030-04612-5_12, https://doi.org/10.1007/978-3-030-04612-5_12

[41] Goldberg, P.W., Cossío, F.J.M., Wu, Z.S.: Logarithmic query complexity for approximate nash computation in large games. Theory Comput. Syst. **63**(1), 26–53 (2019). https://doi.org/10.1007/s00224-018-9851-8, https://doi.org/10.1007/s00224-018-9851-8

[42] Goldberg, P.W., Katzman, M.J.: Lower bounds for the query complexity of equilibria in lipschitz games. In: Caragiannis, I., Hansen, K.A. (eds.) Algorithmic Game Theory - 14th International Symposium, SAGT 2021, Aarhus, Denmark, September 21-24, 2021, Proceedings. Lecture Notes

in Computer Science, vol. 12885, pp. 124–139. Springer (2021). https://doi.org/10.1007/978-3-030-85947-3_9, https://doi.org/10.1007/978-3-030-85947-3_9

[43] Goldberg, P.W., Katzman, M.J.: Ppad-complete pure approximate nash equilibria in lipschitz games. In: Kanellopoulos, P., Kyropoulou, M., Voudouris, A.A. (eds.) Algorithmic Game Theory - 15th International Symposium, SAGT 2022, Colchester, UK, September 12-15, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13584, pp. 169–186. Springer (2022). https://doi.org/10.1007/978-3-031-15714-1_10, https://doi.org/10.1007/978-3-031-15714-1_10

[44] Goldberg, P.W., Papadimitriou, C.H.: Reducibility among equilibrium problems. In: Kleinberg, J.M. (ed.) Proceedings of the 38th Annual ACM Symposium on Theory of Computing. pp. 61–70. ACM (2006). https://doi.org/10.1145/1132516.1132526

[45] Goldberg, P.W., Roth, A.: Bounds for the query complexity of approximate equilibria. ACM Trans. Economics and Comput. **4**(4), 24:1–24:25 (2016). https://doi.org/10.1145/2956582, https://doi.org/10.1145/2956582

[46] Goldberg, P.W., Turchetta, S.: Query complexity of approximate equilibria in anonymous games. J. Comput. Syst. Sci. **90**, 80–98 (2017). https://doi.org/10.1016/j.jcss.2017.07.002, https://doi.org/10.1016/j.jcss.2017.07.002

[47] Goldwasser, S., Grossman, O., Holden, D.: Pseudo-deterministic proofs. In: Karlin, A.R. (ed.) 9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA. LIPIcs, vol. 94, pp. 17:1–17:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). https://doi.org/10.4230/LIPIcs.ITCS.2018.17, https://doi.org/10.4230/LIPIcs.ITCS.2018.17

[48] Govindan, S., Wilson, R.: Computing nash equilibria by iterated polymatrix approximation. Journal of Economic Dynamics and Control **28**(7), 1229–1241 (2004). https://doi.org/https://doi.org/10.1016/S0165-1889(03)00108-8, https://www.sciencedirect.com/science/article/pii/S0165188903001088

[49] Gradwohl, R., Reingold, O.: Partial exposure in large games. Games and Economic Behavior **68**(2), 602–613 (2010). https://doi.org/https://doi.org/10.1016/j.geb.2009.09.006, https://www.sciencedirect.com/science/article/pii/S0899825609001894

[50] Gradwohl, R., Reingold, O.: Fault tolerance in large games. Games Econ. Behav. **86**, 438–457 (2014). https://doi.org/10.1016/j.geb.2013.06.007, https://doi.org/10.1016/j.geb.2013.06.007

[51] Graham, F.C.: Math 262 lecture notes: Edge isoperimetric inequalities. https://mathweb.ucsd.edu/ fan/teach/262/notes/paul/11_7_notes.pdf (2005), transcribed by Ross M. Richardson

[52] Hart, S., Mansour, Y.: How long to equilibrium? the communication complexity of uncoupled equilibrium procedures. Games Econ. Behav. **69**(1), 107–126 (2010). https://doi.org/10.1016/j.geb.2007.12.002, https://doi.org/10.1016/j.geb.2007.12.002

[53] Hart, S., Nisan, N.: The query complexity of correlated equilibria. Games Econ. Behav. **108**, 401–410 (2018). https://doi.org/10.1016/j.geb.2016.11.003

[54] Hollender, A., Deligkas, A., Fearnley, J., Melissourgos, T.: Total search problems in game theory and economics (Sep 2022)

[55] Howson, J.T.: Equilibria of polymatrix games. Management Science **18**(5), 312–318 (1972), http://www.jstor.org/stable/2634798

[56] Janovskaja, E.: Equilibrium points in polymatrix games. Lithuanian Mathematical Journal **8**, 381–384 (April 1968). https://doi.org/10.15388/LMJ.1968.20224

[57] Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: How easy is local search? J. Comput. Syst. Sci. **37**(1), 79–100 (1988). https://doi.org/10.1016/0022-0000(88)90046-3, https://doi.org/10.1016/0022-0000(88)90046-3

[58] Kalai, E.: Large robust games. Econometrica **72**(6), 1631–1665 (2004)

[59] Kopalle, P., Shumsky, R.A.: Game theory models of pricing. Handbook of Pricing Management (2010)

[60] Lipton, R.J., Markakis, E., Mehta, A.: Playing large games using simple strategies. In: Menascé, D.A., Nisan, N. (eds.) Proceedings 4th ACM Conference on Electronic Commerce (EC-2003), San Diego, California, USA, June 9-12, 2003. pp. 36–41. ACM (2003). https://doi.org/10.1145/779928.779933, https://doi.org/10.1145/779928.779933

[61] Megiddo, N., Papadimitriou, C.H.: On total functions, existence theorems and computational complexity. Theor. Comput. Sci. **81**(2), 317–324 (1991). https://doi.org/10.1016/0304-3975(91)90200-L, https://doi.org/10.1016/0304-3975(91)90200-L

[62] Miller, D.A., Zucker, S.W.: Copositive-plus lemke algorithm solves polymatrix games. Oper. Res. Lett. **10**(5), 285–290 (1991). https://doi.org/10.1016/0167-6377(91)90015-H, https://doi.org/10.1016/0167-6377(91)90015-H

[63] Morioka, T.: Classification of search problems and their definability in bounded arithmetic. Electron. Colloquium Comput. Complex. **082** (2001), http://eccc.hpi-web.de/eccc-reports/2001/TR01-082/index.html

[64] Nash, J.: Non-Cooperative Games. Ph.D. thesis, Princeton University (May 1950)

[65] Nash, J.: Non-cooperative games. Annals of mathematics pp. 286–295 (1951)

[66] Nisan, N., Roughgarden, T., Tardos, É., Vazirani, V.V. (eds.): Algorithmic Game Theory. Cambridge University Press (2007). https://doi.org/10.1017/CBO9780511800481, https://doi.org/10.1017/CBO9780511800481

[67] Papadimitriou, C.H.: On the complexity of the parity argument and other inefficient proofs of existence. J. Comput. Syst. Sci. **48**(3), 498–532 (1994). https://doi.org/10.1016/S0022-0000(05)80063-7, https://doi.org/10.1016/S0022-0000(05)80063-7

[68] Papadimitriou, C.H., Peng, B.: Public goods games in directed networks. In: EC '21: The 22nd ACM Conference on Economics and Computation. pp. 745–762. ACM (2021)

[69] Peretz, R., Schreiber, A., Schulte-Geers, E.: The Lipschitz constant of perturbed anonymous games. CoRR **abs/2004.14741** (2020), https://arxiv.org/abs/2004.14741

[70] Phillips, M., Lorenz, T.: 'dumb money' is on gamestop, and it's beating wall street at its own game. https://www.nytimes.com/2021/01/27/business/gamestop-wall-street-bets.html (Feb 2021), accessed: 2021-02-05

[71] Rosenthal, R.W.: A class of games possessing pure-strategy nash equilibria. International Journal of Game Theory **2**(1), 65–67 (1973)

[72] Rubinstein, A.: Settling the complexity of computing approximate two-player nash equilibria. In: Dinur, I. (ed.) IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA. pp. 258–265. IEEE Computer Society (2016). https://doi.org/10.1109/FOCS.2016.35, https://doi.org/10.1109/FOCS.2016.35

[73] Rubinstein, A.: Inapproximability of Nash equilibrium. SIAM J. Comput. **47**(3), 917–959 (2018). https://doi.org/10.1137/15M1039274

[74] Sandler, T., Arce, D.G.: Terrorism: A game-theoretic approach. Handbook of defense economics **2**, 775–813 (2007)

[75] Schmeidler, D.: Equilibrium points of nonatomic games. Journal of statistical Physics **7**(4), 295–300 (1973)

[76] Schoenebeck, G., Vadhan, S.P.: The computational complexity of Nash equilibria in concisely represented games. ACM Trans. Comput. Theory **4**(2), 4:1–4:50 (2012). https://doi.org/10.1145/2189778.2189779

[77] Shapiro, C.: The theory of business strategy. The Rand journal of economics **20**(1), 125–137 (1989)

[78] Smith, J.M.: Evolution and the Theory of Games. Cambridge University Press (1982). https://doi.org/10.1017/CBO9780511806292

[79] Wellman, M.P.: Game playing meets game theory: Strategic learning from simulated play (2020), university of Oxford, Department of Computer Science Lecture