Efficient Algorithms for Computing Approximate Equilibria in Bimatrix, Polymatrix and Lipschitz Games

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy
by

Argyrios Deligkas

July 2016

Abstract

In this thesis, we study the problem of computing approximate equilibria in several classes of games. In particular, we study approximate Nash equilibria and approximate well-supported Nash equilibria in polymatrix and bimatrix games and approximate equilibria in Lipschitz games, penalty games and biased games. We construct algorithms for computing approximate equilibria that beat the current best algorithms for these problems.

In Chapter 3, we present a distributed method to compute approximate Nash equilibria in bimatrix games. In contrast to previous approaches that analyze the two payoff matrices at the same time (for example, by solving a single LP that combines the two players' payoffs), our algorithm first solves two independent LPs, each of which is derived from one of the two payoff matrices, and then computes an approximate Nash equilibrium using only limited communication between the players.

In Chapter 4, we present an algorithm that, for every δ in the range $0 < \delta \leq 0.5$, finds a $(0.5+\delta)$-Nash equilibrium of a polymatrix game in time polynomial in the input size and $\frac{1}{\delta}$. Note that our approximation guarantee *does not depend on the number of players*, a property that was not previously known to be achievable for polymatrix games, and still cannot be achieved for general strategic-form games.

In Chapter 5, we present an approximation-preserving reduction from the problem of computing an approximate Bayesian Nash equilibrium (ϵ-BNE) for a two-player Bayesian game to the problem of computing an ϵ-NE of a polymatrix game and thus show that the algorithm of Chapter 4 can be applied to two-player Bayesian games. Furthermore, we provide a simple polynomial-time algorithm for computing a 0.5-BNE.

In Chapter 5, we study games with non-linear utility functions for the players. Our key insight is that Lipschitz continuity of the utility function allows us to provide algorithms for finding approximate equilibria in these games. We begin by studying Lipschitz games, which encompass, for example, all concave games

with Lipschitz continuous payoff functions. We provide an efficient algorithm for computing approximate equilibria in these games. Then we turn our attention to penalty games, which encompass biased games and games in which players take risk into account. Here we show that if the penalty function is Lipschitz continuous, then we can provide a quasi-polynomial time approximation scheme. Finally, we study distance biased games, where we present simple strongly polynomial time algorithms for finding best responses in L_1, L_2^2, and L_∞ biased games, and then use these algorithms to provide strongly polynomial algorithms that find 2/3, 5/7, and 2/3 approximations for these norms, respectively.

Contents

Acknowledgments

First, and foremost, I am indebted to my supervisor, Rahul Savani. He was a mentor, a teacher, a friend. Rahul helped me to overcome all the difficulties I faced during the four years I spent in Liverpool. The meetings with Rahul were far away from the typical supervisor-student meetings. They were starting with an equilibrium problem to solve, going to trading, music, politics, betting, and ending to a solution to the original problem. And this was fun.

I am also indebted to Paul Spirakis and John Fearnley. The collaboration with them made me a better researcher and a better person. Alongside with Rahul, they formed a team such that at any point of the day, there was someone to help me and answer my questions. The inputs of each member of the team individually and of the team as a whole helped me to create my academic identity.

I want to thank my second supervisor Piotr Krysta and my advisors Giorgos Christodoulou and Martin Gairing. They never failed to provide valuable advice and guidance during my time here. I would also like to thank Martin and Carmine Ventre for acting as my examiners and helping me to significantly improve the presentation of my thesis.

I feel indebted to all my collaborators: Rahul, Paul, John, Artur Czumaj, Michail Fasoulakis, Marcin Jurdziski, George Mertzios, Tobenna Peter Igwe, Eleftherios Anastasiadis, Mingyu Guo. Mingyu deserves an honorable mention, since he was my initial supervisor in Liverpool and he was the person that gave me the opportunity to come to Liverpool in the first place.

I also want to thank my friends that were by me during all these years: Nikos, Thanasis, Katerina, Katerina, Anastasia, Lefteris, Eleni, Alkmini, Dimitris, Yiannis, Themistoklis.

Last but not least, I am grateful to my family Giorgos, Angeliki and Kostas for all their love and encouragement they provided me.

Chapter 1

Introduction

In this thesis, we study the problem of computing approximate equilibria in several classes of games. In particular, we study approximate Nash equilibria and approximate well supported Nash equilibria in polymatrix and bimatrix games and approximate equilibria in Lipschitz games, penalty games and biased games. We construct algorithms for computing approximate equilibria that beat the current best algorithms that tackle these problems. In this chapter, we give an overview of the problems we are considering, and the results that are obtained in this thesis.

1.1 Game Theory

Game Theory is one of the most important mathematical fields established in the 20-th century. The seminal book, the *Theory of Games and Economic Behavior*, of John von Neumann and Oskar Morgenstern [69] is considered the text that established the modern Game Theory. Since then, many books have been written on Game Theory or use Game Theory as a tool. As Roger Myerson states in his book [59]

> *"Game theory has a very general scope, encompassing questions that are basic to all of the social sciences. It can offer insights into any economic, political, or social situation that involves individuals who have different goals or preferences."*

As the quote above states, Game Theory studies mathematical models where two or more individuals, or players as they are called in the game theoretic language, interact with each other. The details and the rules of the interactions between the players are called *games*. A game can be described by the players that participate, the set of available actions each player can choose from, known

as *strategies*, and the *utility* every player gets from every possible outcome of the game. The utility a player gets can be given implicitly by functions, or explicitly by matrices if this is possible.

For example, Figure 1.1 demonstrates probably the most famous game; the *Prisoners' Dilemma* Game. In the Prisoners' Dilemma Game there are two players, player I and player II, accused of committing a crime. Each player has two choices:

1. denying the commitment of the crime, or "Quiet",
2. confessing that the other player committed the crime, or "Fink".

If both players deny the commitment of the crime, i.e. choose "Quiet", then each one of them will stay in prison for a year. If only one player chooses "Quiet" while the other player "Fink", then the player who choose "Quiet" will be sentenced for five years whereas the other player will be set free. Finally, if both players choose "Fink", then each one of them will stay in prison for three years. The utilities of the players for this game can be given by two 2×2 matrices or by one table, Figure 1.1, that combines them. The interpretation of Figure 1.1 is the following. Each cell corresponds to a possible outcome of the game and the numbers correspond to the utility of the players for this outcome. The number in the bottom-left corner of the cell corresponds to the utility of the player I, usually called as the *row* player, while the number in the top-right corner corresponds to the utility of the player II, usually called the *column* player.

In the game theoretic language the two-player games, that can be fully described using two matrices, are known as *bimatrix* games.

I \ II	Quiet	Fink
Quiet	-1 -1	0 -5
Fink	-5 0	-3 -3

Figure 1.1: The Prisoners' Dilemma Game.

So, how can we study the Prisoners' Dilemma Game? The objective is to study the behavior of the players and argue about the outcome from their interaction. The major assumption in game theory is that the players are *rational*.

This means that each player chooses the strategy that maximizes his utility given the chosen action of his opponent. In game theoretic terms the players are *utility maximizers*. Thus, if player I chooses Quiet, then player II should choose Fink. This is because given that player I chooses Quiet, player II gets utility -1 by choosing Quiet, whereas if he chooses Fink he gets utility 0. If the player I chooses Fink, then player II gets utility -5 if he chooses Quiet and utility -3 if he chooses Fink. Hence, under the assumption that both players are rational, the player II will choose Fink. By symmetry, for the player I we can conclude that the only *stable* pair of choices is when both players choose Fink and get utility -3 each. Neither player can increase their utility by changing strategy. We call this pair of actions a pure *Nash equilibrium* of the game [1]. In a pure equilibrium the strategy for every player involves only one action from the actions available to him. Intuitively, a collection of strategies is a pure Nash equilibrium for a game if no player can increase their utility by unilaterally deviating from his chosen strategy, given that the rest of the players do not change their strategies. Notice though that in the Prisoners' Dilemma Game the players could *cooperate* and both choose Quiet and get utility -1, but this violates the assumption of rationality for the players.

The notion of pure Nash equilibrium is easy to understand and in bimatrix games is easy to find. Note although that in other more complicated games like hedonic games is hard to find [35]. However, pure Nash equilibria do not exist in every game. Consider for example the *Penalty Shot* game described in Figure 1.2 played between a goalkeeper, player $\mathcal{G}$ and a kicker, player $\mathcal{K}$.

$\mathcal{G}$ \ $\mathcal{K}$	L	R
L	1 -1	-1 1
R	-1 1	1 -1

Figure 1.2: The Penalty Shot Game.

The numerical values correspond to the following rules: if the goalkeeper and the penalty kicker choose the same side (Left or Right) then the goalkeeper wins

[1] Actually, this pair of actions is a dominant strategy profile, since every player has to choose the specified action irrespectively from the action his opponent chooses

one point and the penalty kicker loses one; if they choose different strategies, then the goalkeeper loses a point and the penalty kicker wins one point. Note that for each pair of pure strategies for the players the sum of their utilities adds up to zero. These games are called *zero sum games*.

It is easy to see that there is no pure Nash equilibrium in the Penalty Shot Game. So, what should be the outcome of a game that does not possess a pure Nash equilibrium? Then, the players can randomize by selecting a probability distribution over the set of their strategies and play *mixed strategies*. Suppose that each player chooses Left with probability $\frac{1}{2}$ and Right with probability $\frac{1}{2}$, i.e. the players play uniformly at random their pure strategies, and communicate that to their opponent. Then, no player can increase their *expected utility* by switching to a different strategy (mixed or pure). The pair of uniform strategies of the players is called *mixed Nash equilibrium*, or simply *Nash equilibrium*. Formally, the Nash equilibrium is defined as a collection of mixed strategies, one for every player of the game, such that none of the players can improve their expected utility by unilaterally changing their strategy.

1.2 Existence of Equilibria

Although in the Prisoners' Dilemma Game and in the Penalty Shot Game it is easy to decide whether a Nash equilibrium exists, for larger games it was not clear whether a Nash equilibrium exists or not.

John von Neumann [68] extended the work of Emile Borel [6] and showed that any bimatrix zero sum game possesses at least one mixed Nash equilibrium. Later it was understood that the existence of a Nash equilibrium in zero sum games is equivalent to Linear Programming duality [17] and thus it is computationally easy to find a Nash equilibrium in such games using the ellipsoid algorithm of Khachiyan [51], or the interior point method of Karmakar [50].

John Nash [60] in his seminal paper showed that *every* game, with finite number of players and finite number of strategies available to each player, possesses a mixed Nash equilibrium, irrespective from the structure of the utilities for the players. These games are known as *strategic form* games.

Rosen in his seminal work [62] considered a more general setting of games, with respect to the utilities of the players and the strategies available to each player, called *concave games*. There, the available actions for each player correspond to vectors from a convex set, thus each player can have infinite number of

strategies. The payoff of each player is specified by a function that satisfies the following condition: if every other player's strategy is fixed, then the payoff to a player is a concave function over his strategy space. Rosen proved that concave games always possess an equilibrium. A natural subclass of concave games, studied by Caragiannis, Kurokawa and Procaccia [10], is the class of biased games. A biased game is defined by a strategic form game, a *base strategy* and a *penalty function*. The players play the strategic form game as normal, but they all suffer a penalty for deviating from their base strategy. This penalty can be a non-linear function, such as the L_2^2 norm.

1.3 Computation of Nash Equilibria

Although the existence of equilibria was understood, there were no efficient algorithms for finding one. Nash's proof was based on the Brouwer's fixed point theorem and it was non constructive. This means that Nash's proof did not provide or imply an algorithm that computes a Nash equilibrium. Since then, the quest of an efficient algorithm for computing Nash equilibria has started. Lemke [55] gave an algorithm that computes a Nash equilibrium in bimatrix games. This algorithm was fast in practice but its complexity remained unknown until 2004 when Savani and von Stengel [65] showed that there exist games in which the algorithm needs exponential time to compute a Nash equilibrium.

In 1994 Papadimitriou [61] defined the complexity class `PPAD` in order to capture the complexity of the equilibrium computation problem. This class captures problems that can be reduced to the end of line problem:

- Given a succinctly represented directed graph consisting of vertices with indegree and outderee at most one and a vertex of outdegree one, find a vertex with zero outdegree.

Ten years later, a line of work of Daskalakis, Goldberg and Papadimitriou [18], and Chen, Deng and Teng [12] showed that computing an exact Nash equilibrium is `PPAD`-complete even for bimatrix games, and so there are unlikely to be polynomial time algorithms for this problem. The hardness of computing exact Nash equilibria has lead to the study of *approximate* Nash equilibria: while an exact equilibrium requires that all players have no incentive to deviate from their current strategy, an ϵ-approximate Nash equilibrium, simply ϵ-NE, requires only

that their incentive to deviate is less than ϵ, i.e. no player can increase their payoff more than ϵ by changing their strategy. However, in order the approximation guarantee to have a consistent meaning over all games it is usually assumed that the utilities of the players are in $[0, 1]$, so $\epsilon \in [0, 1]$.

1.3.1 Bimatrix games

A fruitful line of work has developed studying the best approximations that can be found in polynomial-time for *bimatrix games*. The most known and most studied notion is that of approximate Nash equilibrium. There, after a number of papers by Daskalakis, Mehta and Papadimitriou [19, 20], Bosse, Byrka and Markakis [7], the best known algorithm was given by Spirakis and Tsaknakis [66] who provided a polynomial time algorithm, known as `TS` algorithm, that finds a 0.3393-NE. Although, it is not known whether the guarantee of `TS` algorithm is tight, there are examples upon which the algorithm finds no better than a 0.3385-NE [31].

A different notion of approximation is the *ϵ-well-supported Nash equilibrium*. An *ϵ-well-supported Nash equilibrium* (ϵ-WSNE) is a pair of strategies in which both players only place probability on strategies whose payoff is within ϵ of the maximum payoff. Every ϵ-WSNE is an ϵ-NE, but the converse does not hold. So, ϵ-WSNE is a more restrictive notion. Although the computation of ϵ-NE received a lot of attention the progress on computing ϵ-WSNE has been less forthcoming. The first correct algorithm was provided by Kontogiannis and Spirakis [52] (KS algorithm), who gave a polynomial time algorithm for finding a $\frac{2}{3}$-WSNE. This was later slightly improved by Fearnley, Goldberg, Savani, and Sørensen [30] who gave a new polynomial-time algorithm that extends the KS algorithm and finds a 0.6608-WSNE; prior to this work, this was the best known approximation guarantee for WSNEs. For the special case of symmetric games, i.e. bimatrix games where the payoff matrix or the column player equals the transposed payoff matrix of the row player, there is a polynomial-time algorithm for finding a 0.5-WSNE [16].

The existence of a fully polynomial approximation scheme (FPTAS) was ruled out by Chen, Deng and Teng [12] unless `PPAD` = `P`. Recently, Rubinstein proved that there is no polynomial approximation scheme (PTAS) assuming the End of Line problem requires exponential time [64]. There is however a *quasi-polynomial* approximation scheme given by Lipton, Markakis and Mehta [56].

1.3.2 Many-player games

The study of ϵ-Nash equilibria in the context of *many-player* games has received much less attention. A simple approximation algorithm for many-player games can be obtained by generalising the algorithm of Daskalakis, Mehta and Papadimitriou [20] from the two-player setting to the n-player setting, which provides a guarantee of $\epsilon = 1 - \frac{1}{n}$. This has since been improved independently by three sets of authors [7, 8, 45]. They provide a method that converts a polynomial-time algorithm for finding ϵ-Nash equilibria in $(n-1)$-player games into an algorithm that finds a $\frac{1}{2-\epsilon}$-Nash equilibrium in n-player games. Using the polynomial-time 0.3393 algorithm of Tsaknakis and Spirakis [66] for 2-player games as the base case for this recursion, this allows us to provide polynomial-time algorithms with approximation guarantees of 0.6022 in 3-player games, and 0.7153 in 4-player games. These guarantees tend to 1 as n increases, and so far, no constant $\epsilon < 1$ is known such that, for all n, an ϵ-Nash equilibrium of an n-player game can be computed in polynomial time.

For n-player games, we have lower bounds for ϵ-Nash equilibria. More precisely, Rubinstein has shown that when the number of players is not constant there exists a constant but very small ϵ such that it is PPAD-hard to compute an ϵ-Nash equilibrium [63]. This is quite different from the bimatrix game setting, where the existence of a quasi-polynomial time approximation scheme rules out such a lower bound, unless all of PPAD can be solved in quasi-polynomial time [57]. On the other hand for any number of players with *constant* number of pure strategies per player and every $\epsilon > 0$, Babichenko, Barman and Peretz [4] showed that an ϵ-Nash equilibrium can be computed in quasi polynomial time.

1.3.3 Polymatrix games

Polymatrix games form a special class of many player games. In a polymatrix game, the interaction between the players is specified by an n-vertex graph, where each vertex represents one of the players. Each edge of the graph specifies a bimatrix game that will be played by the two respective players, and thus a player with degree d will play d bimatrix games simultaneously. More precisely, each player picks a strategy, and then plays this strategy in *all* of the bimatrix games that he is involved in. His payoff is then the sum of the payoffs that he obtains in each of the games.

Polymatrix games form a class of *succinctly represented* n-player games: a polymatrix game is specified by at most n^2 bimatrix games, each of which can be written down in quadratic space with respect to the number of strategies. This is

unlike general n-player strategic form games, which require a representation that is exponential in the number of players.

The problem of computing exact Nash equilibria in polymatrix games can be tackled in exponential time by Lemke's algorithm [46]. For the special subclass of generalized zero sum games on networks Cai and Daskalakis [9] showed that a Nash equilibrium can be computed in polynomial time. On the other hand, there has been relatively little work on polynomial time algorithms for computing approximate Nash equilibria in polymatrix games. The approximation algorithms for general games can be applied in this setting in an obvious way, but prior to this work there have been no upper bounds that are specific to polymatrix games. On the other hand, the lower bound of Rubinstein mentioned above is actually proved by constructing polymatrix games. Thus, there is a constant but very small ϵ such that it is PPAD-hard to compute an ϵ-Nash equilibrium [63], and this again indicates that approximating equilibria in polymatrix games is quite different to approximating equilibria in bimatrix games.

Polymatrix games have played a central role in the reductions that have been used to show PPAD-hardness of games and other equilibrium problems [12, 13, 18, 27, 33]. Computing an *exact* Nash equilibrium in a polymatrix game is PPAD-hard even when all the bimatrix games played are either zero-sum games or coordination games [9].

Polymatrix games have been used in other contexts too. For example, Govindan and Wilson proposed a (non-polynomial-time) algorithm for computing Nash equilibria of an n-player game by approximating the game with a sequence of polymatrix games [40]. Later, they presented a (non-polynomial) reduction that reduces n-player games to polymatrix games while preserving approximate Nash equilibria [41]. Their reduction introduces a central coordinator player, who interacts bilaterally with every player.

1.3.4 Lipschitz games

The results for games that are not in strategic form are even scarcer. An exception is the class of games that the utility functions for the players are Lipschitz continuous. Intuitively, in a Lipschitz game a small change in a player's strategy does not change significantly his or his opponents' payoff. Several papers have studied how the Lipschitz continuity of the players' payoff functions affects the existence, the quality, and the complexity of the equilibria of the underlying game. Azriely and Shmaya [2] studied many player games and derived bounds for the Lipschitz constant of the utility functions for the players that guaran-

tees the existence of a pure approximate equilibrium for the game. Daskalakis and Papadimitriou [21] proved that anonymous games possess pure approximate equilibria whose quality depends on the Lipschitz constant of the payoff functions and the number of pure strategies the players have. They also proved that this approximate equilibrium can be computed in polynomial time. Furthermore, they gave a polynomial-time approximation scheme for anonymous games with many players and constant number of pure strategies. Babichenko [3] presented a best-reply dynamic for n players Lipschitz anonymous games with two strategies which reaches an approximate pure equilibrium in $O(n \log n)$ steps. Deb and Kalai [22] studied how some variants of the Lipschitz continuity of the utility functions are sufficient to guarantee hindsight stability of equilibria.

1.3.5 Communication and Query Complexity of equilibria

Communication Complexity. Approximate Nash equilibria can also be studied from the *communication complexity* point of view, which captures the amount of communication the players need to find a good approximate Nash equilibrium. It models a natural scenario where the two players know their own payoff matrix, but do not know their opponent's payoff matrix. The players must then follow a communication protocol that eventually produces strategies for both players. The goal is to design a protocol that produces a sufficiently good ϵ-NE or ϵ-WSNE while also minimizing the amount of communication between the two players.

Communication complexity of equilibria in games has been studied in previous works [14, 44]. The recent paper of Goldberg and Pastink [38] initiated the study of communication complexity in the bimatrix game setting. There they showed $\Theta(n^2)$ communication, of the payoff matrices, is required to find an exact Nash equilibrium of an $n \times n$ bimatrix game. Since these games have $\Theta(n^2)$ payoffs in total, this implies that there is no communication efficient protocol for finding exact Nash equilibria in bimatrix games. For approximate equilibria, they showed that one can find a $\frac{3}{4}$-Nash equilibrium *without any communication*, and that in the no-communication setting, finding an $\frac{1}{2}$-Nash equilibrium is impossible. Motivated by these positive and negative results, they focused on the most interesting setting, which allows only a polylogarithmic in the number of pure strategies to be exchanged between the players. They demonstrated that one can compute 0.438-NE and 0.732-WSNE in this setting.

Query Complexity. The payoff query model is motivated by practical applications of game theory. It is often the case that we know that there is a game

to be solved, but we do not know what the payoffs are, and in order to discover the payoffs, we would have to play the game. This may be quite costly, so it is natural to ask whether we can find an equilibrium of a game while minimizing the number of experiments that we must perform.

Payoff queries model this situation. In the payoff query model we are told the structure of the game, i.e., the strategy space, but we are not told the payoffs. We can then make payoff queries, where we propose a pure strategy profile, and we are told the payoff of each player under that strategy profile. Our task is to compute an equilibrium of the game while minimizing the number of payoff queries that we make.

The study of query complexity in bimatrix games was initiated by Fearnley, Gairing, Goldberg and Savani [29], who gave a deterministic algorithm for finding a $\frac{1}{2}$-NE using $2n-1$ payoff queries. A subsequent paper of Fearnley and Savani [32] showed a number of further results. Firstly, they showed an $\Omega(n^2)$ lower bound on the query complexity of finding an ϵ-NE with $\epsilon < \frac{1}{2}$, which combined with the result above, gives a complete view of the deterministic query complexity of approximate Nash equilibria in bimatrix games. They then gave a randomized algorithm that finds a $(\frac{3-\sqrt{5}}{2} + \epsilon)$-NE using $O(\frac{n \cdot \log n}{\epsilon^2})$ queries, and a randomized algorithm that finds a $(\frac{2}{3} + \epsilon)$-WSNE using $O(\frac{n \cdot \log n}{\epsilon^4})$ queries.

1.4 Overview of Results

The contributions of this thesis are *efficient algorithms for computing approximate equilibria* for several classes of games.

1.4.1 Bimatrix games

We introduce a *distributed* technique that allows us to efficiently compute approximate Nash equilibria and approximate well-supported Nash equilibria using limited communication between the players.

Traditional methods for computing WSNEs have used an LP based approach that, when used on a bimatrix game (R, C), solves the zero-sum game $(R-C, C-R)$, where R and C denote the payoff matrix for the row and column player respectively. The Kontogiannis and Spirakis algorithm [53] (KS algorithm) uses the fact that if there is no pure $\frac{2}{3}$-WSNE, then the solution to this zero-sum game is a $\frac{2}{3}$-WSNE. The slight improvement of the algorithm of Fearnley, Goldberg, Savani and Sørensen [30] (FGSS algorithm) to 0.6608 was obtained by adding two further methods to the KS algorithm: if the KS algorithm does not produce

a 0.6608-WSNE, then either there is a 2×2 *matching pennies* sub-game that is 0.6608-WSNE for the bimatrix game (R, C), or the strategies computed from the zero-sum game can be improved by shifting the probabilities of both players within their supports in order to produce a 0.6608-WSNE.

We take a different approach. We first show that the bound of $\frac{2}{3}$ can be matched using a pair of *distributed* LPs. Given a bimatrix game (R, C), we solve the two zero-sum games $(R, -R)$ and $(-C, C)$, and then give a straightforward procedure that we call the *base algorithm*, which uses the solutions to these games to produce a $\frac{2}{3}$-WSNE of (R, C). Goldberg and Pastnik [38] also considered this pair of LPs, but their algorithm only produces a 0.732-WSNE. We then show that the base algorithm can be improved by applying the probability-shifting and matching-pennies ideas from the FGSS-algorithm. That is, if the base algorithm fails to find a 0.6528-WSNE, then a 0.6528-WSNE can be obtained either by shifting the probabilities of one of the two players, or by identifying a 2×2 sub-game where its exact NE corresponds to a 0.6528-WSNE. This gives a polynomial time algorithm that computes a 0.6528-WSNE, which provides the best known approximate guarantees for WSNEs.

It is worth pointing out that, while these techniques are thematically similar to the ones used by the FGSS-algorithm, the actual implementation is significantly different. The FGSS-algorithm attempts to improve the strategies by shifting probabilities *within the supports* of the strategies returned by the zero-sum game, with the goal of reducing the other player's payoff. In our algorithm, we shift probabilities *away from bad strategies* in order to improve that player's payoff. This type of analysis is possible because the base algorithm produces a strategy profile in which one of the two players plays a pure strategy, which makes the analysis we need to carry out much simpler. On the other hand, the KS-algorithm can produce strategies in which both players play many strategies, and so the analysis used for the FGSS-algorithm is necessarily more complicated.

Since our algorithm solves the two LPs separately, it can be used to improve upon the best known algorithms in the limited communication setting. This is because no communication is required for the row player to solve $(R, -R)$ and the column player to solve $(-C, C)$. The players can then carry out the rest of the algorithm using only poly-logarithmic communication. Hence, we obtain a randomized expected-polynomial-time algorithm that uses poly-logarithmic communication and finds a 0.6528-WSNE. Moreover, the base algorithm can be implemented as a communication efficient algorithm for finding a $(0.5 + \epsilon)$-WSNE in a *win-lose* bimatrix game, where all payoffs are either 0 or 1.

The algorithm can also be used to beat the best known bound in the query complexity setting. It has already been shown by Goldberg and Roth [39] that an ϵ-NE of a *zero-sum game* can be found by a randomized algorithm that uses $O(\frac{n \log n}{\epsilon^2})$ payoff queries. Since the rest of the steps used by our algorithm can also be carried out using $O(n \log n)$ payoff queries, this gives us a query efficient algorithm for finding a 0.6528-WSNE.

We also show that the base algorithm can be adapted to find a $\frac{3-\sqrt{5}}{2}$-NE in a bimatrix game, which matches the bound given for the first algorithm of Bosse et al. [7]. Once again, this can be implemented in a communication efficient manner, and so we obtain an algorithm that computes a $(\frac{3-\sqrt{5}}{2} + \epsilon)$-NE (i.e., 0.382-NE) using only poly-logarithmic communication.

1.4.2 Polymatrix games

Our main result is an algorithm that, for every δ in the range $0 < \delta \leq 0.5$, finds a $(0.5 + \delta)$-NE of a polymatrix game in time polynomial in the input size and $\frac{1}{\delta}$, assuming every player has maximum payoff 1 and minimum payoff 0. Note that our approximation guarantee *does not depend on the number of players,* which is a property that was not previously known to be achievable for polymatrix games, for any constant $\epsilon < 1$, and still cannot be achieved for general strategic form games.

We prove this result by adapting the algorithm of Tsaknakis and Spirakis [66], (TS algorithm). They give a gradient descent algorithm for finding a 0.3393-Nash equilibrium in a bimatrix game. We generalise their gradient descent techniques to the polymatrix setting, and show that it always arrives at a $(0.5 + \delta)$-Nash equilibrium after a polynomial number of iterations.

In order to generalise the TS algorithm we had to overcome several issues. Firstly, the TS algorithm makes the regrets of the two players equal in every iteration, but there is no obvious way to achieve this in the polymatrix setting. Instead, we show how gradient descent can be applied to a strategy profile where the regrets are not necessarily equal. Secondly, the output of the TS algorithm is either a point found by gradient descent, or a point obtained by modifying the result of gradient descent. In the polymatrix game setting, it is not immediately obvious how such a modification can be derived with a non-constant number of players (without an exponential blowup). Thus, we apply a different analysis, which proves that the point resulting from gradient descent always has our approximation guarantee. It is an interesting open question whether a better approximation guarantee can be achieved when there is a constant number of

players.

Furthermore, we show that our algorithm can be applied to two-player Bayesian games. Rosenthal and Howson showed that the problem of finding an exact equilibrium in a two-player Bayesian game is equivalent to finding an exact equilibrium in a polymatrix game [48]. We show that this correspondence also holds for approximate equilibria: finding an ϵ-Nash equilibrium in these games can be reduced to the problem of finding an ϵ-Nash equilibrium in a polymatrix game, and therefore, our algorithm can be used to efficiently find a $(0.5+\delta)$-Nash equilibrium of a two-player Bayesian game.

1.4.3 Lipschitz games

We study three classes of games that are not strategic form; *Lipschitz games, Penalty games* and *Distance Biased games*.

Lipschitz games. This is a very general class of games, where each player's strategy space is continuous, is represented by a convex set of vectors, and where the only restriction is that the payoff function is Lipschitz continuous. This class encompasses, for example, every concave game in which the payoffs are Lipschitz continuous. This class is so general that exact equilibria, and even approximate equilibria may not exist. Nevertheless, we give an efficient algorithm that either outputs an ϵ-equilibrium, or determines that the game has no exact equilibria. More precisely, for M-player games that are λ-continuous in the L_p norm, for $p \geq 2$, and where $\gamma = \max \|\mathbf{x}\|_p$ over all $\mathbf{x}$ in the strategy space, we either compute an ϵ-equilibrium or determine that no exact equilibrium exists in time $O\left(Mn^{Mk+l}\right)$, where $k = O\left(\frac{\lambda^2 M p \gamma^2}{\epsilon^2}\right)$ and $l = O\left(\frac{\lambda^2 p \gamma^2}{\epsilon^2}\right)$. Observe that this is a polynomial time algorithm when λ, p, γ, M, and ϵ are constant.

To prove this result, we utilize a recent result of Barman [5], which states that for every vector in a convex set, there is another vector that is ϵ close to the original in the L_p norm, and is a convex combination of b points on the convex hull, where b depends on p and ϵ, but does not depend on the dimension. This result and the Lipschitz continuity of the payoff functions allow us to reduce the task of finding an ϵ-equilibrium to checking only a small number of strategy profiles, and thus we get a brute-force algorithm that is reminiscent of the QPTAS given by Lipton, Mehta and Markakis [56] for bimatrix games.

However, life is not so simple for us. Since we study a very general class of games, verifying whether a given strategy profile is an ϵ-equilibrium is a non-trivial task. It requires us to compute a *regret* for each player, which is the differ-

ence between the player's best response payoff and their actual payoff. Computing a best response in a bimatrix game is trivial, but for Lipschitz games, computing a best response may be a hard problem. We get around this problem by instead giving an algorithm to compute *approximate* best responses. Hence we find *approximate* regrets, and it turns out that this is sufficient for our algorithm to work.

Penalty games. In these games, the players play a strategic form game, and their utility is the payoff achieved in the game *minus* a penalty. The penalty function can be an arbitrary function that depends on the player's strategy. This is a general class of games that encompasses a number of games that have been studied before. The biased games studied in [10] are penalty games where the penalty is determined by the amount each player deviates from a specified base strategy. The biased model was studied in the past by psychologists [67] and it is close to what they call *anchoring* [49, 11]. Anchoring is common in poker [2] and in fact there are several papers on poker that are reminiscent of anchoring [36, 37, 47]. In their seminal paper, Fiat and Papadimitriou [34] introduced a model for *risk prone* games. This model resembles penalty games since the risk component can be encoded in the penalty function. Mavronicolas and Monien [58] followed this line of research and provided results on the complexity of deciding if such games possess an equilibrium.

We again show that Lipschitz continuity helps us to find approximate equilibria. The only assumption that we make is that the penalty function is Lipschitz continuous in an L_p norm with $p \geq 2$. Again, this is a weak restriction and it does not guarantee that exact equilibria exist. Even so, we give a quasi-polynomial time algorithm that either finds an ϵ-equilibrium, or verifies that the game has no exact equilibrium.

We take a similar approach, but since our games are more complicated, our proof is necessarily more involved. In particular, in [56], proving that the sampled strategies are an approximate equilibrium only requires showing that the expected payoff is close the payoff of a pure best response. In penalty games, best response strategies are not necessarily pure, and so the events that we must consider are more complex.

Distance biased games. Biased games, are a subclass of penalty games that have been studied recently by Caragiannis, Kurokawa and Procaccia [10]. They

[2] http://www.pokerology.com/articles/anchoring-bias/

showed that, under very mild assumptions on the bias function, biased games always have an exact equilibrium. Furthermore, for the case where the bias function is either the L_1 norm, or the L_2^2 norm, they give an exponential time algorithm for finding an exact equilibrium.

Our results for penalty games already give a QPTAS for biased games, but we are also interested in whether there are polynomial-time algorithms that can find non-trivial approximations. We give a positive answer to this question for games where the bias is the L_1 norm, the L_2^2 norm, or the L_∞ norm. We follow the well-known approach of Daskalakis, Mehta and Papadimitriou [20], who gave a simple algorithm for finding a 0.5-approximate equilibrium in a bimatrix game. Their approach is as follows: start with an arbitrary strategy $\mathbf{x}$ for player 1, compute a best response j for player 2 against $\mathbf{x}$, and then compute a best response i for player 1 against j. Player 1 mixes uniformly between $\mathbf{x}$ and i, while player 2 plays j.

We show that this algorithm also works for biased games, although the generalisation is not entirely trivial. Again, this is because best responses cannot be trivially computed in biased games. For the L_1 and L_∞ norms, best responses can be computed via linear programming, and for the L_2^2 norm, best responses can be formulated as a quadratic program. It turns out that this particular QP can be solved in polynomial time by the ellipsoid method. However, none of these algorithms is strongly polynomial. We show that, for each of the norms, best responses can be found by a simple strongly-polynomial combinatorial algorithm. We then analyse the quality of approximation provided by the technique of [20]. We obtain a strongly polynomial algorithm for finding a $2/3$ approximation in L_1 and L_∞ biased games, and a strongly polynomial algorithm for finding a $5/7$ approximation in L_2^2 biased games. For the latter result, in the special case where the bias function is the inner product of the player's strategy we find a $13/21$ approximation.

Papers in the thesis

This thesis is primarily based on three papers published during my PhD studies.

Chapter 3 is based on the paper "Distributed Methods for Computing Approximate Equilibria" [15] published in WINE 2016: The 12-th Conference on Web and Internet Economics.

Chapters 4 and 5 are based on the paper " Computing approximate Nash Equilibria in Polymatrix Games" [24] published in WINE 2014 and accepted for publication in Algorithmica. The algorithm presented in Section 5.3 was derived by Michail Fasoulakis who kindly agreed to add it in my thesis.

Chapter 6 is based on the paper "Lipschitz Continuity and Approximate Equilibria" [25] published in SAGT 2016: The 9th International Symposium on Algorithmic Game Theory.

Furthermore, during my Phd studies I was a co-author in the following papers:

- "An Empirical Study on Computing Equilibria in Polymatrix Games" [23] published in AAMAS 2016.
- "Inapproximability Results for Approximate Nash Equilibria" [28] published in WINE 2016.
- "Increasing VCG Revenue by Decreasing the Quality of Items" [43] published in AAAI 2014.
- "Revenue Maximization via Hiding Item Attributes" [42] published in IJCAI 2013.
- "On the Complexity of Weighted Greedy Matchings" [26].
- "Minmax Heterogeneous Facility Location Games" [1].

Chapter 2

Preliminaries

In this Chapter we introduce the necessary notation that will be used throughout this thesis.

2.1 Games, Strategies and Utility Functions

We start by fixing some notation. For each positive integer n we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$, and when a universe $[n]$ is clear, we will use $\bar{S} = \{i \in [n], i \notin S\}$ to denote the complement of $S \subseteq [n]$. For an n-dimensional vector x, we use x_{-S} to denote the elements of x with indices in $\bar{S}$, and in the case where $S = \{i\}$ has only one element, we simply write x_{-i} for x_{-S}. We use Δ_n to denote the $(n-1)$-dimensional simplex, formally $\Delta_n := \{x : x \in \mathbb{R}^n, x \geq 0, \sum_{i=1}^{n} x_i = 1\}$. Furthermore, we use $\|x\|_p$ to denote the p-norm of a vector $x \in \mathbb{R}^d$, i.e. $\|x\|_p = \left(\sum_{i\in[d]} |x_i|^p\right)^{1/p}$. Given a set $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$, we use $conv(X)$ to denote the convex hull of X. A vector $y \in conv(X)$ is said to be k-uniform with respect to X if there exists a size k multiset S of $[n]$ such that $y = \frac{1}{k}\sum_{i\in S} x_i$. When X is clear from the context we will simply say that a vector is k uniform without mentioning that uniformity is with respect to X.

Games and strategies. A game with n players can be described by a set of available actions for each player and a utility function for each player that depends both on his chosen action and the actions the rest of the players chose.

Definition 1 (Game)**.** *An n-player game Γ is defined by*

- *a set of players $\mathcal{N} = [n]$,*
- *a set of available actions $\mathcal{S}_i$ for every player $i \in [n]$,*
- *a utility function $u_i : \mathcal{S}_1 \times \ldots \times \mathcal{S}_n \to \mathbb{R}$ for every player $i \in [n]$.*

Strategies. For each player $i \in [n]$ we will call $\mathcal{S}_i$ as *strategy space*. We will use $x_i \in \mathcal{S}_i$ to denote a specific action chosen by player i and we will call it as the *strategy* of player i. In order to play the game, all players simultaneously select a strategy from their strategy set and we use $\mathbb{X} = (x_1, \ldots, x_n)$ to denote a *strategy profile* of the game. When player $i \in [n]$ chooses an action deterministically from his strategy space, we will say that player i plays a pure strategy. If the player randomises over some actions according to a probability distribution, we will say that he plays a *mixed strategy*. We use $u_i(\mathbb{X}) = u_i(x_i, \mathbb{X}_{-i})$ to denote the utility of player i when he plays the strategy x_i and the rest of the players play according to the strategy profile $\mathbb{X}_{-i}$.

Best responses. A strategy $\hat{x}_i$ is a *best response* against the strategy profile $\mathbb{X}_{-i}$, if $u_i(\hat{x}_i, \mathbb{X}_{-i}) \geq u_i(x_i, \mathbb{X}_{-i})$ for all $x_i \in S_i$. The *regret* player i suffers under a strategy profile $\mathbb{X}$ is the difference between the utility of his best response and his utility under $\mathbb{X}$, i.e. $u_i(\hat{x}_i, \mathbb{X}_{-i}) - u_i(x_i, \mathbb{X}_{-i})$.

The Definition 1 above is quite general. If extra constraints are imposed on the number of the players and/or on the utility functions, then several classes of games can be constructed. In this thesis the following classes of games are studied: bimatrix (or two player) games, polymatrix games, Bayesian two player games, Lipschitz games, concave games, penalty games, biased games, and distance biased games. In each of the following chapters we will define explicitly and study the aforementioned classes games.

2.2 Solution Concepts

The standard solution concept in game theory is the *equilibrium*. A strategy profile is an equilibrium if no player can increase his utility by unilaterally changing his strategy.

Definition 2 (Equilibrium)**.** *The strategy profile $\mathbb{X}$ is an equilibrium for an n-player game Γ if and only if for all $i \in [n]$ it holds that $u_i(x_i, \mathbb{X}_{-i}) \geq u_i(x_i', \mathbb{X}_{-i})$ for all possible $x_i' \in \Delta_{\mathcal{S}_i}$.*

When we refer to bimatrix, polymatrix, or Bayesian two player games we will call the equilibria as Nash equilibria, while for the rest of the classes we will call them simply as equilibria. We make this distinction because Nash's theorem [60] does not hold for the classes of Lipschitz, concave, penalty, biased, and distance biased games.

2.2.1 Approximate Equilibria

In this thesis we study a "relaxed" notion of equilibria which we call *approximate* equilibria. More specifically, we study *additive* approximate equilibria. For any $\epsilon > 0$, an additive ϵ-approximate equilibrium, or simply ϵ-equilibrium, is a strategy profile where no player can increase his payoff more than ϵ by unilaterally changing his strategy.

Definition 3 (ϵ-equilibrium)**.** *Let $\epsilon > 0$. The strategy profile $\mathbb{X}$ is an ϵ-equilibrium for an n-player game if and only if for all players $i \in [n]$ it holds that $u_i(x_i, \mathbb{X}_{-i}) \geq u_i(x'_i, \mathbb{X}_{-i}) - \epsilon$ for all possible $x'_i \in \Delta_{\mathcal{S}_i}$.*

If under the strategy profile $\mathbb{X}$ a player can increase his payoff by ϵ by unilaterally changing his strategy x_i, then we say that the player suffers ϵ regret.

2.2.2 Payoff rescaling

Observe that in order an ϵ-equilibrium to be a meaningful solution concept for a game, it must have the same meaning for every player. Consider for example the scenario of a strategy profile for a two player game where no player can increase his payoff more than one. Assume furthermore that under this strategy profile the first player gets utility zero, while the second player gets utility of a million. The incentive for the first player to change his strategy is much stronger than the incentive of the second player.

In order to overcome this inefficiency of the definition of ϵ-equilibrium we must make some assumptions on the utility functions. The standard assumption in the field of approximate equilibria is that the utility for every player is normalised in $[0, 1]$, thus the value of ϵ has the same effect in every player. This normalisation does not affect the exact equilibria of the game and if it is applied carefully it does not affect the approximate equilibria of the game either. In the following chapters, if we do not mention how to normalise the utilities of the players we implicitly assume that are already normalised in $[0, 1]$. In Chapter 4 and in Chapter 5 we explain in detail how to normalise the utilities in polymatrix and Bayesian two player games respectively.

2.3 Game Sizes

The size of a game is defined by the number of bits required in order to represent it. In order to define the size of a game it is sufficient just to describe the utility functions of the players that are involved in it. Notice that for an n-player game

if $|\mathcal{S}_i| = s$ for all $i \in [n]$, then there are s^n different strategy profiles for the game. This means that ns^n payoffs have to be described in order to represent the game. This is exponential in the number of players. In this thesis we focus only on games with *succint representation*, i.e. the size of the game is polynomial in the number of the players of the game.

Bimatrix games are succinctly represented games; when each player has s pure strategies, then the size of the game is $2s^2$. Polymatrix games form a class of succinctly represented n-player games: a polymatrix game is specified by at most n^2 bimatrix games, each of which can be written down in quadratic space with respect to the number of strategies and Bayesian two-player games have a succinct representation too, since such a game can be reduced to a polymatrix game (we will explain these in detail in Chapters 4 and 5).

Observe though that we implicitly make an assumption on the payoffs representation. We assume that each payoff needs at most polylogarithmic bits to be described.

In games with concave utility functions the size of the game depends on how the utility functions for the players are represented. Throughout this thesis we will assume that the utility functions for all players have a succinct representation.

Chapter 3

Bimatrix Games

In this chapter we present a distributed method to compute approximate Nash equilibria in bimatrix games. In contrast to previous approaches that analyze the two payoff matrices at the same time (for example, by solving a single LP that combines the two players' payoffs), our algorithm first solves two independent LPs, each of which is derived from one of the two payoff matrices, and then computes an approximate Nash equilibrium using only limited communication between the players.

Our method gives improved bounds on the complexity of computing approximate Nash equilibria in a number of different settings. Firstly, it gives a polynomial-time algorithm for computing *approximate well supported Nash equilibria (WSNE)* that always finds a 0.6528-WSNE, beating the previous best guarantee of 0.6608. Secondly, since our algorithm solves the two LPs separately, it can be applied to give an improved bound in the limited communication setting, giving a randomized expected-polynomial-time algorithm that uses poly-logarithmic communication and finds a 0.6528-WSNE which beats the previous best known guarantee of 0.732. It can also be applied to the case of *approximate Nash equilibria*, where we obtain a randomized expected-polynomial-time algorithm that uses poly-logarithmic communication and always finds a 0.382-approximate Nash equilibrium, which improves the previous best guarantee of 0.438. Finally, the method can also be applied in the query complexity setting to give an algorithm that makes $O(n \log n)$ payoff queries and always finds a 0.6528-WSNE, which improves the previous best known guarantee of 2/3.

3.1 Bimatrix games preliminaries

An $n \times m$ bimatrix game is a pair (R, C) of two $n \times m$ matrices: R defines the payoff for the *row* player, and C defines the payoff for the *column* player. Without loss of generality and for notation simplicity, we will assume that $m = n$. We note however that our algorithm works even if $m \neq n$. We make the standard assumption that all payoffs lie in the range $[0, 1]$. So, each player has n *pure* strategies. To play the game, the row player selects a row $i \in [n]$, and the column player selects a column $j \in [n]$. The row player then receives payoff $R_{i,j}$, and the column player receives payoff $C_{i,j}$.

A *mixed strategy* is a probability distribution over $[n]$. We denote a mixed strategy for the row player as a vector $\mathbf{x}$ of length n, such that $\mathbf{x}_i$ is the probability that the row player assigns to pure strategy i. A mixed strategy of the column player is a vector $\mathbf{y}$ of length n, with the same interpretation. Given a mixed strategy $\mathbf{x}$ for either player, the *support* of $\mathbf{x}$ is the set of pure strategies i with $\mathbf{x}_i > 0$. If $\mathbf{x}$ and $\mathbf{y}$ are mixed strategies for the row and the column player, respectively, then we call $(\mathbf{x}, \mathbf{y})$ a *mixed strategy profile*. The expected payoff for the row player under the strategy profile $(\mathbf{x}, \mathbf{y})$ is given by $\mathbf{x}^T R \mathbf{y}$ and for the column player by $\mathbf{x}^T C \mathbf{y}$. We denote the *support* of a strategy $\mathbf{x}$ as $\text{supp}(\mathbf{x})$, which gives the set of pure strategies i such that $\mathbf{x}_i > 0$.

Nash equilibria. Let $\mathbf{y}$ be a mixed strategy for the column player. The set of *pure best responses* against $\mathbf{y}$ for the row player is the set of pure strategies that maximize the payoff against $\mathbf{y}$. More formally, a pure strategy $i \in [n]$ is a best response against $\mathbf{y}$ if, for all pure strategies $i' \in [n]$ we have: $\sum_{j \in [n]} \mathbf{y}_j \cdot R_{i,j} \geq \sum_{j \in [n]} \mathbf{y}_j \cdot R_{i',j}$. Column player best responses are defined analogously.

A mixed strategy profile $(\mathbf{x}, \mathbf{y})$ is a *mixed Nash equilibrium* if every pure strategy in $\text{supp}(\mathbf{x})$ is a best response against $\mathbf{y}$, and every pure strategy in $\text{supp}(\mathbf{y})$ is a best response against $\mathbf{x}$. Observe that in a Nash equilibrium, each player's expected payoff is equal to their best response payoff.

Approximate Equilibria. There are two commonly studied notions of approximate equilibrium, and we consider both of them in this chapter. The first notion is of an *ϵ-approximate Nash equilibrium* (ϵ-NE), which weakens the requirement that a player's expected payoff should be equal to their best response payoff. Formally, given a strategy profile $(\mathbf{x}, \mathbf{y})$, we define the *regret* suffered by the row player to be the difference between the best response payoff and the

actual payoff:

$$\max_{i \in [n]} \big((R \cdot y)_i \big) - \mathbf{x}^T \cdot R \cdot \mathbf{y}.$$

Regret for the column player is defined analogously. We have that $(\mathbf{x}, \mathbf{y})$ is an ϵ-NE if and only if both players have regret less than or equal to ϵ.

The other notion is of an ϵ-approximate-well-supported equilibrium (ϵ-WSNE), which weakens the requirement that players only place probability on best response strategies. Given a strategy profile $(\mathbf{x}, \mathbf{y})$ and a pure strategy $j \in [n]$, we say that j is an ϵ-best-response for the row player if:

$$\max_{i \in [n]} \big((R \cdot y)_i \big) - (R \cdot y)_j \leq \epsilon.$$

An ϵ-WSNE requires that both players only place probability on ϵ-best-responses. Formally, the row player's *pure strategy regret* under $(\mathbf{x}, \mathbf{y})$ is defined to be:

$$\max_{i \in [n]} \big((R \cdot y)_i \big) - \min_{i \in \text{supp}(\mathbf{x})} \big((R \cdot y)_i \big).$$

Pure strategy regret for the column player is defined analogously. A strategy profile $(\mathbf{x}, \mathbf{y})$ is an ϵ-WSNE if both players have pure strategy regret less than or equal to ϵ.

Communication complexity. We consider the communication model for bimatrix games introduced by Goldberg and Pastink [38]. In this model, both players know the payoffs in their own payoff matrix, but do not know the payoffs in their opponent's matrix. The players then follow an algorithm that uses a number of communication rounds, where in each round they exchange a single bit of information. Between each communication round, the players are permitted to perform arbitrary randomized computations (although it should be noted that, in this chapter, the players will only perform polynomial-time computations) using their payoff matrix, and the bits that they have received so far. At the end of the algorithm, the row player outputs a mixed strategy $\mathbf{x}$, and the column player outputs a mixed strategy $\mathbf{y}$.

The goal is to produce a strategy profile $(\mathbf{x}, \mathbf{y})$ that is an ϵ-NE or ϵ-WSNE for a sufficiently small ϵ while limiting the number of communication rounds used by the algorithm. The algorithms given in this chapter will use at most $O(\log^2 n)$ communication rounds.

Query complexity. In the query complexity setting, the algorithm knows that the players will play an $n \times n$ game (R, C), but it does not know any of the entries of R or C. These payoffs are obtained using *payoff queries* in which the algorithm proposes a pure strategy profile (i, j), and then it is told the value of R_{ij} and C_{ij}. After each payoff query, the algorithm can make arbitrary computations (although, again, in this chapter the algorithms that we consider take polynomial time) in order to decide the next pure strategy profile to query. After making a sequence of payoff queries, the algorithm then outputs a mixed strategy profile $(\mathbf{x}, \mathbf{y})$. Again, the goal is to ensure that this strategy profile is an ϵ-NE or ϵ-WSNE, while minimizing the number of queries made overall.

3.2 An Algorithm for Finding a $\frac{2}{3}$-WSNE

In this section, we introduce an algorithm that we call the *base algorithm*. This algorithm provides a simple way to find a $\frac{2}{3}$-WSNE. We present this algorithm separately for three reasons.

- We believe that the algorithm is interesting on its own right, since it provides a relatively straightforward method for finding a $\frac{2}{3}$-WSNE that is quite different from the technique used in the KS-algorithm.
- Our algorithm for finding a 0.6528-WSNE will replace the final step of the algorithm with two more involved procedures, so it is worth understanding this algorithm before we describe how it can be improved.
- We will show that this algorithm can be adapted to provide a communication efficient way to find a $(0.5 + \epsilon)$-WSNE in win-lose games.

The algorithm. Our algorithm solves two zero-sum games. We say that the strategy $\mathbf{x}$ secures value v for the row player, if $\mathbf{x}^T R\mathbf{y} \geq v$ for every possible $\mathbf{y}$.

Algorithm 1: The base algorithm

1. Solve the zero-sum games $(R, -R)$ and $(-C, C)$.
   - Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a NE of $(R, -R)$, and let $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ be a NE of $(-C, C)$.
   - Let v_r be the value secured by $\mathbf{x}^*$ in $(R, -R)$, and let v_c be the value secured by $\hat{\mathbf{y}}$ in $(-C, C)$. Without loss of generality assume that $v_c \leq v_r$.
2. If $v_r \leq 2/3$, then return $(\hat{\mathbf{x}}, \mathbf{y}^*)$.
3. If for all $j \in [n]$ it holds that $C_j^T \cdot \mathbf{x}^* \leq 2/3$, then return $(\mathbf{x}^*, \mathbf{y}^*)$.
4. Otherwise:
   - Let $\mathbf{j}^*$ be a pure best response to $\mathbf{x}^*$.
   - Find a row i such that $R_{i\mathbf{j}^*} > 1/3$ and $C_{i\mathbf{j}^*} > 1/3$.
   - Return $(i, \mathbf{j}^*)$.

We argue that this algorithm is correct. For that reason, we must prove that the row i used in Step 4 actually exists, which we do in the following lemma.

Lemma 1. *If Algorithm 1 reaches Step 4, then there exists a row i such that $R_{i\mathbf{j}^*} > 1/3$ and $C_{i\mathbf{j}^*} > 1/3$.*

Proof. Let i be a row sampled from $\mathbf{x}^*$. We will show that there is a positive probability that row i satisfies the desired properties.

We begin by showing that the probability $\Pr(R_{i\mathbf{j}^*} \leq \frac{1}{3})$ is less than 0.5. Let the random variable $T = 1 - R_{i\mathbf{j}^*}$. Since $v_r > \frac{2}{3}$, we have that $E[T] < \frac{1}{3}$. Thus, applying Markov's inequality we obtain:

$$\Pr(T \geq \frac{2}{3}) \leq \frac{E[T]}{2/3} < 0.5.$$

Since $\Pr(R_{i\mathbf{j}^*} \leq \frac{1}{3}) = \Pr(T \geq \frac{2}{3})$ we can therefore conclude that $\Pr(R_{i\mathbf{j}^*} \leq \frac{1}{3}) < 0.5$. The exact same technique can be used to prove that $\Pr(C_{i\mathbf{j}^*} \leq \frac{1}{3}) < 0.5$, by using the fact that $C_{\mathbf{j}^*}^T \cdot \mathbf{x}^* > \frac{2}{3}$.

We can now apply the union bound to argue that:

$$\Pr(R_{i\mathbf{j}^*} \leq \frac{1}{3} \text{ or } C_{i\mathbf{j}^*} \leq \frac{1}{3}) < 1.$$

Hence, there is positive probability that row i satisfies $R_{i\mathbf{j}^*} > \frac{1}{3}$ and $C_{i\mathbf{j}^*} > \frac{1}{3}$, so such a row must exist. □

We now argue that the algorithm always produces a $\frac{2}{3}$-WSNE. There are three possible strategy profiles that can be returned by the algorithm, which we consider individually.

The algorithm returns in Step 2. Since $v_c \leq v_r$ by assumption, and since $v_r \leq \frac{2}{3}$, we have that $(R \cdot \mathbf{y}^*)_i \leq \frac{2}{3}$ for every row i, and $((\hat{\mathbf{x}})^T \cdot C)_j \leq \frac{2}{3}$ for every column j. So, both players can have pure strategy regret at most $\frac{2}{3}$ in $(\hat{\mathbf{x}}, \mathbf{y}^*)$, and thus this profile is a $\frac{2}{3}$-WSNE.

The algorithm returns in Step 3. Much like in the previous case, when the column player plays $\mathbf{y}^*$, the row player can have pure strategy regret at most $\frac{2}{3}$. The requirement that $C_j^T \mathbf{x}^* \leq \frac{2}{3}$ also ensures that the column player has pure strategy regret at most $\frac{2}{3}$. Thus, we have that $(\mathbf{x}^*, \mathbf{y}^*)$ is a $\frac{2}{3}$-WSNE.

The algorithm returns in Step 4. Both players have payoff at least $\frac{1}{3}$ under $(i, \mathbf{j}^*)$ for the sole strategy in their respective supports. Hence, the maximum pure strategy regret that can be suffered by a player is $1 - \frac{1}{3} = \frac{2}{3}$.

Observe that the zero-sum game solved in Step 1 can be solved via linear programming, and so the algorithm runs in polynomial time. Therefore, we have shown the following.

Theorem 1. *Algorithm 1 always produces a $\frac{2}{3}$-WSNE in polynomial time.*

3.3 An Algorithm for Finding a 0.6528-WSNE

In this section, we show how Algorithm 1 can be modified to produce a 0.6528-WSNE. We begin by giving an overview of the techniques used, we then give the algorithm, and finally we analyse the quality of WSNE that it produces.

Outline. The idea behind our algorithm is to replace Step 4 of Algorithm 1 with a more involved procedure. This procedure uses two techniques that both find an ϵ-WSNE with $\epsilon < \frac{2}{3}$.

Firstly, we attempt to turn $(\mathbf{x}^*, \mathbf{j}^*)$ into a WSNE by *shifting probabilities*. Observe that, since $\mathbf{j}^*$ is a best response, the column player has a pure strategy regret of 0 in $(\mathbf{x}^*, \mathbf{j}^*)$. On the other hand, we have no guarantees about the row player since $\mathbf{x}^*$ might place a small amount of probability on strategies with payoff

strictly less than $\frac{1}{3}$. However, since $\mathbf{x}^*$ achieves a high *expected* payoff, (due to Step 2) it cannot place too much probability on these low payoff strategies. Thus, the idea is to shift the probability that $\mathbf{x}^*$ assigns to entries of j^* with payoff less than or equal to $\frac{1}{3}$ to entries with payoff strictly greater than $\frac{1}{3}$, and thus ensure that the row player's pure strategy regret is below $\frac{2}{3}$. Of course, this procedure will increase the pure strategy regret of the column player, but if it is also below $\frac{2}{3}$ once all probability has been shifted, then we have found an ϵ-WSNE with $\epsilon < \frac{2}{3}$.

If shifting probabilities fails to find an ϵ-WSNE with $\epsilon < \frac{2}{3}$, then we show that the game contains a *matching pennies* sub-game. More precisely, we show that there exists a column j', and rows b and s such that the 2×2 sub-game induced by j^*, j', b, and s has the following form:

I \ II	j^*	j'
b	0 ≈ 1	≈ 1 0
s	≈ 1 0	0 ≈ 1

Thus, if both players play uniformly over their respective pair of strategies, then j^*, j', b, and s with have payoff ≈ 0.5, and so this yields an ϵ-WSNE with $\epsilon < \frac{2}{3}$.

The algorithm. We now formalize this approach, and show that it always finds an ϵ-WSNE with $\epsilon < \frac{2}{3}$. In order to quantify the precise ϵ that we obtain, we parametrise the algorithm by a variable z, which we constrain to be in $[0, \frac{1}{24})$.

With the exception of the matching pennies step, all other steps of the algorithm will return a $(\frac{2}{3} - z)$-WSNE, while the matching pennies step will return a $(\frac{1}{2} + f(z))$-WSNE for some increasing function f. Optimising the tradeoff between $\frac{2}{3} - z$ and $\frac{1}{2} + f(z)$ then allows us to determine the quality of WSNE found by our algorithm.

Algorithm 2 presents the aforementioned. Observe that Steps 1, 2, and 3 are versions of the corresponding steps from Algorithm 1 which have been adapted to produce a $(\frac{2}{3} - z)$-WSNE. Step 4 implements the probability shifting procedure, while Step 5 finds a matching pennies sub-game.

Observe that the probabilities used in $\mathbf{x_{mp}}$ and $\mathbf{y_{mp}}$ are only well defined when $z \leq \frac{1}{24}$, because we have that $\frac{1-15z}{2-39z} > 1$ whenever $z > \frac{1}{24}$, which explains our required upper bound on z.

Algorithm 2

1. Solve the zero-sum games $(R, -R)$ and $(-C, C)$.
   - Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a NE of $(R, -R)$, and let $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ be a NE of $(C, -C)$.
   - Let v_r be the value secured by $\mathbf{x}^*$ in $(R, -R)$, and let v_c be the value secured by $\hat{\mathbf{y}}$ in $(-C, C)$. Without loss of generality assume that $v_c \leq v_r$.
2. If $v_r \leq 2/3 - z$, then return $(\hat{\mathbf{x}}, \mathbf{y}^*)$.
3. If for all $j \in [n]$ it holds that $C_j^T \mathbf{x}^* \leq 2/3 - z$, then return $(\mathbf{x}^*, \mathbf{y}^*)$.
4. Otherwise:
   - Let $\mathbf{j}^*$ be a pure best response against $\mathbf{x}^*$. Define:
   $$S := \{i \in \text{supp}(\mathbf{x}^*) : R_{i\mathbf{j}^*} < 1/3 + z\}$$
   $$B := \text{supp}(\mathbf{x}^*) \setminus S$$
   - Define the strategy $\mathbf{x_b}$ as follows. For each $i \in [n]$ we have:
   $$(\mathbf{x_b})_i = \begin{cases} \frac{1}{\Pr(B)} \cdot \mathbf{x}_i^* & \text{if } i \in B \\ 0 & \text{otherwise.} \end{cases}$$
   - If $(\mathbf{x_b}^T \cdot C)_{\mathbf{j}^*} \geq \frac{1}{3} + z$, then return $(\mathbf{x_b}, \mathbf{j}^*)$.
5. Otherwise:
   - Let $\mathbf{j}'$ be a pure best response against $\mathbf{x_b}$.
   - If there exists an $i \in \text{supp}(\mathbf{x}^*)$ such that $(i, \mathbf{j}^*)$ or $(i, \mathbf{j}')$ is a pure $(\frac{2}{3} - z)$-WSNE, then return it.
   - Find a row $b \in B$ such that $R_{b\mathbf{j}^*} > 1 - \frac{18z}{1+3z}$ and $C_{b\mathbf{j}'} > 1 - \frac{18z}{1+3z}$.
   - Find a row $s \in S$ such that $C_{s\mathbf{j}^*} > 1 - \frac{27z}{1+3z}$ and $R_{s\mathbf{j}'} > 1 - \frac{27z}{1+3z}$.
   - Define the row player strategy $\mathbf{x_{mp}}$ and the column player strategy $\mathbf{y_{mp}}$ as follows. For each $i \in [n]$ we have:
   $$\mathbf{x_{mp}}_i = \begin{cases} \frac{1-24z}{2-39z} & \text{if } i = b, \\ \frac{1-15z}{2-39z} & \text{if } i = s, \\ 0 & \text{otherwise.} \end{cases} \qquad \mathbf{y_{mp}}_i = \begin{cases} \frac{1-24z}{2-39z} & \text{if } i = \mathbf{j}^*, \\ \frac{1-15z}{2-39z} & \text{if } i = \mathbf{j}', \\ 0 & \text{otherwise.} \end{cases}$$
   - Return $(\mathbf{x_{mp}}, \mathbf{y_{mp}})$.

The correctness of Step 5. This step of the algorithm relies on the existence of the rows b and s, which is not at all trivial. This is shown in the following lemma. The proof of this lemma is quite lengthy, and is given in full detail in Section 3.3.3.

Lemma 2. *Suppose that the following conditions hold:*

1. $\mathbf{x}^*$ *has payoff at least* $\frac{2}{3} - z$ *against* j^*.
2. j^* *has payoff at least* $\frac{2}{3} - z$ *against* $\mathbf{x}^*$.
3. $\mathbf{x}^*$ *has payoff at least* $\frac{2}{3} - z$ *against* j'.
4. *Neither* j^* *or* j' *contains a pure* $(\frac{2}{3} - z)$*-WSNE* (i, j) *with* $i \in \mathrm{supp}(\mathbf{x}^*)$.

Then, both of the following are true:

- *There exists a row* $b \in B$ *such that* $R_{b\mathrm{j}^*} > 1 - \frac{18z}{1+3z}$ *and* $C_{b\mathrm{j}'} > 1 - \frac{18z}{1+3z}$.
- *There exists a row* $s \in S$ *such that* $C_{s\mathrm{j}^*} > 1 - \frac{27z}{1+3z}$ *and* $R_{s\mathrm{j}'} > 1 - \frac{27z}{1+3z}$.

The lemma explicitly states the preconditions that need to hold because we will reuse it in our communication complexity and query complexity results. Observe that the preconditions are indeed true if the Algorithm reaches Step 5. The first and third conditions hold because, due to Step 2, we know that $\mathbf{x}^*$ is a minmax strategy that secures payoff at least $v_r > \frac{2}{3} - z$. The second condition holds because Step 3 ensures that the column player's best response payoff is at least $\frac{2}{3} - z$. The fourth condition holds because Step 5 explicitly checks for these pure strategy profiles.

Overview of the proof of Lemma 2. We now give an overview of the ideas used in the proof which can be found in Section 3.3.3. The majority of the proof is dedicated to proving four facts, which we outline below. First we determine the structure of the row j^*. Here we use the fact that in the strategy profile $(\mathbf{x}^*, \mathrm{j}^*)$ both players have expected payoff close to $\frac{2}{3}$, but there does not exist a row $i \in \mathrm{supp}(\mathbf{x}^*)$ such that $R_{i\mathrm{j}^*} \geq \frac{1}{3} + z$ and $C_{i\mathrm{j}^*} \geq \frac{1}{3} + z$ (because such a row would constitute a pure $(\frac{2}{3} - z)$-WSNE.) The only way this is possible is when both of the following facts hold.

1. Most of the probability assigned to B is placed on rows i with $R_{i\mathrm{j}^*} \approx 1$ and $C_{i\mathrm{j}^*} \approx \frac{1}{3}$.

2. Most of the probability assigned to S is placed on rows i with $R_{ij^*} \approx \frac{1}{3}$ and $C_{ij^*} \approx 1$.

Moreover, $\mathbf{x}^*$ must assign roughly half of its probability to rows in B and half of its probability to rows in S.

Next, we observe that since Step 4 failed to produce a $(\frac{2}{3} - z)$-WSNE, it must be the case that j^* is not a $(\frac{2}{3} - z)$-best-response against $\mathbf{x_b}$, and the payoff of j^* against $\mathbf{x_b}$ is approximately $\frac{1}{3}$, it must be the case that the payoff of j' against $\mathbf{x_b}$ is close to 1. The only way this is possible is if most column player payoffs for rows in B are close to 1. However, if this is the case, then since j^* does not contain a pure $(\frac{2}{3} - z)$-WSNE, we have that most row player payoffs in B must be below $\frac{1}{3} + z$. This gives us our third fact.

3. Most of the probability assigned to B is placed on rows i with $R_{ij'} < \frac{1}{3} + z$ and $C_{ij'} \approx 1$.

For the fourth fact, we recall that $\mathbf{x}^*$ is a min-max strategy that guarantees payoff at least $v_r > \frac{2}{3} - z$, so the payoff of $\mathbf{x}^*$ against j' must be at least $\frac{2}{3} - z$. However, since most rows $i \in B$ have $R_{ij'} < \frac{1}{3} + z$, and since $\mathbf{x}^*$ places roughly half its probability on B, it must be the case that most row player payoffs in S are close to 1. This gives us our final fact.

4. Most of the probability assigned to S is placed on rows i with $R_{ij'} \approx 1$.

Our four facts only describe the *expected* payoff of the rows in B and S for the columns j^* and j'. The final step of the proof is to pick out two particular rows that satisfy the desired properties. For the row b we use Facts 1 and 3, observing that if most of the probability assigned to B is placed on rows i with $R_{ij^*} \approx 1$, and on rows i with $C_{ij^*} \approx 1$, then it must be the case that both of these conditions can be simultaneously satisfied by a single row b. The existence of s is proved by the same argument using Facts 2 and 4.

Quality of approximation. We now analyse the quality of WSNE that our algorithm produces. Steps 2, 3, 4, 5 each return a strategy profile. Observe that Steps 2 and 3 are the same as the respective steps in the base algorithm, but with the threshold changed from $\frac{2}{3}$ to $\frac{2}{3} - z$. Hence, we can use the same reasoning as we gave for the base algorithm to argue that these steps always return $(\frac{2}{3} - z)$-WSNE. We now consider the other two steps.

The algorithm returns in Step 4. By definition all rows $r \in B$ satisfy $R_{ij^*} \geq \frac{1}{3} + z$, so since $\text{supp}(\mathbf{x_b}) \subseteq B$, the pure strategy regret of the row player can

be at most $1-(\frac{1}{3}+z)=\frac{2}{3}-z$. For the same reason, since $(\mathbf{x_b}^T \cdot C)_{\mathsf{j}^*} \geq \frac{1}{3}+z$ holds, the pure strategy regret of the column player can also be at $\frac{2}{3}-z$. Thus, the profile $(\mathbf{x_b}, \mathsf{j}^*)$ is a $(\frac{2}{3}-z)$-WSNE.

The algorithm returns in Step 5. Since $R_{b\mathsf{j}^*} > 1-\frac{18z}{1+3z}$, the payoff of b when the column player plays $\mathbf{y_{mp}}$ is at least:

$$\frac{1-24z}{2-39z} \cdot \left(1-\frac{18z}{1+3z}\right) = \frac{1-39z+360z^2}{2-33z-117z^2}$$

Similarly, since $R_{s\mathsf{j}'} > 1-\frac{27z}{1+3z}$, the payoff of s when the column player plays $\mathbf{y_{mp}}$ is at least:

$$\frac{1-15z}{2-39z} \cdot \left(1-\frac{27z}{1+3z}\right) = \frac{1-39z+360z^2}{2-33z-117z^2}$$

In the same way, one can show that the payoffs of j^* and j' are also $\frac{1-39z+360z^2}{2-33z-117z^2}$ when the row player plays $\mathbf{x_{mp}}$. Thus, we have that $(\mathbf{x_{mp}}, \mathbf{y_{mp}})$ is a $(1-\frac{1-39z+360z^2}{2-33z-117z^2})$-WSNE.

To find the optimal value for z, we need to find the largest value of z for which the following inequality holds.

$$1-\frac{1-39z+360z^2}{2-33z-117z^2} \leq \frac{2}{3}-z.$$

Setting the inequality to an equality and rearranging gives the following cubic polynomial equation.

$$117\,z^3+432\,z^2-30\,z+\frac{1}{3}=0.$$

Since the discriminant of this polynomial is positive, this polynomial has three real roots, which can be found via the trigonometric method. Only one of these roots lies in the range $0 \leq z < \frac{1}{24}$, which is the following:

$$z=\frac{1}{117}\sqrt{3}\left(\sqrt{2434}\sqrt{3}\cos\left(\frac{1}{3}\arctan\left(\frac{39}{240073}\sqrt{9749}\sqrt{3}\right)\right)\right.$$
$$\left.-3\sqrt{2434}\sin\left(\frac{1}{3}\arctan\left(\frac{39}{240073}\sqrt{9749}\sqrt{3}\right)\right)-48\sqrt{3}\right).$$

Thus, we get $z \approx 0.013906376$, and as result we have found an algorithm that always produces a 0.6528-WSNE. Hence, we have the following theorem.

Theorem 2. *There is a polynomial time algorithm that, given a bimatrix game, finds a* 0.6528*-WSNE.*

3.3.1 Communication Complexity of the Algorithm

We claim that our algorithm can be adapted for the limited communication setting. We make the following modification to our algorithm. In order to achieve this, we use the following result of [38].

Lemma 3 (Goldberg and Pastink [38]). *Given a mixed strategy* $\mathbf{x}$ *for the row player and an* $\epsilon > 0$*, there is a randomized expected-polynomial-time algorithm that uses* $O(\frac{\log^2 n}{\epsilon^2})$*-communication to transmit a strategy* $\mathbf{x}_s$ *to the column player where* $|\text{supp}(\mathbf{x}_s)| \in O(\frac{\log n}{\epsilon^2})$ *and for every strategy* $i \in [n]$ *we have:*

$$|(\mathbf{x}^T \cdot R)_i - (\mathbf{x}_s^T \cdot R)_i| \leq \epsilon.$$

The algorithm uses the well-known sampling technique of Lipton, Markakis, and Mehta [56] to construct the strategy $\mathbf{x}_s$, so for this reason we will call the strategy $\mathbf{x}_s$ the *sampled strategy* from $\mathbf{x}$. Since this strategy has a logarithmically sized support, it can be transmitted by sending $O(\frac{\log n}{\epsilon^2})$ strategy indexes, each of which can be represented using $\log n$ bits. By symmetry, the algorithm can obviously also be used to transmit approximations of column player strategies to the row player.

After Algorithm 2 computes $\mathbf{x}^*, \mathbf{y}^*, \hat{\mathbf{x}}$, and $\hat{\mathbf{y}}$, we then use Lemma 3 to construct and communicate the sampled strategies $\mathbf{x}_s^*, \mathbf{y}_s^*, \hat{\mathbf{x}}_s$, and $\hat{\mathbf{y}}_s$. These strategies are communicated between the two players using $4 \cdot (\log n)^2$ bits of communication, and the players also exchange $v_r = (\mathbf{x}_s^*)^T R \mathbf{y}_s^*$ and $v_c = \hat{\mathbf{x}}_s^T C \hat{\mathbf{y}}_s$ using $\log n$ rounds of communication. The algorithm then continues as before, except the sampled strategies that are now used in place of their non-sampled counterparts. Finally, in Steps 2 and 3, we test against the threshold $\frac{2}{3} - z + \epsilon$ instead of $\frac{2}{3} - z$.

Observe that, when sampled strategies are used, all steps of the algorithm can be carried out in at most $(\log n)^2$ communication. In particular, to implement Step 4, the column player can communicate j^* to the row player, and then the row player can communicate R_{ij^*} for all rows $i \in \text{supp}(\mathbf{x}_s^*)$ using $(\log n)^2$ bits of communication, if we assume that the payoff entries have constant or polylogarithmic length, which allows the column player to determine j'. Once j' has been determined, there are only $2 \cdot \log n$ payoffs in each matrix that are relevant to the algorithm (the payoffs in rows $i \in \text{supp}(\mathbf{x}_s^*)$ in columns j^* and j') and so the two players can communicate all of these payoffs to each other, and then no further communication is necessary.

Now, we must argue that this modified algorithm is correct. Firstly, we argue that if the modified algorithm reaches Step 5, then the rows b and s exist. To do this, we observe that the required preconditions of Lemma 2 are satisfied by $\mathbf{x}_s^*$,

j^*, and j'. Condition 2 holds because the modified Step 3 ensures that the column player's best response payoff is at least $\frac{2}{3} - z + \epsilon > \frac{2}{3} - z$, while Condition 4 is ensured by the explicit check in Step 5. For Conditions 1 and 3, we use the fact that $(\mathbf{x}^*, \mathbf{y}^*)$ is an ϵ-Nash equilibrium of the zero-sum game $(R, -R)$. The following lemma shows that any approximate Nash equilibrium of a zero-sum game behaves like an approximate min-max strategy.

Lemma 4. *If $(\mathbf{x}, \mathbf{y})$ is an ϵ-NE of a zero-sum game $(M, -M)$, then for every strategy $\mathbf{y}'$ we have:*

$$\mathbf{x}^T M \mathbf{y}' \geq \mathbf{x}^T M \mathbf{y} - \epsilon.$$

Proof. Let $v = \mathbf{x}^T M \mathbf{y}$ be the payoff to the row player under $(\mathbf{x}, \mathbf{y})$. Suppose, for the sake of contradiction, that there exists a column player strategy $\mathbf{y}'$ such that:

$$\mathbf{x}^T M \mathbf{y}' < v - \epsilon.$$

Since the game is zero-sum, this means that the column player's payoff under $(\mathbf{x}, \mathbf{y}')$ is strictly larger than $-v + \epsilon$, which then directly implies that the best response payoff for the column player against $\mathbf{x}$ is strictly larger than $-v + \epsilon$. However, since the column player's expected payoff under $(\mathbf{x}, \mathbf{y})$ is $-v$, this then implies that $(\mathbf{x}, \mathbf{y})$ is not an ϵ-NE, which provides our contradiction. □

Since Step 2 suggests that the row player's payoff in $(\mathbf{x}^*, \mathbf{y}^*)$ is at least $\frac{2}{3} - z + \epsilon$, Lemma 4 implies that $\mathbf{x}^*$ secures a payoff of $\frac{2}{3} - z$ no matter what strategy the column player plays, which then implies that Conditions 1 and 3 of Lemma 2 hold.

Finally, we argue that the algorithm finds a $(0.6528 + \epsilon)$. The modified Steps 2 and 3 now return a $(\frac{2}{3} - z + \epsilon)$-WSNE, whereas the approximation guarantees of the other steps are unchanged. Thus, we can reuse our original analysis to obtain the following theorem.

Theorem 3. *For every $\epsilon > 0$, there is a randomized expected-polynomial-time algorithm that uses $O\left(\frac{\log^2 n}{\epsilon^2}\right)$ communication and finds a $(0.6528 + \epsilon)$-WSNE.*

3.3.2 Query Complexity of the Algorithm

We now show that Algorithm 2 can be implemented in a payoff-query efficient manner. There are two results that we will use for this setting. Goldberg and Roth [39] have given a randomized algorithm that, with high probability, finds an ϵ-NE of a zero-sum game using $O(\frac{n \cdot \log n}{\epsilon^2})$ payoff queries. Given a mixed strategy profile $(\mathbf{x}, \mathbf{y})$, an *ϵ-approximate payoff vector* for the row player is a vector v such

that, for all $i \in [n]$ we have $|v_i - (R \cdot \mathbf{y})_i| \leq \epsilon$. Approximate payoff vectors for the column player are defined symmetrically. Fearnley and Savani [32] observed that there is a randomized algorithm that when given the strategy profile $(\mathbf{x}, \mathbf{y})$, finds approximate payoff vectors for both players using $O(\frac{n \cdot \log n}{\epsilon^2})$ payoff queries and that succeeds with high probability. We summarise these two results in the following lemma.

Lemma 5 ([39, 32])**.** *Given an $n \times n$ zero-sum bimatrix game, with probability at least $(1 - n^{-\frac{1}{8}})(1 - \frac{2}{n})^2$, we can compute an ϵ-Nash equilibrium $(\mathbf{x}, \mathbf{y})$, and ϵ-approximate payoff vectors for both players under $(\mathbf{x}, \mathbf{y})$, using $O(\frac{n \cdot \log n}{\epsilon^2})$ payoff queries.*

Let $\epsilon > 0$ be a positive constant. We now outline the changes needed in the algorithm.

- In Step 1 we use the algorithm of Lemma 5 to find $\frac{\epsilon}{2}$-NEs of $(R, -R)$, and $(C, -C)$. We denote the mixed strategies found as $(\mathbf{x}_a^*, \mathbf{y}_a^*)$ and $(\hat{\mathbf{x}}_a, \hat{\mathbf{y}}_a)$, respectively, and we use these strategies in place of their original counterparts throughout the rest of the algorithm. We also compute $\frac{\epsilon}{2}$-approximate payoff vectors for each of these strategies, and use them whenever we need to know the payoff of a particular strategy under one of these strategies. In particular, we set v_r to be the payoff of $\mathbf{x}_a^*$ according to the approximate payoff vector of $\mathbf{y}_a^*$, and we set v_c to be the payoff of $\hat{\mathbf{y}}_a$ according to the approximate payoff vector for $\hat{\mathbf{x}}_a$.

- In Steps 2 and 3 we test against the threshold of $\frac{2}{3} - z + \epsilon$ rather than $\frac{2}{3} - z$.

- In Step 4 we select j^* to be the column that is maximal in the approximate payoff vector against $\mathbf{x}_a^*$. We then spend n payoff queries to query every row in column j^*, which allow us to proceed with the rest of this step as before.

- In Step 5 we use the algorithm of Lemma 5 to find an approximate payoff vector v for the column player against $\mathbf{x_b}$. We then select j' to be a column that maximizes v, and then spend n payoff queries to query every row in j^*, which allows us to proceed with the rest of this step as before.

Observe that the query complexity of the algorithm is $O(\frac{n \cdot \log n}{\epsilon^2})$, where the dominating term arises due to the use of the algorithm from Lemma 5 to approximate solutions to the zero-sum games.

We now argue that this modified algorithm produces a $(0.6528 + \epsilon)$-WSNE. Firstly, we need to reestablish the existence of the rows b and s used in Step 5. To do this, we observe that the preconditions of Lemma 2 hold for $\mathbf{x}_a^*$. We start with Conditions 1 and 3. Note that the payoff for the row player under $(\mathbf{x}_a^*, \mathbf{y}_a^*)$ is at least $v_r - \frac{\epsilon}{2}$ (since v_r was estimated with approximate payoff vectors,) and Step 2 ensures that $v_r > \frac{2}{3} - z + \epsilon$. Hence, we can apply Lemma 4 to argue that $\mathbf{x}_a^*$ secures payoff at least $\frac{2}{3} - z$ against every strategy of the column player, which proves that Conditions 1 and 3 hold. Condition 2 holds because the check in Step 3, ensures that the approximate payoff of j^* against $\mathbf{x}^*$ is at least $\frac{2}{3} - z + \epsilon$, and therefore the actual payoff of j^* against $\mathbf{x}^*$ is at least $\frac{2}{3} - z + \frac{\epsilon}{2}$. Finally, Condition 4 holds because pure strategy profiles of this form are explicitly checked for in Step 5.

Steps 2 and 3 in the modified algorithm return a $(\frac{2}{3} - z + \epsilon)$-WSNE, while the other steps provide the same approximation guarantee as the original algorithm. So, we can reuse the analysis for the original algorithm to prove the following theorem. Observe though that this time we get a randomized algorithm since we use Lemma 5 which finds an ϵ-NE for zero sum games with high probability.

Theorem 4. *There is a randomized algorithm that, with high probability, finds a* $(0.6528 + \epsilon)$*-WSNE using* $O(\frac{n \cdot \log n}{\epsilon^2})$ *payoff queries.*

3.3.3 Proof of Lemma 2

In this section we assume that Steps 1 through 4 of our algorithm did not return a $(\frac{2}{3} - z)$-WSNE, and that neither j^* nor j' contained a pure $(\frac{2}{3} - z)$-WSNE. We show that, under these assumptions, the rows b and s required by Step 5 do indeed exist.

Probability bounds. We begin by proving bounds on the amount of probability that $\mathbf{x}^*$ can place on S and B. The following lemma uses the fact that $\mathbf{x}^*$ secures an expected payoff of at least $\frac{2}{3} - z$ to give an upper bound on the amount of probability that $\mathbf{x}^*$ can place on S. To simplify notation, we use $\Pr(B)$ to denote the probability assigned by $\mathbf{x}^*$ to the rows in B, and we use $\Pr(S)$ to denote the probability assigned by $\mathbf{x}^*$ to the rows in S.

Lemma 6. $\Pr(S) \leq \frac{1+3z}{2-3z}$.

Proof. We will prove our claim using Markov's inequality. Consider the random variable $T = 1 - R_{ij^*}$ where i is sampled from $\mathbf{x}^*$. Since by our assumption the expected payoff of the row player is greater than $2/3 - z$ we get that $E(T) \leq$

$1/3 + z$. If we apply Markov's inequality we get

$$Pr(T \geq \frac{2}{3} - z) \leq \frac{E(T)}{\frac{2}{3} - z} \leq \frac{1 + 3z}{2 - 3z}$$

which is the claimed result. □

Next we show an upper bound on $\Pr(B)$. Here we use the fact that j^* does not contain a $(\frac{2}{3} - z)$-WSNE to argue that all column player payoffs in B are smaller than $\frac{1}{3} + z$. Since we know that the payoff of j^* against $\mathbf{x}^*$ is at least $\frac{2}{3} - z$, this can be used to prove a upper bound on the amount of probability that $\mathbf{x}^*$ assigns to B.

Lemma 7. $\Pr(B) \leq \frac{1+3z}{2-3z}$.

Proof. Since there is no $i \in \text{supp}(\mathbf{x}^*)$ such that (i, j^*) is a pure $(\frac{2}{3} - z)$-WSNE , and since each row $i \in B$ satisfies $R_{i\mathsf{j}^*} \geq \frac{1}{3} + z$, we must have that $C_{i\mathsf{j}^*} < \frac{1}{3} + z$ for every $i \in B$. By assumption we know that $C_{\mathsf{j}^*}^T \mathbf{x}^* > 2/3 - z$. So, we have the following inequality:

$$\frac{2}{3} - z < \Pr(B) \cdot (\frac{1}{3} + z) + \big(1 - \Pr(B)\big) \cdot 1.$$

Solving this inequality for $\Pr(B)$ gives the desired result. □

Payoff inequalities for j^*. We now show properties about the average payoff obtained from the rows in B and S. Recall that $\mathbf{x_b}$ was defined in Step 4 of our algorithm, and that it denotes the normalization of the probability mass assigned by $\mathbf{x}^*$ to rows in B. The following lemma shows that the expected payoff to the row player in the strategy profile $(\mathbf{x_b}, \mathsf{j}^*)$ is close to 1.

Lemma 8. *We have* $(\mathbf{x_b}^T \cdot R)_{\mathsf{j}^*} > \frac{1-6z}{1+3z}$.

Proof. By definition we have that:

$$(\mathbf{x_b}^T \cdot R)_{\mathsf{j}^*} = \frac{1}{\Pr(B)} \cdot \sum_{i \in B} \mathbf{x}_i^* \cdot R_{i\mathsf{j}^*}. \tag{3.1}$$

We begin by deriving a lower bound for $\sum_{i \in B} \mathbf{x}_i^* \cdot R_{i\mathsf{j}^*}$. Using the fact that $\mathbf{x}^*$ secures an expected payoff of at least $2/3 - z$ against j^* and then applying the bound from Lemma 6 gives:

$$\begin{aligned} \frac{2}{3} - z &< \sum_{i \in B} \mathbf{x}_i^* \cdot R_{i\mathsf{j}^*} + (\frac{1}{3} + z) \cdot \Pr(S) \\ &\leq \sum_{i \in B} \mathbf{x}_i^* \cdot R_{i\mathsf{j}^*} + (\frac{1}{3} + z) \cdot \frac{1 + 3z}{2 - 3z}. \end{aligned}$$

Hence we can conclude that:

$$\begin{aligned}\sum_{i\in B}\mathbf{x}_i^*\cdot R_{ij^*} &> \frac{2}{3}-z-\frac{1}{3}\cdot\frac{(1+3z)^2}{2-3z}\\ &= \frac{1-6z}{2-3z}.\end{aligned}$$

Substituting this into Equation (3.1), along with the upper bound on $\Pr(B)$ from Lemma 7, allows us to conclude that:

$$\begin{aligned}(\mathbf{x_b}^T\cdot R)_{\mathrm{j}^*} &\geq \frac{2-3z}{1+3z}\cdot\sum_{i\in B}\mathbf{x}_i^*\cdot R_{ij^*}\\ &> \frac{2-3z}{1+3z}\cdot\frac{1-6z}{2-3z}\\ &= \frac{1-6z}{1+3z}.\end{aligned}$$

□

Next we would like to show a similar bound on the expected payoff to the column player of the rows in S. To do this, we define $\mathbf{x_s}$ to be the normalisation of the probability mass that $\mathbf{x}^*$ assigns to the rows in S. More formally, for each $i\in[n]$, we define:

$$(\mathbf{x_s})_i=\begin{cases}\frac{1}{\Pr(S)}\cdot\mathbf{x}_i^* & \text{if } i\in S\\ 0 & \text{otherwise.}\end{cases}$$

The next lemma shows that the expected payoff to the column player in the profile $(\mathbf{x_s},\mathrm{j}^*)$ is close to 1.

Lemma 9. *We have* $(\mathbf{x_s}^T\cdot C)_{\mathrm{j}^*}>\frac{1-6z}{1+3z}$.

Proof. By definition we have that:

$$(\mathbf{x_s}^T\cdot C)_{\mathrm{j}^*}=\frac{1}{\Pr(S)}\cdot\sum_{i\in S}\mathbf{x}_i^*\cdot C_{ij^*}. \tag{3.2}$$

We begin by deriving a lower bound for $\sum_{i\in S}\mathbf{x}_i^*\cdot C_{ij^*}$. By assumption, we know that $C_{\mathrm{j}^*}^T\mathbf{x}^*>2/3-z$. Moreover, since j^* does not contain a $(\frac{2}{3}-z)$-WSNE we have that all rows i in B satisfy $C_{ij^*}<1/3-z$. If we combine these facts that with Lemma 7 we obtain:

$$\begin{aligned}\frac{2}{3}-z &< \sum_{i\in S}\mathbf{x}_i^*\cdot C_{ij^*}+(\frac{1}{3}+z)\cdot\Pr(B)\\ &\leq \sum_{i\in S}\mathbf{x}_i^*\cdot C_{ij^*}+(\frac{1}{3}+z)\cdot\frac{1+3z}{2-3z}.\end{aligned}$$

Hence we can conclude that:

$$\begin{aligned}\sum_{i \in S} \mathbf{x}_i^* \cdot C_{ij^*} &> \frac{2}{3} - z - \frac{1}{3} \cdot \frac{(1+3z)^2}{2-3z} \\ &= \frac{1-6z}{2-3z}.\end{aligned}$$

Substituting this into Equation (3.2), along with the upper bound on $\Pr(S)$ from Lemma 7, allows us to conclude that:

$$\begin{aligned}({\mathbf{x_b}}^T \cdot R)_{\mathsf{j}^*} &\geq \frac{2-3z}{1+3z} \cdot \sum_{i \in B} \mathbf{x}_i^* \cdot R_{ij^*} \\ &> \frac{2-3z}{1+3z} \cdot \frac{1-6z}{2-3z} \\ &= \frac{1-6z}{1+3z}.\end{aligned}$$

□

Payoff inequalities for j'. We now want to prove similar inequalities for the column j'. The next lemma shows that the expected payoff for the column player in the profile $(\mathbf{x_b}, \mathsf{j}')$ is close to 1. This is achieved by first showing a lower bound on the payoff to the column player in the profile $(\mathbf{x_b}, \mathsf{j}^*)$, and then using the fact that j^* is not a $(\frac{2}{3} - z)$-best-response against $\mathbf{x_b}$, and that j' is a best response against $\mathbf{x_b}$.

Lemma 10. *We have* $({\mathbf{x_b}}^T \cdot C)_{\mathsf{j}'} > \frac{1-6z}{1+3z}$.

Proof. We first establish a lower bound on $({\mathbf{x_b}}^T \cdot C)_{\mathsf{j}^*}$. By assumption, we know that $C_{\mathsf{j}^*}^T \mathbf{x}^* > 2/3 - z$. Using this fact, along with the bounds from Lemmas 6 and 7 gives:

$$\begin{aligned}\frac{2}{3} - z &< \Pr(B) \cdot ({\mathbf{x_b}}^T \cdot C)_{\mathsf{j}^*} + \Pr(S) \cdot 1 \\ &\leq \frac{1+3z}{2-3z} \cdot ({\mathbf{x_b}}^T \cdot C)_{\mathsf{j}^*} + \frac{1+3z}{2-3z}.\end{aligned}$$

Solving this inequality for $({\mathbf{x_b}}^T \cdot C)_{\mathsf{j}^*}$ yields:

$$({\mathbf{x_b}}^T \cdot C)_{\mathsf{j}^*} > \frac{1}{3} \cdot \frac{1-21z+9z^2}{1+3z}.$$

Now we can prove the lower bound on $(\mathbf{x_b}^T \cdot C)_{\mathsf{j}'}$. Since j^* is not a $(\frac{2}{3} - z)$-best-response against $\mathbf{x_b}$, and since j' is a best response against $\mathbf{x_b}$ we obtain:

$$\begin{aligned}(\mathbf{x_b}^T \cdot C)_{\mathsf{j}'} &> (\mathbf{x_b}^T \cdot C)_{\mathsf{j}^*} + \frac{2}{3} - z \\ (\mathbf{x_b}^T \cdot C)_{\mathsf{j}'} &> \frac{1}{3} \cdot \frac{1 - 21z + 9z^2}{1 + 3z} + \frac{2}{3} - z \\ &= \frac{1 - 6z}{1 + 3z}.\end{aligned}$$

□

The only remaining inequality that we require is a lower bound on the expected payoff to the row player in the profile $(\mathbf{x_s}, \mathsf{j}')$. However, before we can do this, we must first prove an upper bound on the expected payoff to the row player in $(\mathbf{x_b}, \mathsf{j}')$, which we do in the following lemma. Here we first prove that most of the probability mass of $\mathbf{x_b}$ is placed on rows i where $C_{i\mathsf{j}'} > \frac{1}{3} + z$, which when combined with the fact that there is no $i \in \text{supp}(\mathbf{x}^*)$ such that (i, j') is a pure $(\frac{2}{3} - z)$-WSNE, is sufficient to provide an upper bound.

Lemma 11. *We have* $(\mathbf{x_b}^T \cdot R)_{\mathsf{j}'} < \frac{1}{3} \cdot \frac{1+33z+9z^2}{1+3z}$.

Proof. We begin by proving an upper bound on the amount of probability mass assigned by $\mathbf{x_b}$ to rows i with $C_{i\mathsf{j}'} < \frac{1}{3} + z$. Let $T = 1 - C_{i\mathsf{j}'}$ be a random variable where the row i is sampled according to $\mathbf{x_b}$. Lemma 10 implies that:

$$E[T] = 1 - \frac{1 - 6z}{1 + 3z} = \frac{9z}{1 + 3z}.$$

Observe that $\Pr(T \geq 1 - (\frac{1}{3} + z)) = \Pr(T \geq \frac{2}{3} - z)$ is equal to the amount of probability that $\mathbf{x_b}$ assigns to rows i with $C_{i\mathsf{j}'} < \frac{1}{3} + z$. Applying Markov's inequality then establishes our bound.

$$\Pr(T \geq \frac{2}{3} - z) \leq \frac{\frac{9z}{1+3z}}{\frac{2}{3} - z}.$$

So, if $p = \frac{9z}{(1+3z)(2/3-z)}$ then we know that at least $1 - p$ probability is assigned by $\mathbf{x_b}$ to rows i such that $C_{i\mathsf{j}'} > \frac{1}{3} + z$. Since we have assumed that there is no $i \in \text{supp}(\mathbf{x}^*)$ such that (i, j') is a pure $(\frac{2}{3} - z)$-WSNE, we know that any such row i must satisfy $R_{i\mathsf{j}'} < \frac{1}{3} + z$. Hence, we obtain the following bound:

$$\begin{aligned}(\mathbf{x_b}^T \cdot R)_{\mathsf{j}'} &< (1 - p) \cdot (\frac{1}{3} + z) + p \\ &= \frac{1}{3} \cdot \frac{1 + 33z + 9z^2}{1 + 3z}.\end{aligned}$$

□

Finally, we show that the expected payoff to the row player in the profile $(\mathbf{x_s}, \mathsf{j}')$ is close to 1. Here we use the fact that $\mathbf{x}^*$ is a min-max strategy along with the bound from Lemma 11 to prove our lower bound.

Lemma 12. *We have* $({\mathbf{x_s}}^T \cdot R)_{\mathsf{j}'} > \frac{1-15z}{1+3z}$.

Proof. Since $\mathbf{x}^*$ is a min-max strategy that secures a value strictly larger than $\frac{2}{3} - z$, we have:

$$\frac{2}{3} - z < \Pr(B) \cdot ({\mathbf{x_b}}^T \cdot R)_{\mathsf{j}'} + \Pr(S) \cdot ({\mathbf{x_s}}^T \cdot R)_{\mathsf{j}'}.$$

Substituting the bounds from Lemmas 6, 7, and 11 then gives:

$$\frac{2}{3} - z < \frac{1+3z}{2-3z} \cdot \frac{1}{3} \cdot \frac{1+33z+9z^2}{1+3z} + \frac{1+3z}{2-3z} \cdot ({\mathbf{x_s}}^T \cdot R)_{\mathsf{j}'}.$$

Solving for $({\mathbf{x_s}}^T \cdot R)_{\mathsf{j}'}$ then yields the desired result. □

Finding rows b and s. So far, we have shown that the expected payoff to the row player in $(\mathbf{x_b}, \mathsf{j}^*)$ is close to 1, and that the expected payoff to the column player in $(\mathbf{x_b}, \mathsf{j}')$ is close to 1. We now show that there exists a row $b \in B$ such that $R_{b\mathsf{j}^*}$ is close to 1, and $C_{b\mathsf{j}'}$ is close to 1, and that there exists a row $s \in S$ in which $C_{s\mathsf{j}^*}$ and $R_{s\mathsf{j}'}$ are both close to 1. The following lemma uses Markov's inequality to show a pair of probability bounds that will be critical in showing the existence of b.

Lemma 13. *We have:*

- $\mathbf{x_b}$ *assigns strictly more than* 0.5 *probability to rows* i *with* $R_{i\mathsf{j}^*} > 1 - \frac{18z}{1+3z}$.
- $\mathbf{x_b}$ *assigns strictly more than* 0.5 *probability to rows* i *with* $C_{i\mathsf{j}'} > 1 - \frac{18z}{1+3z}$.

Proof. We begin with the first case. Consider the random variable $T = 1 - R_{i\mathsf{j}^*}$ where i is sampled from $\mathbf{x_b}$. By Lemma 8, we have that:

$$E[T] < 1 - \frac{1-6z}{1+3z} = \frac{9z}{1+3z}.$$

We have that $T \geq \frac{18z}{1+3z}$ whenever $R_{i\mathsf{j}^*} \leq 1 - \frac{18z}{1+3z}$, so we can apply Markov's inequality to obtain:

$$\Pr(T \geq \frac{18z}{1+3z}) < \frac{\frac{9z}{1+3z}}{\frac{18z}{1+3z}} = 0.5.$$

The proof of the second case is identical to the proof given above, but uses the (identical) bound from Lemma 10. □

The next lemma uses the same techniques to prove a pair of probability bounds that will be used to prove the existence of s.

Lemma 14. *We have:*

- $\mathbf{x_s}$ *assigns strictly more than* $\frac{1}{3}$ *probability to rows* i *with* $C_{ij^*} > 1 - \frac{27z}{1+3z}$.
- $\mathbf{x_s}$ *assigns strictly more than* $\frac{2}{3}$ *probability to rows* i *with* $R_{ij'} > 1 - \frac{27z}{1+3z}$.

Proof. We begin with the first claim. Consider the random variable $T = 1 - C_{ij^*}$ where i is sampled from $\mathbf{x_s}$. By Lemma 9, we have that:

$$E[T] < 1 - \frac{1-6z}{1+3z} = \frac{9z}{1+3z}.$$

We have that $T \geq \frac{27z}{1+3z}$ whenever $C_{ij^*} \leq 1 - \frac{27z}{1+3z}$, so we can apply Markov's inequality to obtain:

$$\Pr(T \geq \frac{27z}{1+3z}) < \frac{\frac{9z}{1+3z}}{\frac{27z}{1+3z}} = \frac{1}{3}.$$

We now move on to the second claim. Consider the random variable $T = 1 - R_{ij^*}$ where i is sampled from $\mathbf{x_b}$. By Lemma 12, we have that:

$$E[T] < 1 - \frac{1-15z}{1+3z} = \frac{18z}{1+3z}.$$

We have that $T \geq \frac{27z}{1+3z}$ whenever $R_{ij^*} \leq 1 - \frac{27z}{1+3z}$, so we can apply Markov's inequality to obtain:

$$\Pr(T \geq \frac{27z}{1+3z}) < \frac{\frac{18z}{1+3z}}{\frac{27z}{1+3z}} = \frac{2}{3}.$$

□

Finally, we can formally prove the existence of b and s, which completes the proof of correctness for our algorithm.

Proof of Lemma 2. We begin by proving the first claim. If we sample a row b randomly from $\mathbf{x_b}$, then Lemma 13 implies that probability that $R_{bj^*} \leq 1 - \frac{18z}{1+3z}$ is strictly less than 0.5 and that the probability that $C_{bj'} \leq 1 - \frac{18z}{1+3z}$ is strictly less than 0.5. Hence, by the union bound, the probability that at least one of these events occurs is strictly less than 1. So, there is a positive probability that neither of the events occurs, which implies that there exists at least one row b that satisfies the desired properties.

The second claim is proved using exactly the same technique, but using the bounds from Lemma 14, again observing that the probability that a randomly sampled row from $\mathbf{x_s}$ satisfies the desired properties with positive probability. □

This completes the proof of Lemma 2.

3.4 A Communication-Efficient Algorithm for Finding a 0.5-WSNE in win-lose Bimatrix Games

The base algorithm (Algorithm 1) can be adapted to provide a communication efficient method for finding a $(0.5+\epsilon)$-WSNE in win-lose games; bimatrix games where all the payoff entries for the matrices R and C are 0 or 1. In brief, the algorithm can be modified to find a 0.5-WSNE in a win-lose game by making Steps 2 and 3 check against the threshold of 0.5. It can then be shown that if these steps fail, then there exists a pure Nash equilibrium in column $\mathfrak{j}^*$. This can then be implemented as a communication efficient protocol using the algorithm from Lemma 3.

Algorithm 3

1. Solve the zero-sum games $(R, -R)$ and $(-C, C)$.
   - Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a NE of $(R, -R)$, and let $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ be a NE of $(C, -C)$.
   - Let v_r be the value secured by $\mathbf{x}^*$ in $(R, -R)$, and let v_c be the value secured by $\hat{\mathbf{y}}$ in $(-C, C)$. Without loss of generality assume that $v_c \leq v_r$.
2. If $v_r \leq 0.5$, then return $(\hat{\mathbf{x}}, \mathbf{y}^*)$.
3. If for all $j \in [n]$ it holds that $C_j^T \cdot \mathbf{x}^* \leq 0.5$, then return $(\mathbf{x}^*, \mathbf{y}^*)$.
4. Otherwise:
   - Let $\mathfrak{j}^*$ be a pure best response to $\mathbf{x}^*$.
   - Find a row i such that $R_{i\mathfrak{j}^*} = 1$ and $C_{ij} = 1$.
   - Return $(i, \mathfrak{j}^*)$.

We will show that this algorithm always finds a 0.5-WSNE in a win-lose game. Firstly, we show that if the algorithm returns the strategy profile $(i, \mathfrak{j}^*)$ in Step 4, then this is a pure Nash equilibrium for (R, C). The following lemma is similar to Lemma 1, but exploits the fact that the game is win-lose to obtain a stronger conclusion.

Lemma 15. *If Algorithm 3 is applied to a win-lose game, and it reaches Step 4, then then there exists a row $i \in \text{supp}(\mathbf{x}^*)$ such that $R_{i\mathfrak{j}^*} = 1$ and $C_{i\mathfrak{j}^*} = 1$.*

Proof. Let i be a row sampled from $\mathbf{x}^*$. We will show that there is a positive probability that row i satisfies the desired properties.

We begin by showing that the probability that $\Pr(R_{ij^*} = 0) < 0.5$. Let the random variable $T = 1 - R_{ij^*}$. Since $v_r > \frac{1}{2}$, we have that $E[T] < 0.5$. Thus, applying Markov's inequality we obtain:

$$\Pr(T \geq 1) \leq \frac{E[T]}{1} < 0.5.$$

Since $\Pr(R_{ij^*} = 0) = \Pr(T \geq 1)$ we can therefore conclude that $\Pr(R_{ij^*} = 0) < 0.5$. The exact same technique can be used to prove that $\Pr(C_{ij^*} = 0) < 0.5$, by using the fact that $C_{\mathrm{j}^*}^T \cdot \mathbf{x}^* > 0.5$.

We can now apply the union bound to argue that:

$$\Pr(R_{ij^*} = 0 \text{ or } C_{ij^*} = 0) < 1.$$

Hence, there is positive probability that row i satisfies $R_{ij^*} > 0$ and $C_{ij^*} > 0$, so such a row must exist. The final step is to observe that, since the game is win-lose, we have that $R_{ij^*} > 0$ implies $R_{ij^*} = 1$, and that $C_{ij^*} > 0$ implies $C_{ij^*} = 1$. □

We now prove that the algorithm always finds a 0.5-WSNE. The reasoning is very similar to the analysis of the base algorithm. The strategy profiles returned by Steps 2 and 3 are 0.5-WSNEs by the same reasoning that was given for the base algorithm. Step 4 always returns a pure Nash equilibrium.

3.4.1 Communication Complexity of the Algorithm

We now show that Algorithm 3 can be implemented in a communication efficient way.

The zero-sum games in Step 1 can be solved by the two players independently without any communication. Then, the players exchange v_r and v_s using $O(\log n)$ rounds of communication. If both v_r and v_s are smaller than 0.5, then the algorithm from Lemma 3 is applied to communicate $\hat{\mathbf{x}}_s$ to the row player, and $\mathbf{y}_s^*$ to the column player. Since the payoffs under the sampled strategies are within ϵ of the originals, we have that all pure strategies have payoff less than or equal to $0.5 + \epsilon$ under $(\hat{\mathbf{x}}_s, \mathbf{y}_s^*)$, so this strategy profile is a $(0.5 + \epsilon)$-WSNE.

We will assume from now on that $v_r > v_c$. If the algorithm reaches Step 3, then the row player uses the algorithm of Lemma 3 to communicate $\mathbf{x}_s^*$ to the column player. The column player then computes a best response j_s^* against $\mathbf{x}_s^*$, and checks whether the payoff of j_s^* against $\mathbf{x}_s^*$ is less than or equal to $0.5 + \epsilon$. If so, then the players output $(\mathbf{x}_s^*, \mathrm{j}_s^*)$, which is a $0.5 + \epsilon$-WSNE

Otherwise, we claim that there is a pure strategy $i \in \mathrm{supp}(\mathbf{x}_s^*)$ such that (i, j_s^*) is a pure Nash equilibrium. This can be shown by observing that the expected

payoff of $\mathbf{x}_s^*$ against j_s^* is at least $0.5 - \epsilon$, while the expected payoff of j_s^* against $\mathbf{x}_s^*$ is at least $0.5 + \epsilon$. Repeating the proof of Lemma 15 using these inequalities then shows that the pure Nash equilibrium does indeed exist. Since $\text{supp}(\mathbf{x}_s^*)$ has logarithmic size, the row player can simply transmit to the column player all payoffs $R_{ij_s^*}$ for which $i \in \text{supp}(\mathbf{x}_s^*)$, and the column player can then send back a row corresponding to a pure Nash equilibrium.

In conclusion, we have shown that a $(0.5+\epsilon)$-WSNE can be found in randomized expected-polynomial-time using $O\left(\frac{\log^2 n}{\epsilon^2}\right)$ communication.

Theorem 5. *For every win-lose game and every $\epsilon > 0$, there is a randomized expected-polynomial-time algorithm that uses $O\left(\frac{\log^2 n}{\epsilon^2}\right)$ communication and finds a $(0.5 + \epsilon)$-WSNE.*

3.5 A Communication-Efficient Algorithm for Finding a $\left(\frac{3-\sqrt{5}}{2} + \epsilon\right)$-NE

We will study the following algorithm, which is inspired by the algorithm of Bose, Byrka, and Markakis [7] and we will show that it finds a $\left(\frac{3-\sqrt{5}}{2} + \epsilon\right)$-NE in a communication efficient way.

Algorithm 4

1. Solve the zero-sum games $(R, -R)$ and $(-C, C)$.
   - Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a NE of $(R, -R)$, and let $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ be a NE of $(C, -C)$.
   - Let v_r be the value secured by $\mathbf{x}^*$ in $(R, -R)$, and let v_c be the value secured by $\hat{\mathbf{y}}$ in $(-C, C)$. Without loss of generality assume that $v_c \leq v_r$.
   - If $v_r \leq \frac{3-\sqrt{5}}{2}$, return $(\hat{\mathbf{x}}, \mathbf{y}^*)$.
2. Otherwise:
   - Let j be a best response for the column player against $\mathbf{x}^*$.
   - Let r be a best response for the row player against j.
   - Define the strategy profile $\mathbf{x}' = \frac{1}{2-v_r} \cdot \mathbf{x}^* + \frac{1-v_r}{2-v_r} \cdot r$.
   - Return $(\mathbf{x}', j)$.

We show that this algorithm always produces a $\frac{3-\sqrt{5}}{2}$-NE. We start by considering the case the strategy profile by Step 1 is returned by our algorithm. The

maximum payoff that the row player can achieve against $\mathbf{y}^*$ is v_r, so the row player's regret can be at most v_r in the bimatrix game (R, C). This is because the strategy profile $(\mathbf{x}^*, \mathbf{y}^*)$ is an NE for the zero-sum game $(R, -R)$. So, $\mathbb{X}^*$ is a best response to $\mathbf{y}^*$ and thus is a strategy that achieves the maximum payoff for the row player against $\mathbf{y}^*$. Similarly, the maximum payoff that the column player can achieve against $\hat{\mathbf{x}}$ is $v_c \leq v_r$, so the column player's regret can be at most v_r. Step 1 only returns a strategy profile in the case where $v_r \leq \frac{3-\sqrt{5}}{2}$, so this step always produces a $\frac{3-\sqrt{5}}{2}$-NE.

To analyse the quality of approximate equilibrium found by Step 2, we use the following Lemma.

Lemma 16. *The strategy profile $(\mathbf{x}', j)$ is a $\frac{1-v_r}{2-v_r}$-NE.*

Proof. We start by analysing the regret of the row player. By definition, row r is a best response against column j. So, the regret of the row player can be expressed as:

$$\begin{aligned} R_{rj} - (\mathbf{x}' \cdot R)_j &= R_{rj} - \frac{1}{2-v_R} \cdot ((\mathbf{x}^*)^T \cdot R)_j - \frac{1-v_R}{2-v_R} \cdot R_{rj} \\ &\leq \frac{1}{2-v_R} \cdot R_{rj} - \frac{1}{2-v_R} \cdot v_R \\ &\leq \frac{1}{2-v_R} \cdot 1 - \frac{1}{2-v_R} \cdot v_r \\ &= \frac{1-v_R}{2-v_R}, \end{aligned}$$

where in the first inequality we use the fact that $\mathbf{x}^*$ is a min-max strategy that secures payoff at least v_r, and the second inequality uses the fact that $R_{rj} \leq 1$.

We now analyse the regret of the column player. Let c be a best response for the column player against $\mathbf{x}'$. The regret of the column player can be expressed as:

$$\begin{aligned} &((\mathbf{x}')^T \cdot C)_c - ((\mathbf{x}')^T \cdot C)_j \\ &= \frac{1}{2-v_R} \cdot ((\mathbf{x}^*)^T \cdot C)_c + \frac{1-v_R}{2-v_R} \cdot C_{rc} - \frac{1}{2-v_R} \cdot ((\mathbf{x}^*)^T \cdot C)_{x^* j} - \frac{1-v_R}{2-v_R} \cdot C_{rj} \\ &\leq \frac{1-v_R}{2-v_R} \cdot C_{rc} - \frac{1-v_R}{2-v_R} \cdot C_{rj} \\ &\leq \frac{1-v_R}{2-v_R}. \end{aligned}$$

The first inequality holds since j is a best response against x^*, and therefore $((\mathbf{x}^*)^T \cdot C)_c \leq ((\mathbf{x}^*)^T \cdot C)_j$, and the second inequality holds since $C_{rc} \leq 1$ and $C_{rj} \geq 0$. Thus, we have shown that both players have regret at most $\frac{1-v_r}{2-v_r}$ under $(\mathbf{x}', j)$, and therefore $(\mathbf{x}', j)$ is a $\frac{1-v_r}{2-v_r}$-NE. □

Step 2 is only triggered in the case where $v_r > \frac{3-\sqrt{5}}{2}$, and we have that $\frac{1-v_r}{2-v_r} = \frac{3-\sqrt{5}}{2}$ when $v_r = \frac{3-\sqrt{5}}{2}$. Since $\frac{1-v_r}{2-v_r}$ decreases as v_r increases, we therefore have that Step 2 always produces a $\frac{3-\sqrt{5}}{2}$-NE. This completes the proof of correctness for the algorithm.

3.5.1 Communication Complexity of the Algorithm

We now argue that, for every $\epsilon > 0$ the Algorithm 4 can be used to find a $\left(\frac{3-\sqrt{5}}{2} + \epsilon\right)$-NE using $O\left(\frac{\log^2 n}{\epsilon^2}\right)$ rounds of communication.

We begin by considering Step 1. Obviously, the zero-sum games can be solved by the two players independently without any communication. Then, the players exchange v_r and v_c using $O(\log n)$ rounds of communication. If both v_r and v_c are smaller than $\frac{3-\sqrt{5}}{2}$, then the algorithm from Lemma 3 is applied to communicate $\hat{\mathbf{x}}_s$ to the row player, and $\mathbf{y}_s^*$ to the column player. Since the payoffs under the sampled strategies are within ϵ of the originals, we have that $(\hat{\mathbf{x}}_s, \mathbf{y}_s^*)$ is a $\left(\frac{3-\sqrt{5}}{2} + \epsilon\right)$-NE.

If the algorithm reaches Step 2, then the row player uses the algorithm of Lemma 3 to communicate $\mathbf{x}_s^*$ to the column player. The column player then computes a best response j_s against $\mathbf{x}_s^*$, and uses $\log n$ communication rounds to transmit it to the row player. The row player then computes a best response r_s against j_s, then computes: $\mathbf{x}'_s = \frac{1}{2-v_r} \cdot \mathbf{x}_s^* + \frac{1-v_r}{2-v_r} \cdot r$, and the players output $(\mathbf{x}'_s, j_s)$. To see that this produces a $\left(\frac{3-\sqrt{5}}{2} + \epsilon\right)$-NE, observe that $\mathbf{x}_s^*$ secures a payoff of at least $v_r - \epsilon$ for the row player, and repeating the proof of Lemma 16 with this weaker inequality gives that this strategy profile is a $\left(\frac{1-v_r}{2-v_r} + \epsilon\right)$-NE.

Therefore, we have shown the following theorem.

Theorem 6. *For every $\epsilon > 0$, there is a randomized expected-polynomial-time algorithm that uses $O\left(\frac{\log^2 n}{\epsilon^2}\right)$ communication and finds a $\left(\frac{3-\sqrt{5}}{2} + \epsilon\right)$-NE.*

Chapter 4

Computing Approximate Nash Equilibria in Polymatrix Games

In this chapter, we present an algorithm that, for every δ in the range $0 < \delta \leq 0.5$, finds a $(0.5 + \delta)$-Nash equilibrium of a polymatrix game in time polynomial in the input size and $\frac{1}{\delta}$.

This result is proven by adapting the algorithm of Tsaknakis and Spirakis [66] (henceforth referred to as the `TS` algorithm). They give a gradient descent algorithm for finding a 0.3393-Nash equilibrium in a bimatrix game. We generalise their gradient descent techniques to the polymatrix setting, and show that it always arrives at a $(0.5 + \delta)$-Nash equilibrium after a polynomial number of iterations.

4.1 Polymatrix games preliminaries

An n-player polymatrix game is defined by an undirected graph $G = (V, E)$ with n vertices, where every vertex corresponds to a player. The edges of the graph specify which players interact with each other. For each $i \in [n]$, we use $N(i) = \{j \ : \ (i, j) \in E\}$ to denote the neighbors of player i. Each edge $(i, j) \in E$ specifies that a bimatrix game will be played between players i and j. Each player $i \in [n]$ has a fixed number of pure strategies m_i, and the bimatrix game on edge $(i, j) \in E$ will therefore be specified by an $m_i \times m_j$ matrix A_{ij}, which gives the payoffs for player i, and an $m_j \times m_i$ matrix A_{ji}, which gives the payoffs for player j. Thus, on the edge $(i, j) \in E$ the bimatrix game $\big(A_{ij}, A_{ji}\big)$ is played. In order to play the game, each player i chooses a strategy $x_i \in \Delta_{m_i}$ and plays that strategy simultaneously in all bimatrix games he participates in. The expected payoff for a player is the sum of the expected payoffs over all the bimatrix games he is involved. Formally, if $\mathbb{X} \in \Delta$ denotes a mixed strategy profile for the game,

where $\mathbb{X} = (x_1, \ldots, x_n)$ and $\Delta = \Delta_{m_1} \times \ldots \times \Delta_{m_n}$, then the payoff of player i under $\mathbb{X}$ is

$$u_i(\mathbb{X}) := x_i^T \sum_{j \in N(i)} A_{ij} x_j.$$

We denote by $u_i(x_i', \mathbb{X})$ the payoff for player i when he plays x_i' and the other players play according to the strategy profile $\mathbb{X}$. Let $v_i(\mathbb{X})$ be the vector of payoffs for each pure strategy of player i when the rest of players play strategy profile $\mathbb{X}$. Formally:

$$v_i(\mathbb{X}) = \sum_{j \in N(i)} A_{ij} x_j.$$

For each vector $x \in R^m$, we define suppmax(x) to be the set of indices that achieve the maximum of x, that is, we define suppmax$(x) = \{i \in [m] : x_i \geq x_j, \forall j \in [m]\}$. Then the *pure best responses* of player i against a strategy profile $\mathbb{X}$ (where only $\mathbb{X}_{-i}$ is relevant) is given by:

$$\mathrm{Br}_i(\mathbb{X}) = \text{suppmax}\left(\sum_{j \in N(i)} A_{ij} x_j\right) = \text{suppmax}(v_i(\mathbb{X})). \tag{4.1}$$

The corresponding *best response payoff* is given by:

$$u_i^*(\mathbb{X}) = \max_{k \in m_i} \left\{ \left(\sum_{j \in N(i)} A_{ij} x_j \right)_k \right\} = \max_{k \in m_i} \left\{ \left(v_i(\mathbb{X}) \right)_k \right\}. \tag{4.2}$$

4.1.1 Payoff Normalisation

It is common, when proving results about additive notions of approximate equilibria, to rescale the payoffs of the game. This is necessary in order for different results to be comparable. For example, all results about additive approximate equilibria in bimatrix games assume that the payoff matrices have entries in the range $[0, 1]$, and therefore an ϵ-Nash equilibrium always has a consistent meaning. For the same reason, we must rescale the payoffs in a polymatrix in order to give a consistent meaning to an ϵ-approximation.

An initial, naive, approach would be to specify that each of the individual bimatrix games has entries in the range $[0, 1]$. This would be sufficient if we were only interested in polymatrix games played on either complete graphs or regular graphs. However, in this model, if the players have differing degrees, then they also have differing maximum payoffs. This means that an additive approximate

equilibrium must pay more attention to high degree players, as they can have larger regrets.

One solution to this problem is to apply degree based scaling, i.e. to rescale according to the degree. That is, given a polymatrix game where each bimatrix game has payoffs in the range $[0, 1]$, if a player has degree d, then each of his payoff matrices is divided by d. This transformation ensures that every player has regret in the range $[0, 1]$, and therefore low degree players are not treated unfairly by additive approximations.

However, rescaling according to the degree assumes that each bimatrix game actually uses the full range of payoffs in$[0, 1]$. In particular, some bimatrix games may have minimum payoff strictly greater than 0, or maximum payoff strictly less than 1. This issue arises, in particular, in our application of two-player Bayesian games. Note that, unlike the case of a single bimatrix game, we cannot fix this by rescaling individual bimatrix games in a polymatrix game, because we must preserve the relationship between the payoffs in all of the bimatrix games that a player is involved in.

To address this, we will rescale the games so that, for each player, the minimum possible payoff is 0, and the maximum possible payoff is 1. For each player i, we denote by U_i the maximum payoff he can obtain, and by L_i the minimum payoff he can obtain. Formally:

$$U_i := \max_{p \in [m_i]} \left(\sum_{j \in N(i)} \max_{q \in [m_j]} \left(A_{ij}(p, q) \right) \right),$$

$$L_i := \min_{p \in [m_i]} \left(\sum_{j \in N(i)} \min_{q \in [m_j]} \left(A_{ij}(p, q) \right) \right).$$

Then, for all i and all $j \in N(i)$ we will apply the following transformation, which we call $T(\cdot)$, to all the entries z of payoff matrices A_{ij}:

$$T_i(z) = \frac{1}{U_i - L_i} \cdot \left(z - \frac{L_i}{d(i)} \right).$$

Notice that in order the above normalisation to be well defined we need to assume that $U_i \neq L_i$. We argue that this is not a restriction to our normalisation. If $U_i = L_i$, then the player i gets the same payoff irrespectively from the strategy profile played in the game, thus has regret zero. Hence, we can just do not consider this player in the algorithm.

Observe that, since player i's payoff is the sum of $d(i)$ many bimatrix games, it must be the case that after transforming the payoff matrices in this way, player

i's maximum possible payoff is 1, and player i's minimum possible payoff is 0. In what follows, when we study polymatrix games, we will assume that the payoff matrices given by A_{ij} are rescaled in this way.

It is worth noting that this rescaling is stronger than degree-based rescaling: every ϵ-NE under this rescaling is also an ϵ-NE under the degree-based rescaling, but the converse does not hold.

4.2 The TS Algorithm

The TS algorithm was proposed for computing ϵ-Nash equilibria in bimatrix games. The algorithm starts from an arbitrary strategy profile $\mathbb{X}$ and uses a gradient descent like approach which iteratively decreases the maximum regret a player suffer.

In the TS algorithm, when each player has m pure strategies, a function $f : \Delta_m \times \Delta_m \to [0, 1]$ defined. The value of f under the strategy profile $(\mathbf{x}, \mathbf{y})$ is equal to the maximum regret a player suffers. In each iteration of the algorithm the value of the function f, i.e. the maximum regret the players suffer, strictly decreases. Firstly, if the regrets the players suffer are not equal, then the algorithm equalizes them by computing an appropriate strategy profile using a linear program. Next, when the regrets are equal, the algorithm computes the direction that the gradient of the function f is minimized, that is a strategy profile $(\mathbf{x}', \mathbf{y}')$ which specifies the direction that the regret decreases with the highest rate. Then it produces a new profile by combining the strategy profiles $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$. The algorithm stops when it reaches a stationary point of the function f, i.e. a strategy profile $(\mathbf{x}^*, \mathbf{y}^*)$ such that the maximum regret the players suffer weakly increases in every possible direction $(\mathbf{x}', \mathbf{y}')$. This is a stationary point for the regret function.

Given a stationary point $(\mathbf{x}^*, \mathbf{y}^*)$ computed by the procedure described above, the algorithm derives another two strategies $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ and $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ that depend on $(\mathbf{x}^*, \mathbf{y}^*)$ and the best responses of each player against this strategy profile. It is proven that every time one of these three strategy profiles is a 0.3393-Nash equilibrium.

4.3 The Descent Algorithm

As in TS, we define a function f that is equal to the maximum regret the players suffer under a strategy profile and we apply the steepest descent method in order

to compute an (approximate) stationary point of f.

In order to generalise the `TS` algorithm, we had to overcome several issues. Firstly, the TS algorithm makes the regrets of the two players equal in every iteration, but there is no obvious way to achieve this in the polymatrix setting. Instead, we show how gradient descent can be applied to a strategy profile where the regrets are not necessarily equal. Secondly, the output of the `TS` algorithm is either a point found by gradient descent, or a point obtained by modifying the result of gradient descent. In the polymatrix game setting, it is not immediately obvious how such a modification can be derived with a non-constant number of players (without an exponential blowup). Thus we apply a different analysis, which proves that the point resulting from gradient descent always has our approximation guarantee.

4.4 The Function f and ϵ-Nash Equilibria

As it is explained in Section 4.3, for the `Descent` algorithm we have to define a function $f : \Delta \to [0, 1]$ that computes the maximum regret the players suffer under any strategy profile $\mathbb{X} \in \Delta$.

The regret function $f_i : \Delta \to [0, 1]$ is defined, for each player i, as follows:

$$f_i(\mathbb{X}) := u_i^*(\mathbb{X}) - u_i(\mathbb{X}). \tag{4.3}$$

The maximum regret under a strategy profile $\mathbb{X}$ is given by the function $f(\mathbb{X})$ where:

$$f(\mathbb{X}) := \max\{f_1(\mathbb{X}), \ldots, f_n(\mathbb{X})\}. \tag{4.4}$$

Notice that the value of $f(\mathbb{X})$ is essentially the approximation guarantee of the strategy profile $\mathbb{X}$. Hence, $\mathbb{X}$ is an ϵ-approximate Nash equilibrium (ϵ-NE) if:

$$f(\mathbb{X}) \leq \epsilon,$$

and $\mathbb{X}$ is an *exact* Nash equilibrium if $f(\mathbb{X}) = 0$.

4.5 The Gradient

The goal is to apply gradient descent to the regret function f. In this section, we formally define the gradient of f in Definition 4, and give a combinatorial version of that definition in Lemma 5. In order to show that our gradient descent method terminates after a polynomial number of iterations, we actually need to

use a slightly modified version, which we describe at the end of this section in Definition 7.

Given a point $\mathbb{X} \in \Delta$, a *feasible direction* from $\mathbb{X}$ is defined by any other point $\mathbb{X}' \in \Delta$. This defines a line between $\mathbb{X}$ and $\mathbb{X}'$, and formally speaking, the direction of this line is $\mathbb{X}' - \mathbb{X}$. In order to define the gradient of this direction, we consider the function $f((1-\epsilon)\cdot\mathbb{X} + \epsilon\cdot\mathbb{X}') - f(\mathbb{X})$ where ϵ lies in the range $0 \leq \epsilon \leq 1$. The gradient of this direction is given in the following definition.

Definition 4. *Given profiles $\mathbb{X}, \mathbb{X}' \in \Delta$ and $\epsilon \in [0,1]$, we define:*

$$Df(\mathbb{X}, \mathbb{X}', \epsilon) := f((1-\epsilon)\cdot\mathbb{X} + \epsilon\cdot\mathbb{X}') - f(\mathbb{X}).$$

Then, we define the gradient of f at $\mathbb{X}$ in the direction $\mathbb{X}' - \mathbb{X}$ as:

$$Df(\mathbb{X}, \mathbb{X}') = \lim_{\epsilon\to 0} \frac{1}{\epsilon} Df(\mathbb{X}, \mathbb{X}', \epsilon). \tag{4.5}$$

The gradient of f at any point $\mathbb{X} \in \Delta$ along a feasible direction specified by another point $\mathbb{X}' \in \Delta$ provides the rate of decrease, or increase, of the value of f along that direction. At any point $\mathbb{X}$ we wish to find the direction such that f decreases with the highest rate, that is, we want to find the point $\mathbb{X}'$ that minimizes $Df(\mathbb{X}, \mathbb{X}')$, and move along the direction $\mathbb{X}' - \mathbb{X}$, or to find that $\mathbb{X}$ is a stationary point, i.e. $Df(\mathbb{X}, \mathbb{X}') \geq 0$ for all $\mathbb{X}' \in \Delta$. Unfortunately, Equation (4.5) cannot be used directly in a combinatorial algorithm. Instead, in Definition 5 we provide a combinatorial version of the gradient that allows us to compute the steepest descent direction, with respect to the combinatorial gradient, via a linear program.

The intuition for the combinatorial version comes from Equation (4.5). Let us define $\bar{\mathbb{X}} := (1-\epsilon)\cdot\mathbb{X} + \epsilon\cdot\mathbb{X}'$. From the natural gradient defined in Definition 4, we get that:

$$\begin{aligned} Df(\mathbb{X}, \mathbb{X}') &= \lim_{\epsilon\to 0} \frac{1}{\epsilon}\big(f(\bar{\mathbb{X}}) - f(\mathbb{X})\big) \\ &= \lim_{\epsilon\to 0} \frac{1}{\epsilon}\left(\max_{i\in[n]} f_i(\bar{\mathbb{X}}) - f(\mathbb{X})\right) \\ &= \max_{i\in[n]}\left(\lim_{\epsilon\to 0} \frac{1}{\epsilon}\big(f_i(\bar{\mathbb{X}}) - f(\mathbb{X})\big)\right). \end{aligned} \tag{4.6}$$

In the Section 4.8.1 we study the limit $\lim_{\epsilon\to 0} \frac{1}{\epsilon}\big(f_i(\bar{\mathbb{X}}) - f(\mathbb{X})\big)$, and we prove that it is equal to the following combinatorial version. Before we state the result we introduce some useful notation. Given profiles $\mathbb{X}$ and $\mathbb{X}'$ let us denote:

$$Df_i(\mathbb{X}, \mathbb{X}') = \max_{k\in \mathrm{Br}_i(\mathbb{X})} \big\{\big(v_i(\mathbb{X}')\big)_k\big\} - u_i(x_i, \mathbb{X}') + u_i(x_i - x_i', \mathbb{X}). \tag{4.7}$$

The above expression arises from expanding $f_i(\bar{\mathbb{X}}) - f(\mathbb{X})$. The terms above are all multiplied by ϵ in the expansion, whereas the remaining terms all tend to zero when the limit is taken. The following lemma is proved in the Section 4.8.1:

Lemma 17. *Let $\mathbb{X}$ be strategy profile and $i \in [n]$. If $f_i(\mathbb{X}) = f(\mathbb{X})$, then:*

$$\lim_{\epsilon \to 0} \frac{1}{\epsilon}\big(f_i(\bar{\mathbb{X}}) - f(\mathbb{X})\big) = Df_i(\mathbb{X}, \mathbb{X}') - f(\mathbb{X}).$$

otherwise $\lim_{\epsilon \to 0} \frac{1}{\epsilon}\big(f_i(\bar{\mathbb{X}}) - f(\mathbb{X})\big) = -\infty$.

Combining Equation (4.6) with Lemma 17 gives the following combinatorial version of the gradient that we will use throughout the rest of the chapter.

Definition 5 (Combinatorial gradient)**.** *The gradient of f at point $\mathbb{X}$ along direction $\mathbb{X}' - \mathbb{X}$ is:*

$$Df(\mathbb{X}, \mathbb{X}') = \max_{i \in [n]} Df_i(\mathbb{X}, \mathbb{X}') - f(\mathbb{X}).$$

In order to show that our gradient descent algorithm terminates after a polynomial number of steps, we have to use a slight modification of the formula given in Definition 5. More precisely, in $Df_i(\mathbb{X}, \mathbb{X}')$, we need to take the maximum over the δ-best responses, rather than the best responses.

We begin by providing the definition of the δ-best responses.

Definition 6 (δ-best response)**.** *Let $\mathbb{X} \in \Delta$, and let $\delta \in (0, 0.5]$. The δ-best response set $\mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})$ for player $i \in [n]$ is defined as:*

$$\mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x}) := \left\{ j \in [m_i] : \big(v_i(\mathbb{X})\big)_j \geq u_i^*(\mathbb{X}) - \delta \right\}.$$

We now define the function $Df_i^{\delta}(\mathbb{X}, \mathbb{X}')$.

Definition 7. *Let $\mathbb{X}, \mathbb{X}' \in \Delta$, let $\epsilon \in [0, 1]$, and let $\delta \in (0, 0.5]$. We define $Df_i^{\delta}(\mathbb{X}, \mathbb{X}')$ as:*

$$Df_i^{\delta}(\mathbb{X}, \mathbb{X}') := \max_{k \in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})} \left\{ \big(v_i(\mathbb{X}')\big)_k \right\} - u_i(x_i, \mathbb{X}') - u_i(x_i', \mathbb{X}) + u_i(x_i, \mathbb{X}). \tag{4.8}$$

Furthermore, we define $Df^{\delta}(\mathbb{X}, \mathbb{X}')$ as:

$$Df^{\delta}(\mathbb{X}, \mathbb{X}') = \max_{i \in [n]} Df_i^{\delta}(\mathbb{X}, \mathbb{X}') - f(\mathbb{X}). \tag{4.9}$$

Our algorithm works by performing gradient descent using the function Df^δ as the gradient. Obviously, this is a different function to Df, and so we are not actually performing gradient descent on the gradient of f. It is important to note that all of our proofs are in terms of Df^δ, and so this does not affect the correctness of our algorithm. We prove Lemma 17 in order to explain where our definition of the combinatorial gradient comes from, but the correctness of our algorithm does not depend on the correctness of Lemma 17. Furthermore, we will use some of the techniques used in the proof of Lemma 17 in order to prove the running time of our algorithm.

4.6 The Algorithm

In this section, we describe our algorithm for finding a $(0.5+\delta)$-Nash equilibrium in a polymatrix game by gradient descent. In each iteration of the algorithm, we must find the *direction* of steepest descent with respect to Df^δ.

The direction of steepest descent. We show that the direction of steepest descent can be found by solving a linear program. Our goal is, for a given strategy profile $\mathbb{X}$, to find another strategy profile $\mathbb{X}'$ so as to minimize the gradient $Df^\delta(\mathbb{X}, \mathbb{X}')$. Recall that Df^δ is defined in Equation (4.9) to be:

$$Df^\delta(\mathbb{X}, \mathbb{X}') = \max_{i\in[n]} Df_i^\delta(\mathbb{X}, \mathbb{X}') - f(\mathbb{X}).$$

Note that the term $f(\mathbb{X})$ is a constant with respect to $\mathbb{X}'$ in this expression, because it is the same for all directions $\mathbb{X}'$. Thus, it is sufficient to formulate a linear program in order to find the $\mathbb{X}'$ that minimizes $\max_{i\in[n]} Df_i^\delta(\mathbb{X}, \mathbb{X}')$. Using the definition of Df_i^δ in Equation (4.8), we can do this as follows.

Definition 8 (Steepest descent linear program)**.** *Given a strategy profile $\mathbb{X}$, the* steepest descent linear program *is defined as follows. Find $\mathbb{X}' \in \Delta$, $l_1, l_2, \ldots, l_n$, and w such that:*

$$\begin{array}{lll}
\textit{minimize} & w & \\
\textit{subject to} & \big(v_i(\mathbb{X}')\big)_k \le l_i & \forall k \in \mathrm{Br}_{\mathrm{i}}^\delta(\mathbf{x}), \qquad \forall i \in [n] \\
& l_i - u_i(x_i, \mathbb{X}') - u_i(x_i', \mathbb{X}) + u_i(\mathbb{X}) \le w & \forall i \in [n] \\
& \mathbb{X}' \in \Delta. &
\end{array}$$

The l_i variables deal with the maximum in the term $\max_{k\in\mathrm{Br}_{\mathrm{i}}^\delta(\mathbf{x})}\big\{\big(v_i(\mathbb{X}')\big)_k\big\}$, while the variable w is used to deal with the maximum over the functions Df_i^δ.

Since the constraints of the linear program correspond precisely to the definition of Df^{δ}, it is clear that, when we minimize w, the resulting $\mathbb{X}'$ specifies the direction of steepest descent. For each profile $\mathbb{X}$, we define $Q(\mathbb{X})$ to be the direction $\mathbb{X}'$ found by the steepest descent LP for $\mathbb{X}$.

Once we have found the direction of steepest descent, we then need to move in that direction. More precisely, we fix a parameter δ and we define $\epsilon = \frac{\delta}{\delta+2}$ which is used to determine how far we move in the steepest descent direction. The choice of this value for ϵ ensures that in every iteration of our algorithm the value of f is decreasing and moreover, as we will show in Section 4.8, leads to a polynomial bound on the running time of our algorithm.

The algorithm. We can now formally describe our algorithm. The algorithm takes a parameter $\delta \in (0, 0.5]$, which will be used as a tradeoff between running time and the quality of approximation.

`Descent` algorithm

1. Choose an arbitrary strategy profile $\mathbb{X} \in \Delta$.
2. Solve the steepest descent linear program with input $\mathbb{X}$ to obtain $\mathbb{X}' = Q(\mathbb{X})$.
3. Set $\mathbb{X} := \mathbb{X} + \epsilon(\mathbb{X}' - \mathbb{X})$, where $\epsilon = \frac{\delta}{\delta+2}$.
4. If $f(\mathbb{X}) \leq 0.5 + \delta$ then stop, otherwise go to step 2.

A single iteration of this algorithm corresponds to executing steps 2, 3, and 4. Since this only involves solving a single linear program, it is clear that each iteration can be completed in polynomial time.

The rest of this chapter is dedicated to showing the following theorem.

Theorem 7. *Algorithm 1 finds a $(0.5 + \delta)$-NE after at most $O(\frac{1}{\delta^2})$ iterations.*

To prove Theorem 7, we will show two properties. Firstly, in Section 4.7, we show that our gradient descent algorithm never gets stuck in a stationary point before it finds a $(0.5 + \delta)$-NE. To do so, we define the notion of a *δ-stationary point*, and we show that every δ-stationary point is at least a $(0.5+\delta)$-NE, which then directly implies that the gradient descent algorithm will not get stuck before it finds a $(0.5 + \delta)$-NE.

Secondly, in Section 4.8, we prove the upper bound on the number of iterations. To do this we show that, if an iteration of the algorithm starts at a point

that is not a δ-stationary point, then that iteration will make a large enough amount of progress. This then allows us to show that the algorithm will find a $(0.5+\delta)$-NE after $O(\frac{1}{\delta^2})$ many iterations, and therefore the overall running time of the algorithm is polynomial.

4.7 Stationary Points of f

Recall that Definition 8 gives a linear program for finding the direction $\mathbb{X}'$ that minimises $Df^{\delta}(\mathbb{X}, \mathbb{X}')$. Our steepest descent procedure is able to make progress whenever this gradient is negative, and so a stationary point is any point $\mathbb{X}$ for which $Df^{\delta}(\mathbb{X}, \mathbb{X}') \geq 0$ for every $\mathbb{X}'$. In fact, our analysis requires us to consider δ-stationary points, which we now define.

Definition 9 (δ-stationary point)**.** *Let $\mathbb{X}^*$ be a mixed strategy profile, and let $\delta > 0$. We have that $\mathbb{X}^*$ is a δ-stationary point if for all $\mathbb{X}' \in \Delta$:*

$$Df^{\delta}(\mathbb{X}^*, \mathbb{X}') \geq -\delta.$$

We now show that every δ-stationary point of $f(\mathbb{X})$ is a $(0.5+\delta)$-NE. Recall from Definition 7 that:

$$Df^{\delta}(\mathbb{X}, \mathbb{X}') = \max_{i \in [n]} Df_i^{\delta}(\mathbb{X}, \mathbb{X}') - f(\mathbb{X}).$$

Therefore, if $\mathbb{X}^*$ is a δ-stationary point, we must have, for every direction $\mathbb{X}'$:

$$f(\mathbb{X}^*) \leq \max_{i \in [n]} Df_i^{\delta}(\mathbb{X}^*, \mathbb{X}') + \delta. \tag{4.10}$$

Since $f(\mathbb{X}^*)$ is the maximum regret under the strategy profile $\mathbb{X}^*$, in order to show that $\mathbb{X}^*$ is a $(0.5+\delta)$-NE, we only have to find some direction $\mathbb{X}'$ such that $\max_{i \in [n]} Df_i^{\delta}(\mathbb{X}^*, \mathbb{X}') \leq 0.5$. We do this in the following lemma.

Lemma 18. *For every point $\mathbb{X}$, there exists a direction $\mathbb{X}'$ such that:*

$$\max_{i \in [n]} Df_i^{\delta}(\mathbb{X}, \mathbb{X}') \leq 0.5.$$

Proof. First, define $\bar{\mathbb{X}}$ to be a strategy profile in which each player $i \in [n]$ plays a best response against $\mathbb{X}$. We will set $\mathbb{X}' = \frac{\bar{\mathbb{X}}+\mathbb{X}}{2}$. Then for each $i \in [n]$, we have

that $Df_i^\delta(\mathbb{X}, \mathbb{X}')$, is less than or equal to:

$$\begin{aligned}
&\max_{k \in \mathrm{Br}_i^\delta(\mathbb{X})} \left\{ \left(v_i(\frac{\bar{\mathbb{X}} + \mathbb{X}}{2})\right)_k \right\} - u_i(x_i, \frac{\bar{\mathbb{X}} + \mathbb{X}}{2}) - u_i(\frac{\bar{x_i} + x_i}{2}, \mathbb{X}) + u_i(x_i, \mathbb{X}) \\
&= \frac{1}{2} \cdot \max_{k \in \mathrm{Br}_i^\delta(\mathbb{X})} \left\{ \left(v_i(\bar{\mathbb{X}} + \mathbb{X})\right)_k \right\} - \frac{1}{2} \cdot u_i(x_i, \bar{\mathbb{X}}) - \frac{1}{2} \cdot u_i(\bar{x_i}, \mathbb{X}) \\
&\leq \frac{1}{2} \cdot \left(\max_{k \in \mathrm{Br}_i^\delta(\mathbb{X})} \left\{ \left(v_i(\bar{\mathbb{X}})\right)_k \right\} + \max_{k \in \mathrm{Br}_i^\delta(\mathbb{X})} \left\{ \left(v_i(\mathbb{X})\right)_k \right\} - u_i(x_i, \bar{\mathbb{X}}) - u_i(\bar{x_i}, \mathbb{X}) \right) \\
&= \frac{1}{2} \cdot \left(\max_{k \in \mathrm{Br}_i^\delta(\mathbb{X})} \left\{ \left(v_i(\bar{\mathbb{X}})\right)_k \right\} - u_i(x_i, \bar{\mathbb{X}}) \right) \quad \text{because } \bar{x_i} \text{ is a b.r. to } x \\
&\leq \frac{1}{2} \cdot \max_{k \in \mathrm{Br}_i^\delta(\mathbb{X})} \left\{ \left(v_i(\bar{\mathbb{X}})\right)_k \right\} \\
&\leq \frac{1}{2}.
\end{aligned}$$

Thus, the point $\mathbb{X}'$ satisfies $\max_{i \in [n]} Df_i^\delta(\mathbb{X}, \mathbb{X}') \leq 0.5$.

□

We can sum up the results of the section in the following lemma.

Lemma 19. *Every δ-stationary point $\mathbb{X}^*$ is a $(0.5 + \delta)$-Nash equilibrium.*

4.8 Time Complexity of the Algorithm

In this section, we show that Algorithm 1 terminates after a polynomial number of iterations. Let $\mathbb{X}$ be a strategy profile that is considered by Algorithm 1, and let $\mathbb{X}' = Q(\mathbb{X})$ be the solution of the steepest descent LP for $\mathbb{X}$. These two profiles will be fixed throughout this section.

We begin by proving a technical lemma that will be crucial for showing our bound on the number of iterations. To simplify our notation, throughout this section we define $f_{new} := f(\mathbb{X} + \epsilon(\mathbb{X}' - \mathbb{X}))$ and $f := f(\mathbb{X})$. Furthermore, we define $\mathcal{D} = \max_{i \in [n]} Df_i^\delta(\mathbb{X}, \mathbb{X}')$. The following lemma, which is proved in Subsection 4.8.2, gives a relationship between f and f_{new}.

Lemma 20. *In every iteration of Algorithm 1 we have:*

$$f_{new} - f \leq \epsilon(\mathcal{D} - f) + \epsilon^2(1 - \mathcal{D}). \tag{4.11}$$

In the next lemma we prove that, if we are not in a δ-stationary point, then we have a bound on the amount of progress made in each iteration. We use this in order to bound the number of iterations needed before we reach a point $\mathbb{X}$ where $f(\mathbb{X}) \leq 0.5 + \delta$.

Lemma 21. *Fix $\epsilon = \frac{\delta}{\delta+2}$, where $0 < \delta \leq 0.5$. Either $\mathbb{X}$ is a δ-stationary point or:*

$$f_{new} \leq \left(1 - \left(\frac{\delta}{\delta+2}\right)^2\right) f. \tag{4.12}$$

Proof. Recall that by Lemma 20 the gain in every iteration of the steepest descent is:

$$f_{new} - f \leq \epsilon(\mathcal{D} - f) + \epsilon^2(1 - \mathcal{D}). \tag{4.13}$$

We consider the following two cases:

a) $\mathcal{D} - f > -\delta$. Then, by definition, we are in a δ-stationary point.

b) $\mathcal{D} - f \leq -\delta$. We have set $\epsilon = \frac{\delta}{\delta+2}$. If we solve for δ we get that $\delta = \frac{2\epsilon}{1-\epsilon}$. Since $\mathcal{D} - f \leq -\delta$, we have that $(\mathcal{D} - f)(1 - \epsilon) \leq -2\epsilon$. Thus we have:

$$\begin{aligned}
(\mathcal{D} - f)(\epsilon - 1) &\geq 2\epsilon \\
0 &\geq (\mathcal{D} - f)(1 - \epsilon) + 2\epsilon \\
0 &\geq (\mathcal{D} - f) + \epsilon(2 - \mathcal{D} + f) \\
-\epsilon f - \epsilon &\geq (\mathcal{D} - f) + \epsilon(1 - \mathcal{D}) \qquad (\epsilon \geq 0) \\
-\epsilon^2 f - \epsilon^2 &\geq \epsilon(\mathcal{D} - f) + \epsilon^2(1 - \mathcal{D}).
\end{aligned}$$

Thus, since $\epsilon^2 \geq 0$ we get:

$$\begin{aligned}
-\epsilon^2 f &\geq \epsilon(\mathcal{D} - f) + \epsilon^2(1 - \mathcal{D}) \\
&\geq f_{new} - f \qquad \text{According to (4.13).}
\end{aligned}$$

Thus we have shown that:

$$\begin{aligned}
f_{new} - f &\leq -\epsilon^2 f \\
f_{new} &\leq (1 - \epsilon^2) f.
\end{aligned}$$

Finally, using the fact that $\epsilon = \frac{\delta}{\delta+2}$, we get that

$$f_{new} \leq \left(1 - \left(\frac{\delta}{\delta+2}\right)^2\right) f.$$

□

So, when the algorithm has not reached yet a δ-stationary point, there is a decrease on the value of f that is at least as large as the bound specified in (4.12) in every iteration of the gradient descent procedure. In the following lemma we prove that after $O(\frac{1}{\delta^2})$ iterations of the steepest descent procedure the algorithm finds a point $\mathbb{X}$ where $f(\mathbb{X}) \leq 0.5 + \delta$.

Lemma 22. *After $O(\frac{1}{\delta^2})$ iterations of the steepest descent procedure the algorithm finds a point $\mathbb{X}$ where $f(\mathbb{X}) \leq 0.5 + \delta$.*

Proof. Let $\mathbb{X}_1, \mathbb{X}_2, \ldots, \mathbb{X}_k$ be the sequence of strategy profiles that are considered by Algorithm 1. Since the algorithm terminates as soon as it finds a $(0.5+\delta)$-NE, we have $f(\mathbb{X}_i) > 0.5 + \delta$ for every $i < k$. Therefore, for each $i < k$ we we can apply Lemma 19 to argue that $\mathbb{X}_i$ is not a δ-stationary point, which then allows us to apply Lemma 21 to obtain:

$$f(\mathbb{X}_{i+1}) \leq \left(1 - \left(\frac{\delta}{\delta+2}\right)^2\right) f(\mathbb{X}_i).$$

So, the amount of progress made by the algorithm in iteration i is:

$$\begin{aligned} f(\mathbb{X}_i) - f(\mathbb{X}_{i+1}) &\geq f(\mathbb{X}_i) - \left(1 - \left(\frac{\delta}{\delta+2}\right)^2\right) f(\mathbb{X}_i) \\ &= \left(\frac{\delta}{\delta+2}\right)^2 f(\mathbb{X}_i) \\ &> \left(\frac{\delta}{\delta+2}\right)^2 \cdot 0.5. \end{aligned}$$

Thus, each iteration of the algorithm decreases the regret by at least $(\frac{\delta}{\delta+2})^2 \cdot 0.5$. The algorithm starts at a point $\mathbb{X}_1$ with $f(\mathbb{X}_1) \leq 1$, and terminates when it reaches a point $\mathbb{X}_k$ with $f(\mathbb{X}_k) \leq 0.5 + \delta$. Thus the total amount of progress made over all iterations of the algorithm can be at most $1 - (0.5 + \delta)$. Therefore, the number of iterations used by the algorithm can be at most:

$$\begin{aligned} \frac{1 - (0.5+\delta)}{\left(\frac{\delta}{\delta+2}\right)^2 \cdot 0.5} &\leq \frac{1 - 0.5}{\left(\frac{\delta}{\delta+2}\right)^2 \cdot 0.5} \\ &= \frac{(\delta+2)^2}{\delta^2} = \frac{\delta^2}{\delta^2} + \frac{4\delta}{\delta^2} + \frac{4}{\delta^2}. \end{aligned}$$

Since $\delta < 1$, we have that the algorithm terminates after at most $O(\frac{1}{\delta^2})$ iterations. □

Lemma 22 implies that that after polynomially many iterations the algorithm finds a point such that $f(\mathbb{X}) \leq 0.5 + \delta$, and by definition such a point is a $(0.5 + \delta)$-NE. Thus we have completed the proof of Theorem 7.

4.8.1 Proof of Lemma 17

Before we begin with the proof, we introduce the following notation. For a player $i \in [n]$, given a strategy profile $\mathbb{X}$ and a subset of i's pure strategies $S \subseteq [m_i]$, we use $M_i(\mathbb{X}, S)$ for taking the maximum of the payoffs of i when the others play according to $\mathbb{X}$, and player i is restricted to pick elements from S:

$$M_i(\mathbb{X}, S) := \max_{k \in S} \big(v_i(\mathbb{X})\big)_k.$$

In order to find the gradient, we have to calculate the variation of f_i along the direction $\mathbb{X}' - \mathbb{X}$, by evaluating $f(\bar{\mathbb{X}})$ for points $\bar{\mathbb{X}}$ of the form

$$\bar{\mathbb{X}} := \mathbb{X} + \epsilon(\mathbb{X}' - \mathbb{X}) = (1-\epsilon) \cdot \mathbb{X} + \epsilon \cdot \mathbb{X}'.$$

Recall from (4.3), that for $\bar{\mathbb{X}} \in \Delta$ we have that $f_i(\bar{\mathbb{X}}) := u_i^*(\bar{\mathbb{X}}) - u_i(\bar{\mathbb{X}})$. In order to rewrite $u_i^*(\bar{\mathbb{X}})$ we introduce notation $\Lambda_i(\mathbb{X}, \mathbb{X}', \epsilon)$ as follows.

Definition 10. *Given* $(\mathbb{X}, \mathbb{X}', \epsilon)$ *and* $S = \mathrm{Br}_i(\mathbb{X})$ *we define* $\Lambda_i(\mathbb{X}, \mathbb{X}', \epsilon)$ *as:*

$$\Lambda_i(\mathbb{X}, \mathbb{X}', \epsilon) := \max\left\{0, \max_{k \in \bar{S}}\{(v_i(\bar{\mathbb{X}}))_k\} - \max_{l \in S}\{(v_i(\bar{\mathbb{X}}))_l\}\right\}. \tag{4.14}$$

In the following technical lemma we provide an expression for $u_i^*(\bar{\mathbb{X}})$. In order to rewrite $u_i^*(\bar{\mathbb{X}})$, we use the following simple observation. Consider a multiset of numbers $\{a_1, \ldots, a_n\}$, and the index sets $S \subseteq [n]$ and $\bar{S} = [n] \setminus S$. We have the following identity:

$$\max\{a_1, \ldots, a_n\} \equiv \max_{j \in S}\{a_j\} + \max\left\{0,\ \max_{k \in \bar{S}}\{a_k\} - \max_{j \in S}\{a_j\}\right\}. \tag{4.15}$$

In the following lemma, we use this identity with $S = \mathrm{Br}_i(\mathbb{X})$ to rewrite $u_i^*(\bar{\mathbb{X}})$.

We use this particular expression for $u_i^*(\bar{\mathbb{X}}))$ because it helps us to compute the limit when ϵ tends to zero. Moreover, the values $\Lambda_i(\mathbb{X}, \mathbb{X}', \epsilon)$ will be used in order to derive the value of ϵ that it is used in our algorithm.

Lemma 23. *Given profiles* $\mathbb{X}$ *and* $\mathbb{X}'$ *in* Δ *and a player* $i \in [n]$*, let* $S = \mathrm{Br}_i(\mathbb{X})$*. We have:*

$$u_i^*((1-\epsilon) \cdot \mathbb{X} + \epsilon \cdot \mathbb{X}')) = (1-\epsilon) \cdot M_i(\mathbb{X}, S) + \epsilon \cdot M_i(\mathbb{X}', S) + \Lambda_i(\mathbb{X}, \mathbb{X}', \epsilon). \tag{4.16}$$

Proof.

$$\begin{aligned}
u_i^*(\bar{\mathbb{X}}) &= u_i^*((1-\epsilon)\cdot\mathbb{X} + \epsilon\cdot\mathbb{X}')) \\
&= \max_{k\in[m_i]} \big\{\big(v_i(\mathbb{X}+\epsilon(\mathbb{X}'-\mathbb{X}))\big)_k\big\} && \text{By (4.2)} \\
&= \max_{k\in S} \big\{\big(v_i(\mathbb{X}+\epsilon(\mathbb{X}'-\mathbb{X}))\big)_k\big\} + \Lambda_i(\mathbb{X},\mathbb{X}',\epsilon) && \text{By (4.15) and (4.14)} \\
&= \max_{k\in S} \big\{\big((1-\epsilon)\cdot v_i(\mathbb{X}) + \epsilon\cdot v_i(\mathbb{X}')\big)_k\big\} + \Lambda_i(\mathbb{X},\mathbb{X}',\epsilon).
\end{aligned}$$

Since $S = \mathrm{Br}_i(\mathbb{X})$, we know that for all $k \in S$ we have that $(v_i(\mathbb{X}))_k$ are equal, so we have the following:

$$\begin{aligned}
&\max_{k\in S} \big\{\big((1-\epsilon)\cdot v_i(\mathbb{X}) + \epsilon\cdot v_i(\mathbb{X}')\big)_k\big\} \\
&\qquad = \max_{k\in S} \big\{\big((1-\epsilon)\cdot v_i(\mathbb{X})\big)_k\big\} + \max_{k\in S} \big\{\big(\epsilon\cdot v_i(\mathbb{X}')\big)_k\big\} \\
&\qquad = (1-\epsilon)\cdot M_i(\mathbb{X},S) + \epsilon\cdot M_i(\mathbb{X}',S)
\end{aligned}$$

and we get the claimed result. □

We will use the expression (4.16) for $u_i^*(\bar{\mathbb{X}})$, along with the following reformulation of $u_i(\bar{\mathbb{X}})$:

$$\begin{aligned}
&u_i(\bar{\mathbb{X}}) = u_i(\mathbb{X}+\epsilon(\mathbb{X}'-\mathbb{X})) \\
&= u_i(x_i+\epsilon(x_i'-x_i), \mathbb{X}+\epsilon(\mathbb{X}'-\mathbb{X})) \\
&= u_i(x_i,\mathbb{X}) + \epsilon\cdot u_i(x_i,\mathbb{X}'-\mathbb{X}) + \epsilon\cdot u_i(x_i'-x_i,\mathbb{X}) + \epsilon^2\cdot u_i(x_i'-x_i,\mathbb{X}'-\mathbb{X}) \\
&= u_i(\mathbb{X}) + \epsilon\cdot u_i(x_i,\mathbb{X}') - \epsilon\cdot u_i(x_i,\mathbb{X}) + \epsilon\cdot u_i(x_i',\mathbb{X}) + \epsilon\cdot u_i(x_i,\mathbb{X}) - \epsilon^2\cdot u_i(\mathbb{X}'-\mathbb{X}) \\
&= (1-\epsilon)\cdot u_i(\mathbb{X}) + \epsilon\big(u_i(x_i,\mathbb{X}') + u_i(x_i',\mathbb{X}) - u_i(\mathbb{X})\big) + \epsilon^2\cdot u_i(\mathbb{X}'-\mathbb{X}). \qquad (4.17)
\end{aligned}$$

We now use these reformulations to prove the following lemma.

Lemma 24. *We have that $f_i(\bar{\mathbb{X}}) - f(\mathbb{X})$ is equal to:*

$$\epsilon\big(Df_i(\mathbb{X},\mathbb{X}') - f(\mathbb{X})\big) + \Lambda_i(\mathbb{X},\mathbb{X}',\epsilon) - \epsilon^2 u_i(\mathbb{X}'-\mathbb{X}) - (1-\epsilon)\max_{j\in[n]}\big\{f_j(\mathbb{X}) - f_i(\mathbb{X})\big\}.$$

Proof. Recall that $S = \mathrm{Br}_i(\mathbb{X})$. For a given $i \in [n]$, using Lemma 23 and the reformulation for $u_i(\bar{\mathbb{X}})$, we have:

$$\begin{aligned}
f_i(\bar{\mathbb{X}}) - f(\mathbb{X}) &= u_i^*(\bar{\mathbb{X}}) - u_i(\bar{\mathbb{X}}) - f(\mathbb{X}) \\
&= (1-\epsilon)\cdot M_i(\mathbb{X},S) + \epsilon\cdot M_i(\mathbb{X}',S) + \Lambda_i(\mathbb{X},\mathbb{X}',\epsilon) \\
&\quad - (1-\epsilon)u_i(\mathbb{X}) + \epsilon\big(-u_i(x_i,\mathbb{X}') - u_i(x_i',\mathbb{X}) + u_i(\mathbb{X})\big) - \epsilon^2 u_i(\mathbb{X}'-\mathbb{X}) - f(\mathbb{X}).
\end{aligned}$$

Recall from (4.3) that $f_i(\mathbb{X}) = M_i(\mathbb{X}, S) - u_i(\mathbb{X})$, so the formula above is equal to:

$$\epsilon\big(M_i(\mathbb{X}', S) - u_i(x_i, \mathbb{X}') - u_i(x_i', \mathbb{X}) + u_i(\mathbb{X})\big) + \Lambda_i(\mathbb{X}, \mathbb{X}', \epsilon) - \epsilon^2 u_i(\mathbb{X}' - \mathbb{X}) + (1-\epsilon) f_i(\mathbb{X}) - f(\mathbb{X}).$$

Now we can use (4.7) for $Df_i(\mathbb{X}, \mathbb{X}')$ so that the above formula becomes:

$$\epsilon \cdot Df_i(\mathbb{X}, \mathbb{X}') + \Lambda_i(\mathbb{X}, \mathbb{X}', \epsilon) - \epsilon^2 u_i(\mathbb{X}' - \mathbb{X}) + (1-\epsilon) f_i(\mathbb{X}) - f(\mathbb{X}) =$$
$$\epsilon \cdot Df_i(\mathbb{X}, \mathbb{X}') + \Lambda_i(\mathbb{X}, \mathbb{X}', \epsilon) - \epsilon^2 u_i(\mathbb{X}' - \mathbb{X}) + (1-\epsilon) f_i(\mathbb{X}) - (1-\epsilon) f(\mathbb{X}) - \epsilon f(\mathbb{X}) =$$
$$\epsilon\big(Df_i(\mathbb{X}, \mathbb{X}') - f(\mathbb{X})\big) + \Lambda_i(\mathbb{X}, \mathbb{X}', \epsilon) - \epsilon^2 u_i(\mathbb{X}' - \mathbb{X}) - (1-\epsilon)\big(f(\mathbb{X}) - f_i(\mathbb{X})\big).$$

Recall now that $f(\mathbb{X}) = \max_{j\in[n]} f_j(\mathbb{X})$. Thus the term $f(\mathbb{X}) - f_i(\mathbb{X})$ can be written as $\max_{j\in[n]}\big\{f_j(\mathbb{X}) - f_i(\mathbb{X})\big\}$. So, the expression above is equivalent to:

$$\epsilon\big(Df_i(\mathbb{X}, \mathbb{X}') - f(\mathbb{X})\big) + \Lambda_i(\mathbb{X}, \mathbb{X}', \epsilon) - \epsilon^2 u_i(\mathbb{X}' - \mathbb{X}) - (1-\epsilon) \max_{j\in[n]}\big\{f_j(\mathbb{X}) - f_i(\mathbb{X})\big\}.$$

□

We will now use Lemma 24 to study the limit $\lim_{\epsilon\to 0}\big(f_i(\bar{\mathbb{X}}) - f(\mathbb{X})\big)$ for all $i \in [n]$. Firstly, we deal with $\Lambda(\mathbb{X}, \mathbb{X}', \epsilon)$. It is easy to see that $\lim_{\epsilon\to 0}\big(\mathbb{X} + \epsilon(\mathbb{X}' - \mathbb{X})\big) = \mathbb{X}$. Then, when $S = \mathrm{Br}_i(\mathbb{X})$ we have that:

$$\lim_{\epsilon\to 0}\left(\max_{k\in\bar{S}}\{(v_i(\bar{\mathbb{X}}))_k\} - \max_{l\in S}\{(v_i(\bar{\mathbb{X}}))_l\}\right) < 0.$$

This is true from the definition of pure best response strategies. So, from Equation (4.14) for $\Lambda_i(\mathbb{X}, \mathbb{X}', \epsilon)$ it is true that $\lim_{\epsilon\to 0}\Lambda_i(\mathbb{X}, \mathbb{X}', \epsilon) = 0$.

Furthermore, the term $\epsilon^2 \cdot u_i(\mathbb{X}' - \mathbb{X})$ when is divided by ϵ equals to $\epsilon \cdot u_i(\mathbb{X}' - \mathbb{X})$, thus $\lim_{\epsilon\to 0}\big(\epsilon \cdot u_i(\mathbb{X}' - \mathbb{X})\big) = 0$.

Moreover, the term:

$$\lim_{\epsilon\to 0}\left(-\frac{1-\epsilon}{\epsilon} \cdot \max_{j\in[n]}\big\{f_j(\mathbb{X}) - f_i(\mathbb{X})\big\}\right)$$

is either 0 when $f_i(\mathbb{X}) = f(\mathbb{X})$, i.e player i has the maximum regret and $\max_{j\in[n]}\big\{f_j(\mathbb{X}) - f_i(\mathbb{X})\big\} = 0$, or $-\infty$ otherwise, because $\max_{j\in[n]}\big\{f_j(\mathbb{X}) - f_i(\mathbb{X})\big\} > 0$.

To sum up, if $f_i(\mathbb{X})$ achieves the maximum regret at point $\mathbb{X}'$, then the limit $\lim_{\epsilon\to 0}\big(f_i(\bar{\mathbb{X}}) - f(\mathbb{X})\big) = Df_i(\mathbb{X}, \mathbb{X}') - f(\mathbb{X})$, otherwise the limit equals $-\infty$. This completes the proof of Lemma 17.

4.8.2 Proof of Lemma 20

Throughout this proof, $\mathbb{X}, \mathbb{X}', \bar{\mathbb{X}}$, and ϵ will be fixed as they are defined in Section 4.8. In order to prove this lemma, we must show a bound on:

$$f(\bar{\mathbb{X}}) - f(\mathbb{X}) = \max_{i \in [n]} f_i(\bar{\mathbb{X}}) - f(\mathbb{X}).$$

Before we start the analysis we need to redefine the term $\Lambda_i^\delta(\mathbb{X}, \mathbb{X}', \epsilon)$ in order to prove an analogous version of Lemma 23 when δ-best responses are used.

Definition 11. *We define* $\Lambda_i^\delta(\mathbb{X}, \mathbb{X}', \epsilon)$ *as:*

$$\Lambda_i^\delta(\mathbb{X}, \mathbb{X}', \epsilon) := \max\left\{0, \max_{k \in \overline{\mathrm{Br}_{\mathrm{i}}^\delta(\mathbf{x})}} \{(v_i(\bar{\mathbb{X}}))_k\} - \max_{l \in \mathrm{Br}_{\mathrm{i}}^\delta(\mathbf{x})} \{(v_i(\bar{\mathbb{X}}))_l\}\right\}. \tag{4.18}$$

We now use this definition to prove the following lemma.

Lemma 25. *We have:*

$$u_i^*((1-\epsilon)\cdot\mathbb{X} + \epsilon\cdot\mathbb{X}')) \leq (1-\epsilon) \max_{k \in \mathrm{Br}_{\mathrm{i}}^\delta(\mathbf{x})} \big(v_i(\mathbb{X}))_k + \epsilon \max_{k \in \mathrm{Br}_{\mathrm{i}}^\delta(\mathbf{x})} \big(v_i(\mathbb{X}')\big)_k + \Lambda_i^\delta(\mathbb{X}, \mathbb{X}', \epsilon). \tag{4.19}$$

Proof. We have:

$$\begin{aligned} u_i^*((1-\epsilon)\cdot\mathbb{X} + \epsilon\cdot\mathbb{X}')) &= \max_{k \in [m_i]} \big(v_i((1-\epsilon)\cdot\mathbb{X} + \epsilon\cdot\mathbb{X}')\big)_k \\ &= \max_{k \in \mathrm{Br}_{\mathrm{i}}^\delta(\mathbf{x})} \big(v_i((1-\epsilon)\cdot\mathbb{X} + \epsilon\cdot\mathbb{X}')\big)_k + \Lambda_i^\delta(\mathbb{X}, \mathbb{X}', \epsilon) \quad \text{Using (4.15)} \\ &\leq (1-\epsilon) \max_{k \in \mathrm{Br}_{\mathrm{i}}^\delta(\mathbf{x})} \big(v_i(\mathbb{X})\big)_k + \epsilon \max_{k \in \mathrm{Br}_{\mathrm{i}}^\delta(\mathbf{x})} \big(v_i(\mathbb{X}')\big)_k + \Lambda_i^\delta(\mathbb{X}, \mathbb{X}', \epsilon). \end{aligned}$$

□

We will use the reformulation from Equation (4.17) for $u_i(\bar{\mathbb{X}})$:

$$u_i(\bar{\mathbb{X}}) = (1-\epsilon)\cdot u_i(\mathbb{X}) + \epsilon\big(u_i(x_i, \mathbb{X}') + u_i(x_i', \mathbb{X}) - u_i(\mathbb{X})\big) + \epsilon^2 \cdot u_i(\mathbb{X}' - \mathbb{X}). \tag{4.20}$$

The correctness of this was proved in Section 4.8.1. Now we use all the these reformulations in order to prove the following lemma.

Lemma 26. *We have that* $f_i(\bar{\mathbb{X}}) - f(\mathbb{X})$ *is less than or equal to:*

$$\epsilon\big(Df_i^\delta(\mathbb{X}, \mathbb{X}') - f(\mathbb{X})\big) + \Lambda_i^\delta(\mathbb{X}, \mathbb{X}', \epsilon) - \epsilon^2 u_i(\mathbb{X}' - \mathbb{X}) - (1-\epsilon) \max_{j \in [n]} \{f_j - f_i\}. \tag{4.21}$$

Proof. Recall that, by definition, we have that:

$$f_i(\bar{\mathbb{X}}) = u_i^*(\bar{\mathbb{X}}) - u_i(\bar{\mathbb{X}}).$$

Thus, we can apply Lemma 25 along with the reformulation given in Equation (4.20) for $u_i(\bar{\mathbb{X}})$ to prove that $f_i(\bar{\mathbb{X}}) - f(\mathbb{X})$ is less than or equal to:

$$\begin{aligned}(1-\epsilon)\max_{k\in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\big(v_i(\mathbb{X})\big)_k + \epsilon \max_{k\in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\big(v_i(\mathbb{X}')\big)_k + \Lambda_i^{\delta}(\mathbb{X},\mathbb{X}',\epsilon)\\ - (1-\epsilon)u_i(\mathbb{X}) + \epsilon\big(-u_i(x_i,\mathbb{X}') - u_i(x_i',\mathbb{X}) + u_i(\mathbb{X})\big) - \epsilon^2 u_i(\mathbb{X}'-\mathbb{X}) - f(\mathbb{X}).\end{aligned}$$

We can now use the fact that $\max_{k\in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\big(v_i(\mathbb{X})\big)_k - u_i(\mathbb{X}) = f_i(\mathbb{X})$ and the definition of $Df_i^{\delta}(\mathbb{X},\mathbb{X}')$ given in (4.8) to prove that the expression above is equivalent to:

$$\begin{aligned}&\epsilon\cdot Df_i^{\delta}(\mathbb{X},\mathbb{X}') + \Lambda_i^{\delta}(\mathbb{X},\mathbb{X}',\epsilon) - \epsilon^2 u_i(\mathbb{X}'-\mathbb{X}) + (1-\epsilon)f_i(\mathbb{X}) - f(\mathbb{X})\\ &= \epsilon\cdot Df_i^{\delta}(\mathbb{X},\mathbb{X}') + \Lambda_i^{\delta}(\mathbb{X},\mathbb{X}',\epsilon) - \epsilon^2 u_i(\mathbb{X}'-\mathbb{X}) + (1-\epsilon)f_i(\mathbb{X}) - (1-\epsilon)f(\mathbb{X}) - \epsilon f(\mathbb{X})\\ &= \epsilon\big(Df_i^{\delta}(\mathbb{X},\mathbb{X}') - f(\mathbb{X})\big) + \Lambda_i^{\delta}(\mathbb{X},\mathbb{X}',\epsilon) - \epsilon^2 u_i(\mathbb{X}'-\mathbb{X}) - (1-\epsilon)\big(f(\mathbb{X}) - f_i(\mathbb{X})\big)\\ &= \epsilon\big(Df_i^{\delta}(\mathbb{X},\mathbb{X}') - f(\mathbb{X})\big) + \Lambda_i^{\delta}(\mathbb{X},\mathbb{X}',\epsilon) - \epsilon^2 u_i(\mathbb{X}'-\mathbb{X}) - (1-\epsilon)\max_{j\in[n]}\big\{f_j(\mathbb{X}) - f_i(\mathbb{X})\big\}.\end{aligned}$$

This completes the proof. □

Having shown Lemma 26, we will now study each term of (4.21) and provide bounds for each of them. To begin with, it is easy to see that for all $i \in [n]$ we have that $\max_{j\in[n]}\big\{f_j(\mathbb{X}) - f_i(\mathbb{X})\big\} \geq 0$, and since $\epsilon < 1$, we have that $(1-\epsilon)\max_{j\in[n]}\big\{f_j(\mathbb{X}) - f_i(\mathbb{X})\big\} \geq 0$. Thus, Equation (4.21) is less than or equal to:

$$\epsilon\big(Df_i^{\delta}(\mathbb{X},\mathbb{X}') - f(\mathbb{X})\big) + \Lambda_i^{\delta}(\mathbb{X},\mathbb{X}',\epsilon) - \epsilon^2 u_i(\mathbb{X}'-\mathbb{X}). \tag{4.22}$$

Next we consider the term $\Lambda_i^{\delta}(\mathbb{X},\mathbb{X}',\epsilon)$. In the following technical lemma we prove that $\Lambda_i^{\delta}(\mathbb{X},\mathbb{X}',\epsilon) = 0$ for all $i \in [n]$.

Lemma 27. *We have $\Lambda_i^{\delta}(\mathbb{X},\mathbb{X}',\epsilon) = 0$ for all $i \in [n]$.*

Proof. According to equation (4.18) for $\Lambda_i^{\delta}(\mathbb{X},\mathbb{X}',\epsilon)$, we have:

$$\Lambda_i^{\delta}(\mathbb{X},\mathbb{X}',\epsilon) = \max\left\{0, \max_{k\in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\{(v_i(\bar{\mathbb{X}}))_k\} - \max_{l\in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\{(v_i(\bar{\mathbb{X}}))_l\}\right\}.$$

We can rewrite this expression as follows. First define:

$$Z(\mathbb{X},\mathbb{X}',\epsilon,k) = (v_i(\bar{\mathbb{X}}))_k - \max_{l\in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\{(v_i(\bar{\mathbb{X}}))_l\}.$$

Then we have:

$$\Lambda_i^{\delta}(\mathbb{X}, \mathbb{X}', \epsilon) = \max\left\{0, \max_{k \in \overline{\mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}}\left\{Z(\mathbb{X}, \mathbb{X}', \epsilon, k)\right\}\right\}.$$

Our goal is to show that, for our chosen value of ϵ, we have $\Lambda_i^{\delta}(\mathbb{X}, \mathbb{X}', \epsilon) = 0$. For this to be the case, we must have that $Z(\mathbb{X}, \mathbb{X}', \epsilon, k) \leq 0$ for all $k \in \overline{\mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}$. In the rest of this proof, we will show that this is indeed the case.

By definition, we have that:

$$(v_i(\bar{\mathbb{X}}))_k = \big(v_i(\mathbb{X}) + \epsilon(v_i(\mathbb{X}') - v_i(\mathbb{X}))\big)_k. \tag{4.23}$$

The term $\max_{l \in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\{(v_i(\bar{\mathbb{X}}))_l\}$ can be written as follows:

$$\begin{aligned}
\max_{l \in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\{(v_i((1-\epsilon)\mathbb{X} + \epsilon\mathbb{X}'))_l\} &\geq \max_{l \in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\{(v_i((1-\epsilon)\mathbb{X}))_l\} \\
&= (1-\epsilon) \cdot \max_{l \in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\{(v_i(\mathbb{X}))_l\} \\
&= \max_{l \in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\{(v_i(\mathbb{X}))_l\} - \epsilon \cdot \max_{l \in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\{(v_i(\mathbb{X}))_l\}. \quad (4.24)
\end{aligned}$$

We now substitute these two bounds into the definition of $Z(\mathbb{X}, \mathbb{X}', \epsilon, k)$. We have:

$$Z(\mathbb{X}, \mathbb{X}', \epsilon, k) \leq v_i(\mathbb{X})_k - \max_{l \in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\{(v_i(\mathbb{X}))_l\} + \epsilon\left(v_i(\mathbb{X}')_k - v_i(\mathbb{X})_k + \max_{l \in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\{(v_i(\mathbb{X}))_l\}\right). \tag{4.25}$$

From the definition of δ-best responses (Definition 6), we know that for all $k \in \overline{\mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}$:

$$v_i(\mathbb{X})_k - \max_{l \in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\{(v_i(\mathbb{X}))_l\} < -\delta.$$

Furthermore, since we know that the maximum payoff for player $i \in [n]$ is 1, we have the following trivial bound for all $k \in \overline{\mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}$:

$$v_i(\mathbb{X}')_k - v_i(\mathbb{X})_k + \max_{l \in \mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}\{(v_i(\mathbb{X}))_l\} \leq 2.$$

Substituting these two bounds into Equation (4.25) gives, for all $k \in \overline{\mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}$:

$$Z(\mathbb{X}, \mathbb{X}', \epsilon, k) \leq -\delta + \epsilon \cdot 2.$$

Thus, for each $k \in \overline{\mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}$, we have that $Z(\mathbb{X}, \mathbb{X}', \epsilon, k) \leq 0$ whenever:

$$-\delta + \epsilon \cdot 2 \leq 0,$$

and this is equivalent to:

$$\epsilon \leq \frac{\delta}{2}.$$

This inequality holds by the definition of ϵ, so we have $Z(\mathbb{X}, \mathbb{X}', \epsilon, k) \leq 0$ for all $k \in \overline{\mathrm{Br}_{\mathrm{i}}^{\delta}(\mathbf{x})}$, which then implies that $\Lambda_i^{\delta}(\mathbb{X}, \mathbb{X}', \epsilon) \leq 0$. □

Next we consider the term $u_i(\mathbb{X}'-\mathbb{X})$ in Equation (4.22). The following lemma provides a simple lower bound for this term.

Lemma 28. *For all $i \in [n]$, we have $Df_i^\delta(\mathbb{X}, \mathbb{X}') - 1 \leq u_i(\mathbb{X}' - \mathbb{X})$.*

Proof. For $u_i(\mathbb{X}' - \mathbb{X})$ we have the following:

$$\begin{aligned} u_i(\mathbb{X}' - \mathbb{X}) &= u_i(x_i' - x_i, \mathbb{X}' - \mathbb{X}) \\ &= u_i(x_i', \mathbb{X}' - \mathbb{X}) - u_i(x_i, \mathbb{X}' - \mathbb{X}) \\ &= u_i(x_i', \mathbb{X}') - u_i(x_i', \mathbb{X}) - u_i(x_i, \mathbb{X}') + u_i(x_i, \mathbb{X}). \end{aligned} \tag{4.26}$$

Recall from (4.8) that:

$$Df_i^\delta(\mathbb{X}, \mathbb{X}') = \max_{k \in \mathrm{Br}_{\mathrm{i}}^\delta(\mathbf{x})} \left\{ \left(v_i(\mathbb{X}') \right)_k \right\} - u_i(x_i, \mathbb{X}') - u_i(x_i', \mathbb{X}) + u_i(x_i, \mathbb{X}).$$

We can see that (4.26) and (4.8) differ only in terms $u_i(x_i', \mathbb{X}')$ and $\max_{k \in \mathrm{Br}_{\mathrm{i}}^\delta(\mathbf{x})} \left\{ \left(v_i(\mathbb{X}') \right)_k \right\}$ respectively. We know that $\max_{k \in \mathrm{Br}_{\mathrm{i}}^\delta(\mathbf{x})} \left\{ \left(v_i(\mathbb{X}') \right)_k \right\} \leq 1$. Then, we can see that $Df_i^\delta(\mathbb{X}, \mathbb{X}') - 1 \leq u_i(\mathbb{X}' - \mathbb{X})$. □

Recall that $\mathcal{D} = \max_{i \in [n]} Df_i^\delta(\mathbb{X}, \mathbb{X}')$ and $f_{new} = f(\bar{\mathbb{X}})$ and $f = f(\mathbb{X})$. We can now apply the bounds from Lemma 27 and Lemma 28 to Equation (4.22) to obtain:

$$\begin{aligned} f_{new} - f &\leq \max_{i \in [n]} \left\{ \epsilon \big(Df_i^\delta(\mathbb{X}, \mathbb{X}') - f(\mathbb{X}) \big) - \epsilon^2 \big(Df_i^\delta(\mathbb{X}, \mathbb{X}') - 1 \big) \right\} \\ &\leq \max_{i \in [n]} \left\{ \epsilon \big(Df_i^\delta(\mathbb{X}, \mathbb{X}') - f(\mathbb{X}) \big) - \epsilon^2 \big(\mathcal{D} - 1 \big) \right\} \\ &= \epsilon(\mathcal{D} - f) + \epsilon^2(1 - \mathcal{D}). \end{aligned}$$

This completes the proof of Lemma 20.

4.9 Open Questions

We have presented a polynomial-time algorithm that finds a $(0.5+\delta)$-Nash equilibrium of a polymatrix game for any $\delta > 0$. Recently it was shown [31] that the performance guarantee that Tsaknakis and Spirakis proved for their algorithm [66] is almost tight. An empirical study of our algorithm [23] showed that `Descent` is fast and computes approximate Nash equilibria with very good accuracy far away from its theoretical guarantee. Though we do not have examples that show that the approximation guarantee is tight for our algorithm, we do not see an obvious approach to prove a better guarantee. The initial choice of strategy profile affects our algorithm, and it is conceivable that one may be able to start the algorithm from an efficiently computable profile with certain properties that allow a better approximation guarantee. One natural special case is when there is a constant number of players, which may allow one to derive new strategy profiles from a stationary point as done by Tsaknakis and Sprirakis [66]. It may also be possible to develop new techniques when the number of pure strategies available to the players is constant, or when the structure of the graph is restricted in some way. For example, in the games arising from two-player Bayesian games, the graph is always bipartite.

In this chapter we considered ϵ-Nash equilibria, which are the most well-studied type of approximate equilibria. However, ϵ-Nash equilibria have a drawback: since they only require that the expected payoff is within ϵ of a pure best response, it is possible that a player could be required to place probability on a strategy that is arbitrarily far from being a best response. Note, it has been shown that there is a PTAS for finding ϵ-WSNE of bimatrix games if and only if there is a PTAS for ϵ-Nash [18, 12]. For n-player games with $n > 2$ there has been very little work on developing algorithms for finding ϵ-WSNE. This is a very interesting direction, both in general and when $n > 2$ is a constant.

Chapter 5

Approximate Equilibria in Two Player Bayesian Games

In this Chapter, we define two-player Bayesian games, and show how our algorithm can be applied in order to efficiently find a $(0.5 + \delta)$-Bayesian Nash equilibrium. A two-player Bayesian game is played between a *row* player and a *column* player. Each player has a set of possible *types*, and at the start of the game, each player is assigned a type by drawing from a known joint probability distribution. Each player learns his type, but not the type of his opponent. Our task is to find an approximate Bayesian Nash equilibrium (BNE).

We show that this can be reduced to the problem of finding an ϵ-NE in a polymatrix game, and therefore our algorithm from Chapter 4 can be used to efficiently find a $(0.5 + \delta)$-BNE of a two-player Bayesian game.

5.1 Two player Bayesian games preliminaries

Payoff matrices. We will use k_1 to denote the number of pure strategies of the row player and k_2 to denote the number of pure strategies of the column player. Furthermore, we will use m to denote the number of types of the row player, and n to denote the number of types of the column player.

For each pair of types $i \in [m]$ and $j \in [n]$, there is a $k_1 \times k_2$ bimatrix game $(R, C)_{ij} := (R_{ij}, C_{ij})$ that is played when the row player has type i and the column player has type j. We assume that all payoffs in every matrix R_{ij} and every matrix C_{ij} lie in the range $[0, 1]$.

Types. The distribution over types is specified by a joint probability distribution: for each pair of types $i \in [m]$ and $j \in [n]$, the probability that the row player is assigned type i and the column player is assigned type j is given by p_{ij}.

Obviously, we have that:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij} = 1.$$

We also define some useful shorthands: for all $i \in [m]$ we denote by p_i^R (p_j^C) the probability that row (column) player has type $i \in [m]$ ($j \in [n]$). Formally:

$$p_i^R = \sum_{j=1}^{n} p_{ij} \qquad \text{for all } i \in [m],$$
$$p_j^C = \sum_{i=1}^{m} p_{ij} \qquad \text{for all } j \in [n].$$

Note that $\sum_{i=1}^{m} p_i^R = \sum_{j=1}^{n} p_j^C = 1$. Furthermore, we denote by $p_i^R(j)$ the conditional probability that type $j \in [n]$ will be chosen for column player given that type i is chosen for row player. Similarly, we define $p_j^C(i)$ for the column player. Formally:

$$p_i^R(j) = \frac{p_{ij}}{p_i^R} \qquad \text{for all } i \in [m],$$
$$p_j^C(i) = \frac{p_{ij}}{p_j^C} \qquad \text{for all } j \in [n].$$

We can see that for given type $t = (i, j)$ we have that $p_{ij} = p_i^R \cdot p_i^R(j) = p_j^C \cdot p_j^C(i)$.

Strategies. In order to play a Bayesian game, each player must specify a strategy for each of their types. Thus, a strategy profile is a pair $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ such that each $x_i \in \Delta_{k_1}$, and where $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ such that each $y_i \in \Delta_{k_2}$. This means that, when the row player gets type $i \in [m]$ and the column player gets type $j \in [n]$, then the game (R_{ij}, C_{ij}) will be played, and the row player will use strategy x_i while the column player will use strategy y_j.

Given a strategy profile $(\mathbf{x}, \mathbf{y})$, we can define the expected payoff to both players (recall that the players are not told their opponent's type).

Definition 12 (Expected payoff)**.** *Given a strategy profile* $(\mathbf{x}, \mathbf{y})$ *and a type* $t = (i, j)$*, the expected payoff for the row player is given by:*

$$u_R(x_i, \mathbf{y}) = \sum_{j=1}^{n} p_i^R(j) \cdot x_i^T R_{ij} y_j,$$
$$= x_i^T \sum_{j=1}^{n} p_i^R(j) \cdot R_{ij} y_j.$$

Similarly, for the column player the expected payoff is:

$$u_C(\mathbf{x}, y_j) = y_j^T \sum_{i=1}^{m} p_j^C(i) \cdot C_{ij}^T x_i.$$

Rescaling. Before we define approximate equilibria for two-player Bayesian games, we first rescale the payoffs. Much like for polymatrix games, rescaling is needed to ensure that an ϵ-approximate equilibrium has a consistent meaning. Our rescaling will ensure that, for every possible pair of types, both player's expected payoff uses the entire range $[0, 1]$.

For each type i of the row player, we use U_R^i to denote the maximum expected payoff for the row player when he has type i, and we use L_R^i to denote the minimum expected payoff for the row player when he has type i. Formally, these are defined to be:

$$U_R^i = \max_{a \in [k_1]} \sum_{j=1}^{n} \max_{b \in [k_2]} \left(p_i^R(j) \cdot R_{ij} \right)_{a,b},$$

$$L_R^i = \min_{a \in [k_1]} \sum_{j=1}^{n} \min_{b \in [k_2]} \left(p_i^R(j) \cdot R_{ij} \right)_{a,b}.$$

Then we apply the transformation $T_R^i(\cdot)$ to every element z of R_{ij}, for all types j of the column player, where:

$$T_R^i(z) := \frac{1}{U_R^i - L_R^i} \cdot \left(z - \frac{L_R^i}{n} \right). \tag{5.1}$$

Similarly, we transform all payoff matrices for the column player using:

$$T_C^j(z) := \frac{1}{U_C^j - L_C^j} \cdot \left(z - \frac{L_C^j}{m} \right), \tag{5.2}$$

where U_C^j and L_C^j are defined symmetrically. Note that, after this transformation has been applied, both player's expected payoffs lie in the range $[0, 1]$. Moreover, the full range is used: there exists a strategy for the column player against which one of the row player's strategies has expected payoff 1, and there exists a strategy for the column player against which one of the row player's strategies has expected payoff 0. From now on we will assume that the payoff matrices have been rescaled in this way.

We can now define approximate Bayesian Nash equilibria for a two-player Bayesian game.

Definition 13 (Approximate Bayes Nash Equilibrium (ϵ-BNE)). *Let* $(\mathbf{x}, \mathbf{y})$ *be a strategy profile. The profile* $(\mathbf{x}, \mathbf{y})$ *is an* ϵ*-BNE iff the following conditions hold:*

$$u_R(x_i, \mathbf{y}) \geq u_R(x_i', \mathbf{y}) - \epsilon \quad \textit{for all } x_i' \in \Delta_{k_1} \quad \textit{for all } i \in [m], \tag{5.3}$$

$$u_C(\mathbf{x}, y_j) \geq u_C(\mathbf{x}, y_j') - \epsilon \quad \textit{for all } y_j' \in \Delta_{k_2} \quad \textit{for all } j \in [n]. \tag{5.4}$$

5.2 Reducing ϵ-BNE to ϵ-NE

In this section we reduce in polynomial time the problem of computing an ϵ-BNE for a two-player Bayesian game $\mathcal{B}$ to the problem of computing an ϵ-NE of a polymatrix game $\mathcal{P}(\mathcal{B})$. We describe the construction of $\mathcal{P}(\mathcal{B})$ and prove that every ϵ-NE for $\mathcal{P}(\mathcal{B})$ maps to an ϵ-BNE of $\mathcal{B}$.

Construction. Let $\mathcal{B}$ be a two-player Bayesian game where the row player has m types and k_1 pure strategies and the column player has n types and k_2 pure strategies. We will construct a polymatrix game $\mathcal{P}(\mathcal{B})$ as follows.

The game has $m + n$ players. We partition the set of players $[m + n]$ into two sets: the set $K = \{1, 2, \ldots, m\}$ will represent the types of the row player in $\mathcal{B}$, while the set $L = \{m + 1, m + 2, \ldots, m + n\}$ will represent the types of the column player in $\mathcal{B}$. The underlying graph that shows the interactions between the players is a complete bipartite graph $G = (K \cup L, E)$, where every player in K (respectively L) plays a bimatrix game with every player in L (respectively K). The bimatrix game played between vertices $v_i \in K$ and $v_j \in L$ is defined to be (R_{ij}^*, C_{ij}^*), where:

$$R_{ij}^* := p_i^R(j) \cdot R_{ij}, \tag{5.5}$$

$$C_{ij}^* := p_j^C(i) \cdot C_{ij}. \tag{5.6}$$

for all $i \in [m]$ and $j \in [n]$.

Observe that, for each player i in the K, the matrices R_{ij}^* all have the same number of rows, and for each player $j \in L$, the matrices C_{ij}^* all have the same number of columns. Thus, $\mathcal{P}(\mathcal{B})$ is a valid polymatrix game. Moreover, we clearly have that $\mathcal{P}(\mathcal{B})$ has the same size as the original game $\mathcal{B}$. Note that, since we have assumed that the Bayesian game has been rescaled, we have that for every player in $\mathcal{P}(\mathcal{B})$ the minimum (maximum) payoff achievable under pure strategy profiles is 0 (1), so no further scaling is needed in order to apply our algorithm.

We can now prove that every ϵ-NE of the polymatrix game is also an ϵ-BNE of the original two-player Bayesian game, which is the main result of this section.

Theorem 8. *Every ϵ-NE of $\mathcal{P}(\mathcal{B})$ is a ϵ-BNE for $\mathcal{B}$.*

Proof. Let $\mathbf{z} = (x_1, \ldots, x_m, y_1, \ldots, y_n)$ be an ϵ-NE for $\mathcal{P}(\mathcal{B})$. This means that no player can gain more than ϵ by unilaterally changing his strategy. We define the strategy profile $(\mathbf{x}, \mathbf{y})$ for $\mathcal{B}$ where $\mathbf{x} = (x_1, \ldots, x_m)$ and $\mathbf{y} = (y_1, \ldots, y_n)$, and we will show that $(\mathbf{x}, \mathbf{y})$ is an ϵ-BNE for $\mathcal{B}$.

Let $i \in K$ be a player. Since, $\mathbf{z}$ is an ϵ-NE of $\mathcal{P}(\mathcal{B})$, we have:

$$u_i(x_i, \mathbf{z}) \geq u_i(x_i', \mathbf{z}) - \epsilon \quad \text{for all } x_i' \in \Delta_{k_1}.$$

By construction, we can see that player i only interacts with the players from L. Hence his payoff can be written as:

$$u_i(x_i, \mathbf{z}) = x_i^T \sum_{j=1}^{n} R_{ij}^* y_j = u^R(x_i, \mathbf{y}).$$

and since we are in an ϵ-NE, we have:

$$u^R(x_i, \mathbf{y}) \geq u^R(x_i', \mathbf{y}) - \epsilon \quad \text{for all } x_i' \in \Delta_{k_1}. \tag{5.7}$$

This is true for all $i \in K$, thus it is true for all $i \in [m]$.

Similarly, every player $j \in L$ interacts only with players form K, thus:

$$u^C(\mathbf{x}, y_j) = y_j^T \sum_{i=1}^{m} (C_{ij}^*)^T x_i.$$

Since we are in an ϵ-NE we have:

$$u^C(\mathbf{x}, y_j) \geq u_C(\mathbf{x}, y_j') - \epsilon \quad \text{for all } y_j' \in \Delta_{k_2}, \tag{5.8}$$

and this is true for all $j \in K$, thus it is true for all $j \in [n]$.

Combining now the fact that Equation (5.7) is true for all $i \in [n]$ and that Equation (5.8) is true for all $j \in [m]$, it is easy to see that the strategy profile $(\mathbf{x}, \mathbf{y})$ is an ϵ-BNE for $\mathcal{B}$. □

A direct corollary from Theorem **??** is that there is a polynomial time algorithm that computes a $(0.5 + \delta)$-approximate Bayes Nash Equilibrium. However, as it is proved in the next section, there is a much simpler algorithm that computes a 0.5-BNE for two player Bayesian games which can be applied in any polymatrix game played on a bipartite graph.

5.3 A Simple Algorithm for 0.5-BNE

In this section we present a simple algorithm for computing a 0.5-BNE for Bayesian two player games. The algorithm is a generalization of the well known DMP technique of Daskalakis, Mehta and Papadimitriou [20]. We will use the reduction presented above that constructs a bipartite polymatrix game $\mathcal{P}(\mathcal{B})$. We will call as *left side* players the players created for the types of the row player and as *right side* players those created for the types of the column one. The crucial property that allows us to generalise the DMP technique is that all players on the one side of the graph can simultaneously play a best response against a strategy profile of the players from the opposite side without affecting the rest of the players from their side.

The algorithm proceeds as follows. Firstly, every player i in the left side picks a (pure) strategy x_i; we call that as left strategy profile and we denote it by $\mathbf{x} = (x_1, \ldots, x_m)$. Then, every player j in the right side computes a (pure) best response y_j against the left strategy profile $\mathbf{x}$; we call this as right strategy profile and we denote it by $\mathbf{y}^* = (y_1, \ldots, y_n)$. Finally, each player i in the left side computes a (pure) best response $\hat{x}_i$ against the right strategy profile $\mathbf{y}^*$. Let $x_i^* = \frac{x_i + \hat{x}_i}{2}$ and let $\mathbf{x}^* = (x_1^*, \ldots, x_n^*)$. The algorithm returns the strategy profile $\mathbf{z} = (\mathbf{x}^*, \mathbf{y}^*)$.

The Algorithm for 0.5-BNE

1. Pick a pure left strategy profile $\mathbf{x} = (x_1, \ldots, x_m)$.
2. For every $i \in [n]$ compute a best response y_i against $\mathbf{x}$.
3. Define $\mathbf{y}^* = (y_1, \ldots, y_n)$ where y_i is a best response against $\mathbf{x}$.
4. Compute a best response $\hat{\mathbf{x}}$ against $\mathbf{y}^*$.
5. Set $\mathbf{x}^* = \frac{\mathbf{x} + \hat{\mathbf{x}}}{2}$.
6. Return the strategy profile $(\mathbf{x}^*, \mathbf{y}^*)$.

Lemma 29. *The strategy profile $(\mathbf{x}^*, \mathbf{y}^*)$ is a 0.5-NE for the game $\mathcal{P}(\mathcal{B})$.*

Proof. We will study the regret each player suffers under the strategy profile $\mathbf{z} = (\mathbf{x}^*, \mathbf{y}^*)$. First, we study the players from the left side. Let i be a player from the left side and let $x_i^* = \frac{x_i + \hat{x}_i}{2}$ be the strategy computed by the algorithm for that player. Since he interacts only with players from the right side, his regret

$\mathcal{R}_i(\mathbf{z})$ is

$$\begin{aligned}
\mathcal{R}_i(\mathbf{z}) &= \max \sum_{j=1}^{n} R^*_{ij} y_j - x^*_i \cdot \sum_{j=1}^{n} R^*_{ij} y_j \\
&= \hat{x_i} \cdot \sum_{j=1}^{n} R^*_{ij} y_j - x^*_i \cdot \sum_{j=1}^{n} R^*_{ij} y_j \quad \text{(since } \hat{x_i} \text{ is a best response against } \mathbf{y}^*\text{)} \\
&= \frac{1}{2}\hat{x_i} \cdot \sum_{j=1}^{n} R^*_{ij} y_j - \frac{1}{2} x_i \cdot \sum_{j=1}^{n} R^*_{ij} y_j \\
&\leq \frac{1}{2}\hat{x_i} \cdot \sum_{j=1}^{n} R^*_{ij} y_j \\
&\leq \frac{1}{2}.
\end{aligned}$$

Thus, every player from the left side suffers regret at most $\frac{1}{2}$. Then we study the regret $\mathcal{R}_j(\mathbf{z})$ a player j from the right side suffers under the profile $\mathbf{z}$.

$$\begin{aligned}
\mathcal{R}_j(\mathbf{z}) &= \max \sum_{i=1}^{m} C^*_{ji} x^*_i - y_j \cdot \sum_{i=1}^{m} C^*_{ji} x^*_i \\
&\leq \frac{1}{2}\left(\max \sum_{i=1}^{m} C^*_{ji} x_i + \max \sum_{i=1}^{m} C^*_{ji} \hat{x_i} \right) - y_j \cdot \sum_{i=1}^{m} C^*_{ji} x^*_i \\
&= \frac{1}{2} \max \sum_{i=1}^{m} C^*_{ji} \hat{x_i} - \frac{1}{2} y_j \cdot \sum_{i=1}^{m} C^*_{ji} \hat{x_i} \quad \text{(since } y_j \text{ is a best response against } \mathbf{x}\text{)} \\
&\leq \frac{1}{2} \max \sum_{i=1}^{m} C^*_{ji} \hat{x_i} \\
&\leq \frac{1}{2}.
\end{aligned}$$

Hence, all players suffer regret at most $\frac{1}{2}$. The claim follows. □

Combining Lemma 29 with Theorem **??** we get the following Corollary.

Corollary 1. *There is a polynomial time algorithm that computes a 0.5-BNE for two player Bayesian games.*

Note that the simple algorithm described above produces a better theoretical approximation guarantee for Bayes Nash equilibria than the `Descent`. However, it is easy to construct a tight for this generalisation of the DMP technique, while we do not know whether the analysis for the `Descent` is tight. Nevertheless, the strategy profile produced by this simple algorithm can be used as the starting point for the `Descent` and increase its running time efficiency.

Chapter 6

Lipschitz Games

In this chapter we study games with Lipschitz continuous utility functions for the players. Our key insight is that Lipschitz continuity of the utility function allows us to provide algorithms for finding approximate equilibria in these games. We first define formally Lipschitz games and explain how they differ from the games we studied so far. Then, we provide efficient algorithms for computing approximate equilibria for several subclasses of Lipschitz games.

6.1 Lipschitz games preliminaries

We start by fixing some notation. Some notions used in this chapter are already been defined, but we redefine them again here for the self-containment of the chapter.

For each positive integer n we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$, we use Δ^n to denote the $(n-1)$-dimensional simplex, and $\|x\|_p^q$ to denote the (p, q)-norm of a vector $x \in \mathbb{R}^d$, i.e. $\|x\|_p^q = (\sum_{i\in[d]} |x_i|^p)^{q/p}$. When $q = 1$, then we will omit it for notation simplicity. Given a set $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$, we use $conv(X)$ to denote the convex hull of X. A vector $y \in conv(X)$ is said to be k-uniform with respect to X if there exists a size k multiset S of $[n]$ such that $y = \frac{1}{k}\sum_{i\in S} x_i$. When X is clear from the context we will simply say that a vector is k-uniform without mentioning that uniformity is with respect to X. We will use the notion of the λ_p-Lipschitz continuity.

Definition 14 (λ_p-Lipschitz)**.** *A function $f : A \to \mathbb{R}$, with $A \subseteq \mathbb{R}^d$ is λ_p-Lipschitz continuous if for every x and y in A, it is true that*

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

Games and strategies. A game with M players can be described by a set of available actions for each player and a utility function for each player that depends both on his chosen action and the actions the rest of the players chose. For each player $i \in [M]$ we use S_i to denote his set of available actions and we call it his strategy space. We will use $x_i \in S_i$ to denote a specific action chosen by player i and we will call it the strategy of player i, we use $\mathbb{X} = (x_1, \ldots, x_M)$ to denote a strategy profile of the game, and we will use $\mathbb{X}_{-i}$ to denote the strategy profile where the player i is excluded, i.e. $\mathbb{X}_{-i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_M)$. We use $T_i(x_i, \mathbb{X}_{-i})$ to denote the utility of player i when he plays the strategy x_i and the rest of the players play according to the strategy profile $\mathbb{X}_{-i}$. A strategy $\hat{x}_i$ is a best response against the strategy profile $\mathbb{X}_{-i}$, if $T_i(\hat{x}_i, \mathbb{X}_{-i}) \geq T_i(x_i, \mathbb{X}_{-i})$ for all $x_i \in S_i$. The regret player i suffers under a strategy profile $\mathbb{X}$ is the difference between the utility of his best response and his utility under $\mathbb{X}$, i.e. $T_i(\hat{x}_i, \mathbb{X}_{-i}) - T_i(x_i, \mathbb{X}_{-i})$.

Solution Concepts. A strategy profile is an equilibrium if no player can increase his utility by unilaterally changing his strategy. A relaxed version of this concept is the approximate equilibrium, or ϵ-equilibrium, in which no player can increase his utility more than ϵ by unilaterally changing his strategy. Formally, a strategy profile $\mathbb{X}$ is an ϵ-equilibrium if for every player i it holds that

$$T_i(x_i, \mathbb{X}_{-i}) \geq T_i(x_i', \mathbb{X}_{-i}) - \epsilon \quad \text{for all } x_i' \in S_i.$$

6.2 Classes of Lipschitz Games

In this section we define the classes of games studied in this chapter. We will study λ_p-Lipschitz games, penalty games, biased games and distance biased games.

6.2.1 λ_p-Lipschitz games

This is a very general class of games, where each player's strategy space is continuous, and represented by a convex set of vectors, and where the only restriction is that the payoff function is λ_p-Lipschitz continuous for some $p \geq 2$. This class is so general that exact equilibria, and even approximate equilibria may not exist.

Formally, an M-player λ_p-Lipschitz game $\mathfrak{L}$ can be defined by the tuple $(M, n, \lambda, p, \gamma, \mathcal{T})$ where:

- the strategy space S_i of player i is the convex hull of at most n vectors $y_1, \ldots, y_n$ in $\mathbb{R}^d$,
- $\mathcal{T}$ is a set of λ_p-Lipschitz continuous functions and each $T_i(\mathbb{X}) \in \mathcal{T}$,

- and γ is a parameter that that intuitively shows how large the strategy space of the players is, formally $\max_{x_i \in S_i} \|x_i\|_p \leq \gamma$ for every $i \in [M]$.

In what follows in this chapter we will assume that the Lipschitz continuity of a game λ_p is bounded by a constant. Observe that for normal form games this is not the case, since there are bimatrix games that are not constant Lipschitz continuous.

6.2.2 Two Player Penalty Games

In these games, the players play a strategic form game, and their utility is the payoff achieved in the game *minus* a penalty. The penalty function can be an arbitrary function that depends on the player's strategy. Formally, a two-player penalty game $\mathcal{P}$ is defined by a tuple $\big(R, C, \mathfrak{f}_r(\mathbf{x}), \mathfrak{f}_c(\mathbf{y})\big)$, where (R, C) is a bimatrix game and $\mathfrak{f}_r(\mathbf{x})$ and $\mathfrak{f}_c(\mathbf{y})$ are the penalty functions for the row and the column player respectively. The utilities for the players under a strategy profile $(\mathbf{x}, \mathbf{y})$, denoted by $T_r(\mathbf{x}, \mathbf{y})$ and $T_c(\mathbf{x}, \mathbf{y})$, are given by $T_r(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T R \mathbf{y} - \mathfrak{f}_r(\mathbf{x})$ and $T_c(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T C \mathbf{y} - \mathfrak{f}_c(\mathbf{y})$. In this chapter we will focus on games with λ-Lipschitz penalty functions and we will use $\mathcal{P}_\lambda$ to denote this set. A special class of penalty games is obtained when $\mathfrak{f}_r(\mathbf{x}) = \mathbf{x}^T\mathbf{x}$ and $\mathfrak{f}_c(\mathbf{y}) = \mathbf{y}^T\mathbf{y}$. We call these games as *inner product* penalty games.

We note that the class of penalty games is not contained in the class of λ_p-Lipschitz games. In order to see that, observe that the "bimatrix" part of the utility function may not be λ_p-Lipschitz continuous for any constant λ_p. So, $T_r(\mathbf{x}, \mathbf{y})$ and $T_c(\mathbf{x}, \mathbf{y})$ are not λ_p-Lipschitz continuous.

6.2.3 Two Player Biased Games

This is a subclass of penalty games, where extra constraints are added to the penalty functions $\mathfrak{f}_r(\mathbf{x})$ and $\mathfrak{f}_c(\mathbf{y})$ of the players. In this class of games there is a *base strategy* and for each player and the penalty they receive is increasing with the distance between the strategy they choose and their base strategy. Formally, the row player has a base strategy $\mathbf{p} \in \Delta^n$, the column player has a base strategy $\mathbf{q}$ and their strictly increasing penalty functions are defined as $\mathfrak{f}_r(\|\mathbf{x} - \mathbf{p}\|_t^s)$ and $\mathfrak{f}_c(\|\mathbf{y} - \mathbf{q}\|_m^l)$ respectively.

6.2.4 Distance Biased Games

This is a special class of biased games where the penalty function is a fraction of the distance between the base strategy of the player and his chosen strategy. Formally, a two player distance biased game $\mathcal{B}$ is defined by a tuple $\big(R, C, \mathfrak{b}_r(\mathbf{x}, \mathbf{p}), \mathfrak{b}_c(\mathbf{y}, \mathbf{q}), d_r, d_c\big)$, where (R, C) is a bimatrix game, $\mathbf{p} \in \Delta^n$ is a base strategy for the row player, $\mathbf{q} \in \Delta^n$ is a base strategy for the column player,

$$\mathfrak{b}_r(\mathbf{x}, \mathbf{p}) = \|\mathbf{x} - \mathbf{p}\|_t^s \quad \text{and} \quad \mathfrak{b}_c(\mathbf{y}, \mathbf{q}) = \|\mathbf{y} - \mathbf{q}\|_m^l$$

are the penalty functions for the row and the column player respectively.

The utilities for the players under a strategy profile $(\mathbf{x}, \mathbf{y})$, denoted by $T_r(\mathbf{x}, \mathbf{y})$ and $T_c(\mathbf{x}, \mathbf{y})$, are given by

$$T_r(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T R\mathbf{y} - d_r \cdot \mathfrak{b}_r(\mathbf{x}, \mathbf{p}) \quad \text{and} \quad T_c(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T C\mathbf{y} - d_c \cdot \mathfrak{b}_c(\mathbf{y}, \mathbf{q}),$$

where d_r and d_c are non negative constants.

6.3 Comparison Between the Classes of Games

Before we present our algorithms for computing approximate equilibria for Lipschitz games it would be useful to describe the differences between the classes of games and to state what the current status of the equilibrium existence for each one class. The Figure 6.3 shows the relation between the games' classes. It is well known that normal-form games possess an equilibrium [60], known as Nash equilibrium. On the other hand, Fiat and Papadimitriou [34] initiated the study of existence of equilibria in penalty games. They studied games with penalty functions that capture risk, they showed that there exist games with no equilibrium and they proved that it is NP-complete to decide whether a game possess an equilibrium or not. Mavronicolas and Monien [58] followed the work of [34] and proved that it is NP-complete to decide the existence of equilibria for more families of penalty games. Caragiannis, Kurokawa and Proccacia [10] studied biased games and proved that a large family of biased games possess an equilirbium. Distance biased games fall in this family and thus always possess an equilibrium. Finally, for λ_p-Lipschitz games it is an interesting open question whether they possess always an equilibrium, or there are cases that do not possess an equilibrium.

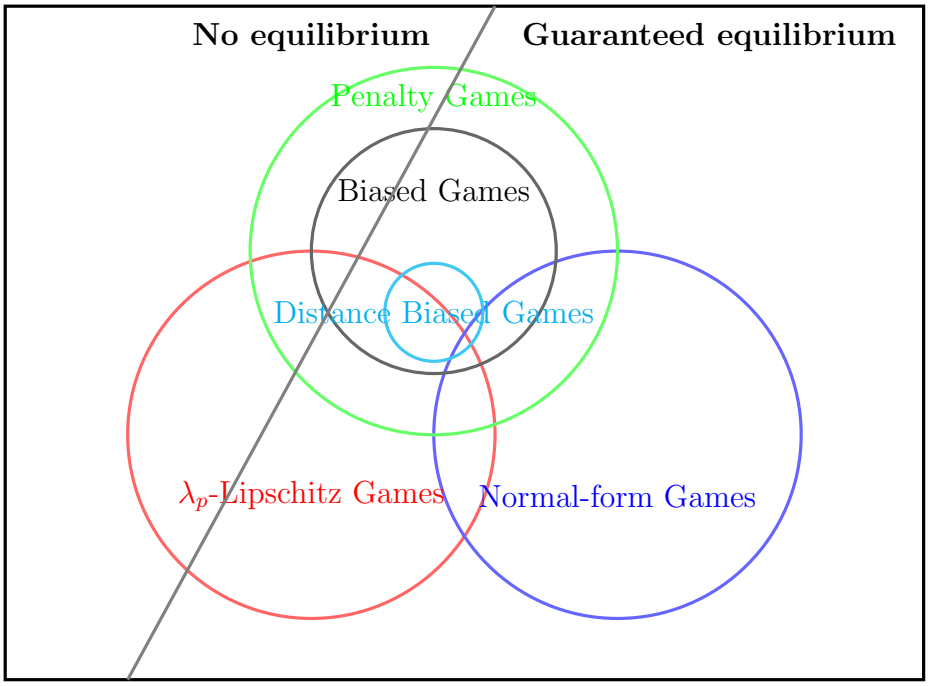

Figure 6.1: A map that depicts the relations between the games studied in this thesis and our current knowledge for the equilibrium existence for each class.

6.4 Approximate Equilibria in λ_p-Lipschitz Games

In this section, we give an algorithm for computing approximate equilibria in λ_p-Lipschitz games. Recall that, as we have already mentioned λ_p-Lipschitz games do not always possess an equilibrium. Nevertheless, our technique can be applied *irrespective* of whether an exact equilibrium exists. If an exact equilibrium does exist, then our technique will always find an ϵ-equilibrium. If an exact equilibrium does not exist, then our algorithm either finds an ϵ-equilibrium or reports that the game does not have an exact equilibrium.

In order to derive our algorithm we will utilize the following theorem that was recently proved by Barman [5]. Intuitively, Barman's theorem states that we can approximate any point μ in the convex hull of n points using a uniform point μ' that needs only "few" samples from μ to construct it.

Theorem 9 ([5]). *Given a set of vectors $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$, let $conv(X)$ denote the convex hull of X. Furthermore, let $\gamma := \max_{x \in X} \|x\|_p$ for some $2 \leq p < \infty$. For every $\epsilon > 0$ and every $\mu \in conv(X)$, there exists an $\frac{4p\gamma^2}{\epsilon^2}$ uniform vector $\mu' \in conv(X)$ such that $\|\mu - \mu'\|_p \leq \epsilon$.*

If we combine the Theorem 9 with the Definition 14 of the Lipschitz continuity,

we get the following lemma.

Lemma 30. *Let $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$, let $f : conv(X) \to \mathbb{R}$ be a λ_p-Lipschitz continuous function for some $2 \leq p < \infty$, let $\epsilon > 0$ and let $k = \frac{4\lambda^2 p \gamma^2}{\epsilon^2}$, where $\gamma := \max_{x \in X} \|x\|_p$. Furthermore, let $f(x^*)$ be the optimum value of f. Then we can compute a k-uniform point $x' \in conv(X)$ in time $O(n^k)$, such that $|f(x^*) - f(x')| < \epsilon$.*

Proof. From Theorem 9 we know that for the chosen value of k there exists a k-uniform point x' such that $\|x' - x^*\|_p < \epsilon/\lambda$. Since the function $f(x)$ is λ_p-Lipschitz continuous, we get that $|f(x^*) - f(x')| < \epsilon$. In order to compute this point, we have to exhaustively evaluate the function f in all k-uniform points and choose the point that maximizes/minimizes its value. Since there are $\binom{n+k-1}{k} = O(n^k)$ possible k-uniform points, the theorem follows. □

High level idea of our algorithm. Before we describe formally our algorithm, let us give some intuition behind it. The high level idea of our algorithm is as follows. We first prove that there exist uniform strategies that are ϵ-equilibria, for every $\epsilon > 0$, when the game possess an exact equilibrium. We prove this by using the Theorem 9. Thus, in order to find an ϵ-equilibrium we discretize the strategy space for each player and then we only have to consider the uniform strategy profiles and pick the profile that it is an ϵ-equilibrium. However, in λ_p-Lipschitz games it is not trivial to decide whether a strategy profile is an ϵ, or even an exact, equilibrium. So, we show how we can efficiently decide whether a profile is 3ϵ-equilibrium and thus we can use our algorithm to compute a 3ϵ-equilibrium.

In what follows we will study a λ_p-Lipschitz game $\mathfrak{L} := (M, n, \lambda, p, \gamma, \mathcal{T})$. Recall that M stands for the number of players, n for the number of points whose convex hull defines the strategy space of each player, $\mathcal{T}$ is the space of the λ_p-Lipschitz continuous utility functions and γ is the value that used in Theorem 9 that roughly shows how "large" is the strategy space of the players. Assuming the existence of an exact Nash equilibrium, we establish the existence of a k-uniform approximate equilibrium in the game $\mathfrak{L}$, where k depends on M, λ, p and γ. Note that λ depends heavily on p and the utility functions for the players.

Since, by the definition of λ_p-Lipschitz games, the strategy space S_i for every player i is the convex hull of n vectors $y_1, \ldots, y_n$ in $\mathbb{R}^d$, any $x_i \in S_i$ can be written as a convex combination of y_j's. Hence, $x_i = \sum_{j=1}^{n} \alpha_j y_j$, where $\alpha_j > 0$ for every $j \in [n]$ and $\sum_{j=1}^{n} \alpha_j = 1$. Then, $\alpha = (\alpha_1, \ldots, \alpha_n)$ is a probability distribution over the vectors $y_1, \ldots, y_n$, i.e. vector y_j is drawn with probability α_j. Thus, we can sample a strategy x_i by the probability distribution α.

So, let $\mathbb{X}^*$ be an equilibrium for $\mathfrak{L}$ and let $\mathbb{X}'$ be a sampled uniform strategy profile from $\mathbb{X}^*$. For each player i we define the following events

$$\begin{aligned}
\phi_i &= \big\{|T_i(x_i', \mathbb{X}'_{-i}) - T_i(x_i^*, \mathbb{X}^*_{-i})| < \epsilon/2\big\},\\
\pi_i &= \big\{T_i(x_i, \mathbb{X}'_{-i}) < T_i(x_i', \mathbb{X}'_{-i}) + \epsilon\big\} \quad \text{for all possible } x_i,\\
\psi_i &= \left\{\|x_i' - x_i^*\|_p < \frac{\epsilon}{2M\lambda}\right\} \quad \text{for some } p > 0.
\end{aligned}$$

Notice that if all the events π_i occur at the same time, then the sampled profile $\mathbb{X}'$ is an ϵ-equilibrium. We will show that if for a player i the events ϕ_i and $\bigcap_j \psi_j$ hold, then the event π_i has to be true too.

Lemma 31. *For all $i \in [M]$ it holds that $\bigcap_{j\in[M]} \psi_j \cap \phi_i \subseteq \pi_i$.*

Proof. Suppose that both events ϕ_i and $\bigcap_j \psi_{j\in[M]}$ hold. We will show that the event π_i must be true too. Let x_i be an arbitrary strategy, let $\mathbb{X}^*_{-i}$ be a strategy profile for the rest of the players, and let $\mathbb{X}'_{-i}$ be a sampled strategy profile from $\mathbb{X}^*_{-i}$. Since we assume that the events ψ_j is true for all j we get $\|\mathbb{X}'_{-i} - \mathbb{X}^*_{-i}\|_p \leq \sum_{j\neq i} \|x_j' - x_j^*\|_p$ we get that

$$\begin{aligned}
\|\mathbb{X}'_{-i} - \mathbb{X}^*_{-i}\|_p &\leq \sum_{j\neq i} \|x_j' - x_j^*\|_p\\
&\leq \sum_{j\neq i} \frac{\epsilon}{2M\lambda}\\
&< \frac{\epsilon}{2\lambda}.
\end{aligned}$$

Furthermore, since by assumption the utility functions for the players are λ_p-Lipschitz continuous we have that

$$\big|T_i(x_i, \mathbb{X}'_{-i}) - T_i(x_i, \mathbb{X}^*_{-i})\big| \leq \frac{\epsilon}{2}.$$

This means that

$$\begin{aligned}
T_i(x_i, \mathbb{X}'_{-i}) &\leq T_i(x_i, \mathbb{X}^*_{-i}) + \frac{\epsilon}{2}\\
&\leq T_i(x_i^*, \mathbb{X}^*_{-i}) + \frac{\epsilon}{2} \qquad (6.1)
\end{aligned}$$

where the last inequality holds since the strategy profile $(x_i^*, \mathbb{X}^*_{-i})$ is an equilibrium of the game. Furthermore, since by assumption the event ϕ_i is true we get that

$$T_i(x_i^*, \mathbb{X}^*_{-i}) < T_i(x_i', \mathbb{X}'_{-i}) + \frac{\epsilon}{2}. \qquad (6.2)$$

Hence, if we combine the inequalities (6.1) and (6.2) we get that $T_i(x_i, \mathbb{X}'_{-i}) < T_i(x_i', \mathbb{X}'_{-i}) + \epsilon$ for all possible x_i. Thus, if the events ϕ_i and ψ_j for every $j \in [M]$ hold, then the event π_i holds too. □

We are ready to prove the main result of the section.

Theorem 10. *Let $\mathfrak{L}$ be a λ_p-Lipschitz game that possess an equilibrium. Then, for any $\epsilon > 0$, there is a k-uniform strategy profile, with $k = \frac{16M^2\lambda^2 p\gamma^2}{\epsilon^2}$ that is an ϵ-equilibrium for $\mathfrak{L}$.*

Proof. In order to prove the claim, it suffices to show that there is a strategy profile where every player plays a k-uniform strategy such that the events π_i hold for all $i \in [M]$. Since the utility functions in $\mathfrak{L}$ are λ_p-Lipschitz continuous it holds that $\bigcap_{i\in[n]} \psi_i \subseteq \bigcap_{i\in[n]} \phi_i$. Furthermore, combining that with the Lemma 31 we get that $\bigcap_{i\in[n]} \psi_i \subseteq \bigcap_{i\in[n]} \pi_i$. Thus, if the event ψ_i is true for every $i \in [n]$, then the event $\bigcap_{i\in[n]} \pi_i$ is true as well.

From the Theorem 9 we get that for each $i \in [M]$ there is a $\frac{16M^2\lambda^2 p\gamma^2}{\epsilon^2}$-uniform point x'_i such that the event ψ_i occurs with positive probability. The claim follows. □

Theorem 10 establishes the existence of a k-uniform approximate equilibrium, but this does not immediately give us our approximation algorithm. The obvious approach is to perform a brute force check of all k-uniform strategies, and then output the one that provides the best approximation. However, there is a problem with this, since computing the quality of approximation requires us to compute the regret for each player, which in turn requires us to compute a best response for each player. Computing an exact best response in a Lipschitz game is a hard problem in general, since we make no assumptions about the utility functions of the players. Fortunately, it is sufficient to, instead, compute an *approximate* best response for each player, and Lemma 30 can be used to do this. The following Lemma is a consequence of Lemma 30.

Lemma 32. *Let $\mathbb{X}$ be a strategy profile for a λ_p-Lipschitz game $\mathfrak{L}$, and let $\hat{x}_i$ be a best response for the player i against the profile $\mathbb{X}_{-i}$. There is a $\frac{4\lambda^2 p\gamma^2}{\epsilon^2}$-uniform strategy x'_i that is an δ-best response against $\mathbb{X}_{-i}$, i.e. $|T_i(\hat{x}_i, \mathbb{X}_{-i}) - T_i(x'_i, \mathbb{X}_{-i})| < \delta$.*

Proof. In order to compute a best response it is equal to maximize the function $T_i(x_i, \mathbb{X}_{-i})$ with respect to x_i. Since we assume that $T_i(\cdot)$ is λ_p continuous we can apply Lemma 30 and compute the optimal solution, i.e. the best response for the player i. □

Our goal is to *approximate* the approximation guarantee for a given strategy profile. More formally, given a strategy profile $\mathbb{X}$ that is an ϵ-equilibrium, and a

constant $\delta > 0$, we want an algorithm that outputs a number within the range $[\epsilon - \delta, \epsilon + \delta]$. Lemma 32 allows us to do this. For a given strategy profile $\mathbb{X}$, we first compute δ-approximate best responses for each player, then we can use these to compute δ-approximate regrets for each player. The maximum over the δ-approximate regrets then gives us an approximation ϵ with a tolerance of δ. This is formalised in the following algorithm.

Algorithm 6. Evaluation of approximation guarantee

Input: A strategy profile $\mathbb{X}$ for $\mathfrak{L}$, and a constant $\delta > 0$.
Output: An additive δ-approximation of the approximation guarantee $\alpha(\mathbb{X})$ for the strategy profile $\mathbb{X}$.

1. Set $l = \frac{4\lambda^2 p\gamma^2}{\delta^2}$.
2. For every player $i \in [M]$
   (a) For every l-uniform strategy x_i' of player i compute $T_i(x_i', \mathbb{X}_{-i})$.
   (b) Set $m^* = \max_{x_i'} T_i(x_i', \mathbb{X}_{-i})$.
   (c) Set $\mathcal{R}_i(\mathbb{X}) = m^* - T_i(x_i, \mathbb{X}_{-i})$.
3. Set $\alpha(\mathbb{X}) = \delta + \max_{i \in [M]} \mathcal{R}_i(\mathbb{X})$.
4. Return $\alpha(\mathbb{X})$.

Utilising the above algorithm, we can now produce an algorithm to find an approximate equilibrium in Lipschitz games. The algorithm checks all k-uniform strategy profiles, using the value of k given by Theorem 10, and for each one, computes an approximation of the quality approximation using the Algorithm 6 given above.

Algorithm 7. 3ϵ-equilibrium for λ_p-Lipschitz game $\mathfrak{L}$

Input: Game $\mathfrak{L}$ and $\epsilon > 0$.
Output: A 3ϵ-equilibrium for $\mathfrak{L}$.

1. Set $k > \frac{16\lambda^2 Mp\gamma^2}{\epsilon^2}$.
2. For every k-uniform strategy profile $\mathbb{X}'$
   (a) Compute an ϵ-approximation of $\alpha(\mathbb{X}')$.
   (b) If the ϵ-approximation of $\alpha(\mathbb{X}')$ is less than 2ϵ, return $\mathbb{X}'$.

If the algorithm returns a strategy profile $\mathbb{X}$, then it must be a 3ϵ equilibrium. This is because we check that an ϵ-approximation of $\alpha(\mathbb{X})$ is less than 2ϵ, and

therefore $\alpha(\mathbb{X}) \leq 3\epsilon$. Secondly, we argue that if the game has an exact Nash equilibrium, then this procedure will always output a 3ϵ-approximate equilibrium. From Theorem 10 we know that if $k > \frac{16\lambda^2 Mp\gamma^2}{\epsilon^2}$, then there is a k-uniform strategy profile $\mathbb{X}$ that is an ϵ-equilibrium for $\mathfrak{L}$. When we apply our approximate regret algorithm to $\mathbb{X}$, to find an ϵ-approximation of $\alpha(\mathbb{X})$, the algorithm will return a number that is less than 2ϵ, hence $\mathbb{X}$ will be returned by the algorithm.

To analyse the running time, observe that there are $\binom{n+k-1}{k} = O(n^k)$ possible k-uniform strategies for each player, thus $O(n^{Mk})$ k-uniform strategy profiles. Furthermore, our regret approximation algorithm runs in time $O(Mn^l)$, where $l = \frac{4\lambda^2 p\gamma^2}{\epsilon^2}$. Hence, we get the next theorem.

Theorem 11. *Given a λ_p-Lipschitz game $\mathfrak{L}$ that possess an equilibrium and any $\epsilon > 0$, a 3ϵ-equilibrium can be computed in time $O\left(Mn^{Mk+l}\right)$, where $k = O\left(\frac{\lambda^2 Mp\gamma^2}{\epsilon^2}\right)$ and $l = O\left(\frac{\lambda^2 p\gamma^2}{\epsilon^2}\right)$.*

Notice that it might be computationally hard to decide whether a game possesses an equilibrium or not. Nevertheless, our algorithm can be applied in *any* λ_p-Lipschitz game, without being affected by the existence or not of an exact equilibrium. If the game does not possesses an exact equilibrium then our algorithm either finds an approximate equilibrium or fails to find an approximate equilibrium. In the latter case the algorithm decides that the game does not possess an exact equilibrium, since if it had one, it would have an ϵ-equilibrium too.

Theorem 12. *For any game λ_p-Lipschitz game $\mathfrak{L}$ in time $O\left(Mn^{Mk+l}\right)$, we can either compute a 3ϵ-equilibrium, or decide that $\mathfrak{L}$ does not possess an exact equilibrium, where $k = O\left(\frac{\lambda^2 Mp\gamma^2}{\epsilon^2}\right)$ and $l = O\left(\frac{\lambda^2 p\gamma^2}{\epsilon^2}\right)$.*

6.5 An Algorithm for Penalty Games

In this section we study two-player penalty games that belong in the class $\mathcal{P}_\lambda$, i.e. penalty games with λ-Lipschitz continuous penalty functions. We present an algorithm that, for any $\epsilon > 0$, can compute an ϵ-equilibrium for any penalty game in $\mathcal{P}_\lambda$ in quasi-polynomial time. This means that if N is the size of the game, then our algorithm can compute an ϵ-equilibrium in time $N^{O(\log N)}$. For the algorithm, we take the same approach as we did in the previous section for λ_p-Lipschitz games: We show that if an exact equilibrium exists, then a k-uniform approximate equilibrium always exists too, and provide a brute-force search algorithm for finding it. Once again, since best response computation may

be hard for this class of games, we must provide an approximation algorithm for finding the quality of an approximate equilibrium. The majority of this section is dedicated to proving an appropriate bound for k, to ensure that k-uniform approximate equilibria always exist.

We first focus on penalty games that possess an exact equilibrium. So, let $(\mathbf{x}^*, \mathbf{y}^*)$ be an equilibrium of the game and let $(\mathbf{x}', \mathbf{y}')$ be a k-uniform strategy profile sampled from this equilibrium. We define the following four events:

$$\begin{aligned}
\phi_r =& \big\{|T_r(\mathbf{x}', \mathbf{y}') - T_r(\mathbf{x}^*, \mathbf{y}^*)| < \epsilon/2\big\} \\
\pi_r =& \big\{T_r(\mathbf{x}, \mathbf{y}') < T_r(\mathbf{x}', \mathbf{y}') + \epsilon\big\} \qquad \text{for all } \mathbf{x} \\
\phi_c =& \big\{|T_c(\mathbf{x}', \mathbf{y}') - T_c(\mathbf{x}^*, \mathbf{y}^*)| < \epsilon/2\big\} \\
\pi_c =& \big\{T_c(\mathbf{x}', \mathbf{y}) < T_c(\mathbf{x}', \mathbf{y}') + \epsilon\big\} \qquad \text{for all } \mathbf{y}.
\end{aligned}$$

The events π_r and π_c ensure that under the k-uniform strategy profile $(\mathbf{x}', \mathbf{y}')$ no player can gain more than ϵ by changing his strategy, i.e. $(\mathbf{x}', \mathbf{y}')$ is an ϵ-equilibrium. The events ϕ_r and ϕ_c ensure that the payoffs the players get under the profile $(\mathbf{x}', \mathbf{y}')$ is at most ϵ-away from the payoffs the players get under the exact equilibrium. The goal is to derive a value for k such that all the four events above are true, or equivalently $Pr(\phi_r \cap \pi_r \cap \phi_c \cap \pi_r) > 0$.

Note that in order to prove that $(\mathbf{x}', \mathbf{y}')$ is an ϵ-equilibrium we *only* have to consider the events π_r and π_c. Nevertheless, as we show in the Lemma 33, the events ϕ_r and ϕ_c are crucial in our analysis. The proof of the main theorem boils down to the events ϕ_r and ϕ_c. Furthermore, proving that there is a k-uniform profile $(\mathbf{x}', \mathbf{y}')$ that fulfills the events ϕ_r and ϕ_c too, proves that the approximate equilibrium we compute approximates the utilities the players receive under an exact equilibrium too.

In what follows, we will focus only on the row player, since similar analysis can be applied for the column player. Firstly, we study the event π_r and we show how we can relate it with the event ϕ_r.

Lemma 33. *For all penalty games it holds that* $Pr(\pi_r^c) \leq n \cdot e^{-\frac{k\epsilon^2}{2}} + Pr(\phi_r^c)$.

Proof. We begin by introducing the following auxiliary events for all $i \in [n]$

$$\psi_{ri} = \big\{R_i\mathbf{y}' < R_i\mathbf{y}^* + \frac{\epsilon}{2}\big\}.$$

We prove how the events ψ_{ri} and the event ϕ_r are related with the event π_r. Assume that the event ϕ_r and the events ψ_{ri} for all $i \in [n]$ are true. Let $\mathbf{x}$ be any mixed strategy for the row player. Since by assumption $R_i\mathbf{y}' < R_i\mathbf{y}^* + \frac{\epsilon}{2}$ and since $\mathbf{x}$ is a probability distribution, it holds that $\mathbf{x}^T R\mathbf{y}' < \mathbf{x}^T R\mathbf{y}^* + \frac{\epsilon}{2}$. If

we subtract $\mathfrak{f}_r(\mathbf{x})$ from each side we get that $\mathbf{x}^T R\mathbf{y}' - \mathfrak{f}_r(\mathbf{x}) < \mathbf{x}^T R\mathbf{y}^* - \mathfrak{f}_r(\mathbf{x}) + \frac{\epsilon}{2}$. This means that $T_r(\mathbf{x}, \mathbf{y}') < T_r(\mathbf{x}, \mathbf{y}^*) + \frac{\epsilon}{2}$ for all $\mathbf{x}$. But we know that $T_r(\mathbf{x}, \mathbf{y}^*) \leq T_r(\mathbf{x}^*, \mathbf{y}^*)$ for all $\mathbf{x} \in \Delta^n$, since $(\mathbf{x}^*, \mathbf{y}^*)$ is an equilibrium. Thus, we get that $T_r(\mathbf{x}, \mathbf{y}') < T_r(\mathbf{x}^*, \mathbf{y}^*) + \frac{\epsilon}{2}$ for all possible $\mathbf{x}$. Furthermore, since the event ϕ_r is true too, we get that $T_r(\mathbf{x}, \mathbf{y}') < T_r(\mathbf{x}', \mathbf{y}') + \epsilon$. Thus, if the events ϕ_r and ψ_{ri} for all $i \in [n]$ are true, then the event π_r must be true as well. Formally, $\phi_r \bigcap_{i \in [n]} \psi_{ri} \subseteq \pi_r$. Thus, $Pr(\pi_r^c) \leq Pr(\phi_r^c) + \sum_i \psi_{ri}$. Using the Hoeffding bound, we get that $Pr(\psi_{ri}^c) \leq e^{-\frac{k\epsilon^2}{2}}$ for all $i \in [n]$. Our claim follows. □

With Lemma 33 in hand, we can see that in order to compute a value for k it is sufficient to study the event ϕ_r. We introduce the following auxiliary events that we will study seperately:

$$\phi_{ru} = \left\{ |\mathbf{x}'^T R\mathbf{y}' - \mathbf{x}^{*^T} R\mathbf{y}^*| < \epsilon/4 \right\}$$
$$\phi_{r\mathfrak{b}} = \left\{ |\mathfrak{f}_r(\mathbf{x}') - \mathfrak{f}_r(\mathbf{x}^*)| < \epsilon/4 \right\}.$$

It is easy to see that if both $\phi_{r\mathfrak{b}}$ and ϕ_{ru} are true, then the event ϕ_r must be true too, formally $\phi_{r\mathfrak{b}} \cap \phi_{ru} \subseteq \phi_r$. Using the analysis from Lipton, Markakis and Mehta [56] we can prove that $Pr(\phi_{ru}^c) \leq 2e^{-\frac{k\epsilon^2}{8}}$. Thus, it remains to study the the event $\phi_{r\mathfrak{b}}^c$.

Lemma 34. $\Pr(\phi_{r\mathfrak{b}}^c) \leq \frac{8\lambda\sqrt{p}}{\epsilon\sqrt{k}}$.

Proof. Since we assume that the penalty function $\mathfrak{f}_r(\mathbf{x}')$ is λ_p-Lipschitz continuous the event $\phi_{r\mathfrak{b}}$ can be replaced by the event $\phi_{r\mathfrak{b}'} = \left\{ \|\mathbf{x}' - \mathbf{x}^*\|_p < \epsilon/4\lambda \right\}$. It is easy to see that $\phi_{r\mathfrak{b}} \subseteq \phi_{r\mathfrak{b}'}$. Then, using the proof of Theorem 2 from [5] we get that $E[\|\mathbf{x}' - \mathbf{x}^*\|_p] \leq \frac{2\sqrt{p}}{\sqrt{k}}$. Thus, using Markov's inequality we get that

$$\begin{aligned} Pr(\|\mathbf{x}' - \mathbf{x}^*\|_p \geq \frac{\epsilon}{4\lambda}) &\leq \frac{E[\|\mathbf{x}' - \mathbf{x}^*\|_p]}{\frac{\epsilon}{4\lambda}} \\ &\leq \frac{8\lambda\sqrt{p}}{\epsilon\sqrt{k}}. \end{aligned}$$

□

We are ready to prove our theorem.

Theorem 13. *For any equilibrium $(\mathbf{x}^*, \mathbf{y}^*)$ of a penalty game from the class $\mathcal{P}_\lambda$, any $\epsilon > 0$, and any $k \in \frac{\Omega(\lambda^2 \log n)}{\epsilon^2}$, there exists a k-uniform strategy profile $(\mathbf{x}', \mathbf{y}')$ such that:*

1. *$(\mathbf{x}', \mathbf{y}')$ is an ϵ-equilibrium for the game,*

2. $|T_r(\mathbf{x}', \mathbf{y}') - T_r(\mathbf{x}^*, \mathbf{y}^*)| < \epsilon/2$,

3. $|T_c(\mathbf{x}', \mathbf{y}') - T_c(\mathbf{x}^*, \mathbf{y}^*)| < \epsilon/2$.

Proof. Let us define the event $GOOD = \phi_r \cap \phi_c \cap \pi_r \cap \pi_c$. In order to prove our theorem it suffices to prove that $Pr(GOOD) > 0$. Notice that for the events ϕ_c and π_c we can use the same analysis as for ϕ_r and π_r and get the same bounds.

Thus, using Lemma 33 and the analysis for the events ϕ_{ru} and $\phi_{r\mathfrak{b}}$ we get that

$$\begin{aligned}
Pr(GOOD^c) &\leq Pr(\phi_r^c) + Pr(\pi_r^c) + Pr(\phi_c^c) + Pr(\pi_c^c) \\
&\leq 2\big(Pr(\phi_r^c) + Pr(\pi_r^c)\big) \\
&\leq 2\big(2Pr(\phi_r^c) + n \cdot e^{-\frac{k\epsilon^2}{2}}\big) \quad \text{(from Lemma 33)} \\
&\leq 2\big(2Pr(\phi_{ru}^c) + 2Pr(\phi_{r\mathfrak{b}'}^c) + n \cdot e^{-\frac{k\epsilon^2}{2}}\big) \\
&\leq 2\big(4e^{-\frac{k\epsilon^2}{8}} + \frac{8\lambda\sqrt{p}}{\epsilon\sqrt{k}} + n \cdot e^{-\frac{k\epsilon^2}{2}}\big) \quad \text{(from Lemma 34)} \\
&< 1 \quad \text{for the chosen value of } k.
\end{aligned}$$

Thus, $Pr(GOOD) > 0$ and our claim follows. □

Theorem 13 establishes the *existence* of a k-uniform strategy profile $(\mathbf{x}', \mathbf{y}')$ that is an ϵ-equilibrium. However, as with the previous section, we must provide an efficient method for approximating the quality of approximation provided by a given strategy profile. To do so, we first give the following lemma, which shows that approximate best responses can be computed in quasi-polynomial time for penalty games.

Lemma 35. *Let $(\mathbf{x}, \mathbf{y})$ be a strategy profile for a penalty game $\mathcal{P}_\lambda$, and let $\hat{\mathbf{x}}$ be a best response against $\mathbf{y}$. There is an l-uniform strategy $\mathbf{x}'$, with $l = \frac{17\lambda^2\sqrt{p}}{\epsilon^2}$, that is an ϵ-best response against $\mathbf{y}$, i.e. $T_r(\hat{\mathbf{x}}, \mathbf{y}) < T_r(\mathbf{x}', \mathbf{y}) + \epsilon$.*

Proof. We will prove that $|T_r(\hat{\mathbf{x}}, \mathbf{y}) - T_r(\mathbf{x}', \mathbf{y})| < \epsilon$ which implies our claim. Let $\phi_1 = \{|\hat{\mathbf{x}}^T R\mathbf{y} - \mathbf{x}'^T R\mathbf{y}| \leq \epsilon/2\}$ and $\phi_2 = \{|\mathfrak{f}_r(\hat{\mathbf{x}}) - \mathfrak{f}_r(\mathbf{x}')| < \epsilon/2\}$. Notice that Lemma 34 does not use anywhere the fact that $\mathbf{x}^*$ is an equilibrium strategy, thus it holds even if $\mathbf{x}^*$ is replaced by $\hat{\mathbf{x}}$. Thus, $Pr(\phi_2^c) \leq \frac{4\lambda\sqrt{p}}{\epsilon\sqrt{k}}$. Furthermore, using the analysis from [56] again, we can prove that $Pr(\phi_1^c) \leq 2e^{-\frac{k\epsilon^2}{4}}$ and using similar arguments as in the proof of Theorem 13 it can be easily proved that for the chosen value of l it holds that $Pr(\phi_1^c) + Pr(\phi_2^c) < 1$, thus the events ϕ_1 and ϕ_2 occur with positive probability and our claim follows. □

Having given this Lemma, we can reuse Algorithm 6, but with l set equal to $\frac{17\lambda^2\sqrt{p}}{\epsilon^2}$, to provide an algorithm that aproximates the quality of approximation of a given strategy profile. Then, we can reuse Algorithm 7 with $k = \frac{\Omega(\lambda^2 \log n)}{\epsilon^2}$ to provide a quasi-polynomial time algorithm that finds approximate equilibia in penalty games. Notice again that our algorithm can be applied in games that it is computationally hard to verify whether an exact equilibrium exists. Our algorithm either will compute an approximate equilibrium or it will fail to find one, thus it will decide that the game does not posses an exact equilibrium.

Theorem 14. *In any penalty game $\mathcal{P}_\lambda$ and any $\epsilon > 0$, in time $O(n^{k+l})$, where $k = \frac{\Omega(\lambda^2 \log n)}{\epsilon^2}$ and $l = \frac{17\lambda^2\sqrt{p}}{\epsilon^2}$, we can either compute a 3ϵ-equilibrium, or decide that $\mathcal{P}_\lambda$ does not possess an exact equilibrium.*

6.6 Distance Biased Games

In this section, we focus on three particular classes of distance biased games, and we provide polynomial-time approximation algorithms for these games. Recall that distance biased games are penalty games with penalty function from a specific family of functions. More specifically, distance biased games have penalty functions of the form

$$T_r(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T R\mathbf{y} - d_r \cdot \mathfrak{b}_r(\mathbf{x}, \mathbf{p}) \quad \text{and} \quad T_c(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T C\mathbf{y} - d_c \cdot \mathfrak{b}_c(\mathbf{y}, \mathbf{q}),$$

where d_r and d_c are positive constants and $\mathbf{p}$ and $\mathbf{q}$ are "base" strategies for the player. Intuitively, the players are penalized for deviating from their base strategies. In this section we study the following three penalty functions:

- L_1 **penalty**: $\mathfrak{b}_r(\mathbf{x}, \mathbf{p}) = \|\mathbf{x} - \mathbf{p}\|_1 = \sum_i |\mathbf{x}_i - \mathbf{p}_i|$.
- L_2^2 **penalty**: $\mathfrak{b}_r(\mathbf{x}, \mathbf{p}) = \|\mathbf{x} - \mathbf{p}\|_2^2 = \sum_i (\mathbf{x}_i - \mathbf{p}_i)^2$.
- L_∞ **penalty**: $\mathfrak{b}_r(\mathbf{x}, \mathbf{p}) = \|\mathbf{x} - \mathbf{p}\|_\infty = \max_i |\mathbf{x}_i - \mathbf{p}_i|$.

Our approach is to follow the well-known technique of Daskalakis, Mehta and Papadimitriou [20] that finds a 0.5-NE in a bimatrix game. The algorithm that we will use for all three penalty functions is given below.

Algorithm 8. The Base Algorithm

1. Compute a best response $\mathbf{y}^*$ against $\mathbf{p}$.
2. Compute a best response $\mathbf{x}$ against $\mathbf{y}^*$.
3. Set $\mathbf{x}^* = \delta \cdot \mathbf{p} + (1 - \delta) \cdot \mathbf{x}$, for some $\delta \in [0, 1]$.
4. Return the strategy profile $(\mathbf{x}^*, \mathbf{y}^*)$.

While this is a well-known technique for bimatrix games, note that it cannot immediately be applied to penalty games. This is because the algorithm requires us to compute two best response strategies, and while computing a best-response is trivial in bimatrix games, this is not the case for penalty games. Best responses for L_1 and L_∞ penalties can be computed in polynomial-time via linear programming, and for L_2^2 penalties, the ellipsoid algorithm can be applied. However, these methods do not provide strongly polynomial algorithms.

In this section we develop simple combinatorial algorithms for computing best response strategies for each of these penalties. Our algorithms are strongly polynomial. Then, we determine the quality of the approximation given by the base algorithm when our best response techniques are used. In what follows we make the common assumption that the payoffs of the underlying bimatrix game (R, C) are in $[0, 1]$.

6.6.1 A 2/3-Approximation Algorithm for L_1-Biased Games

We start by considering L_1-biased games. Suppose that we want to compute a best-response for the row player against a fixed strategy $\mathbf{y}$ of the column player. We will show that best response strategies in L_1-biased games have a very particular form: if b is the best response strategy in the (unbiased) bimatrix game (R, C), then the best-response places all of its probability on b *except* for a certain set of rows S where it is too costly to shift probability away from $\mathbf{p}$. The rows $i \in S$ will be played with $\mathbf{p}_i$ to avoid taking the penalty for deviating.

The characterisation for whether it is too expensive to shift away from $\mathbf{p}$ is given by the following lemma.

Lemma 36. *Let j be a pure strategy, let k be a pure strategy with $\mathbf{p}_k > 0$, and let $\mathbf{x}$ be a strategy with $\mathbf{x}_k = \mathbf{p}_k$. The utility for the row player increases when we shift probability from k to j if and only if $R_j\mathbf{y} - R_k\mathbf{y} - 2d_r > 0$.*

Proof. Suppose that we shift δ probability from k to j, where $\delta \in (0, \mathbf{p}_k]$. Then the utility for the row player is equal to $T_r(\mathbf{x}, \mathbf{y}) + \delta \cdot (R_j\mathbf{y} - R_k\mathbf{y} - 2d_r)$, where

the final term is the penalty for shifting away from k. Thus, the utility for the row player increases under this shift if and only if $R_j\mathbf{y} - R_k\mathbf{y} - 2d_r > 0$. □

Observe that, if we are able to shift probability away from a strategy k, then we should obviously shift it to a best response strategy for the (unbiased) bimatrix game, since this strategy maximizes the increase in our payoff. Hence, our characterisation of best response strategies is correct. This gives us the following simple algorithm for computing best responses.

Algorithm 9. Best Response Algorithm for L_1 penalty

1. Set $S = 0$.
2. Compute a best response b against $\mathbf{y}$ in the unbiased bimatrix game (R, C).
3. For each index $i \neq b$ in the range $1 \leq i \leq n$:
   (a) If $R_b \cdot \mathbf{y} - R_i \cdot \mathbf{y} - 2d_r \leq 0$, then set $\mathbf{x}_i = \mathbf{p}_i$ and $S = S + \mathbf{p}_i$.
   (b) Otherwise set $\mathbf{x}_i = 0$
4. Set $\mathbf{x}_b = 1 - S$.
5. Return $\mathbf{x}$.

Our characterisation has a number of consequences. Firstly, it can be seen that if $d_r \geq 1/2$, then there is no profitable shift of probability between any two pure strategies, since $0 \leq R_i\mathbf{y} \leq 1$ for all $i \in [n]$. Thus, we get the following corollary.

Corollary 2. *If $d_r \geq 1/2$, then the row player always plays the strategy $\mathbf{p}$ irrespectively from which strategy his opponent plays, i.e. $\mathbf{p}$ is a dominant strategy.*

Moreover, since we can compute a best response in polynomial time we get the next theorem.

Theorem 15. *In biased games with L_1 penalty functions and $\max\{d_r, d_c\} \geq 1/2$, an equilibrium can be computed in polynomial time.*

Proof. Assume that $d_r \geq 1/2$. Then from Corollary 2 the row player will play his base strategy $\mathbf{p}$. Then we can use Algorithm 9 to compute a best response against $\mathbf{p}$ for the column player. Then, this profile will be an equilibrium for the game since no player can increase his payoff by unilaterally changing his strategy. □

Finally, using the characterization of best responses we can see that there is a connection between the equilibria of the distance biased game and the well supported Nash equilibria (WSNE) of the underlying bimatrix game.

Theorem 16. *Let $\mathcal{B} = \big(R, C, \mathfrak{b}_r(\mathbf{x}, \mathbf{p}), \mathfrak{b}_c(\mathbf{y}, \mathbf{q}), d_r, d_c\big)$ be a distance biased game with L_1 penalties and let $d := \max\{d_r, d_c\}$. Any equilirbium of $\mathcal{B}$ is a $2d$-WSNE for the bimatrix game (R, C).*

Proof. Let $(\mathbf{x}^*, \mathbf{y}^*)$ be an equilibrium for $\mathcal{B}$. From the best response Algorithm 9 we can see that $\mathbf{x}_i^* > 0$ if and only if $R_b \cdot \mathbf{y}^* - R_i \cdot \mathbf{y}^* - 2d_r \leq 0$, where b is a pure best response against $\mathbf{y}^*$. This means that for every $i \in [n]$ with $\mathbf{x}_i^* > 0$, it holds that $R_i \cdot \mathbf{y}^* \geq \max_{j \in [n]} R_j \cdot \mathbf{y}^* - 2d$. Similarly, it holds that $C_i^T \cdot \mathbf{x}^* \geq \max_{j \in [n]} C_j^T \cdot \mathbf{x}^* - 2d$ for all $i \in [n]$ with $\mathbf{y}_i^* > 0$. This is the definition of a $2d$-WSNE for the bimatrix game (R, C). □

Approximation Algorithm

We now analyse the approximation guarantee provided by the base algorithm for L_1-biased games. So, let $(\mathbf{x}^*, \mathbf{y}^*)$ be the strategy profile that is returned by the base algorithm. Since we have already shown that exact Nash equilibria can be found in games with either $d_c \geq 1/2$ or $d_r \geq 1/2$, we will assume that both d_c and d_r are less than $1/2$, since this is the only interesting case.

We start by considering the regret of the row player. The following lemma will be used in the analysis of all three of our approximation algorithms.

Lemma 37. *Under the strategy profile $(\mathbf{x}^*, \mathbf{y}^*)$ the regret for the row player is at most δ.*

Proof. Notice that for all $i \in [n]$ we have

$$|\delta \mathbf{p}_i + (1 - \delta)\mathbf{x}_i - \mathbf{p}_i| = (1 - \delta)|\mathbf{x}_i - \mathbf{p}_i|,$$

hence $\|\mathbf{x}^* - \mathbf{p}\|_1 = (1-\delta)\|\mathbf{x} - \mathbf{p}\|_1$ and $\|\mathbf{x}^* - \mathbf{p}\|_\infty = (1-\delta)\|\mathbf{x} - \mathbf{p}\|_\infty$. Furthermore, notice that $\sum_i \big((1 - \delta)\mathbf{x}_i + \delta \mathbf{p}_i - \mathbf{p}_i\big)^2 = (1 - \delta)^2 \|\mathbf{x} - \mathbf{p}\|_2^2$, thus $\|\mathbf{x}^* - \mathbf{p}\|_2^2 \leq (1-\delta)\|\mathbf{x} - \mathbf{p}\|_2^2$. Hence, for the payoff of the row player it holds that $T_r(\mathbf{x}^*, \mathbf{y}^*) \geq \delta \cdot T_r(\mathbf{p}, \mathbf{y}^*) + (1 - \delta) \cdot T_r(\mathbf{x}, \mathbf{y}^*)$ and his regret under the strategy profile $(\mathbf{x}^*, \mathbf{y}^*)$ is

$$\begin{aligned}
\mathcal{R}^r(\mathbf{x}^*, \mathbf{y}^*) &= \max_{\tilde{\mathbf{x}}} T_r(\tilde{\mathbf{x}}, \mathbf{y}^*) - T_r(\mathbf{x}^*, \mathbf{y}^*) \\
&= T_r(\mathbf{x}, \mathbf{y}^*) - T_r(\mathbf{x}^*, \mathbf{y}^*) \qquad \text{(since } \mathbf{x} \text{ is a best response against } \mathbf{y}^*\text{)} \\
&\leq \delta\big(T_r(\mathbf{x}, \mathbf{y}^*) - T_r(\mathbf{p}, \mathbf{y}^*)\big) \\
&\leq \delta \qquad \text{(since } \max_{\mathbf{x}} T_r(\mathbf{x}, \mathbf{y}^*) \leq 1 \text{ and } T_r(\mathbf{p}, \mathbf{y}^*) \geq 0\text{)}.
\end{aligned}$$

□

Next, we consider the regret of the column player. The following lemma will be used for both the L_1 case and the L_∞ case. Observe that in the L_1 case, the precondition of $d_c \cdot \mathfrak{b}_c(\mathbf{y}^*, \mathbf{q}) \leq 1$ always holds, since we have $\|\mathbf{y}^* - \mathbf{q}\|_1 \leq 2$. Thus $d_c \cdot \mathfrak{b}_c(\mathbf{y}^*, \mathbf{q}) \leq 1$ since we are only interested in the case where $d_c \leq 1/2$.

Lemma 38. *If $d_c \cdot \mathfrak{b}_c(\mathbf{y}^*, \mathbf{q}) \leq 1$, then under strategy profile $(\mathbf{x}^*, \mathbf{y}^*)$ the column player suffers at most $2 - 2\delta$ regret.*

Proof. The regret of the column player under the strategy profile $(\mathbf{x}^*, \mathbf{y}^*)$ is

$$\begin{aligned}
\mathcal{R}^c(\mathbf{x}^*, \mathbf{y}^*) &= \max_{\mathbf{y}} T_c(\mathbf{x}^*, \mathbf{y}) - T_c(\mathbf{x}^*, \mathbf{y}^*) \\
&= \max_{\mathbf{y}} \Big\{ (1-\delta) T_c(\mathbf{x}, \mathbf{y}) + \delta T_c(\mathbf{p}, \mathbf{y}) \big) \Big\} - (1-\delta) T_c(\mathbf{x}, \mathbf{y}^*) - \delta T_c(\mathbf{p}, \mathbf{y}^*) \\
&\leq (1-\delta) \big(\max_{\mathbf{y}} T_c(\mathbf{x}^*, \mathbf{y}) - T_c(\mathbf{x}, \mathbf{y}^*) \big) \text{(since } \mathbf{y}^* \text{ is a best response against } \mathbf{p}) \\
&\leq (1-\delta)(1 + d_c \cdot \mathfrak{b}_c(\mathbf{y}^*, \mathbf{q})) \quad (\text{since } \max_{\mathbf{x}} T_c(\mathbf{x}^*, \mathbf{y}) \leq 1) \\
&\leq (1-\delta) \cdot 2 \quad (\text{since } d_c \cdot \mathfrak{b}_c(\mathbf{y}^*, \mathbf{q}) \leq 1).
\end{aligned}$$

□

To complete the analysis, we must select a value for δ that equalises the two regrets. It can easily be verified that setting $\delta = 2/3$ ensures that $\delta = 2 - 2\delta$, and so we have the following theorem.

Theorem 17. *In biased games with L_1 penalties a 2/3-equilibrium can be computed in polynomial time.*

6.6.2 A 5/7-Approximation Algorithm for L_2^2-Biased Games

We now turn our attention to biased games with an L_2^2 penalty. Again, we start by giving a combinatorial algorithm for finding a best response. Throughout this section, we fix $\mathbf{y}$ as a column player strategy, and we will show how to compute a best response for the row player.

Best responses in L_2^2-biased games can be found by solving a quadratic program, and actually this particular quadratic program can be solved via the ellipsoid algorithm [54]. We will give a simple combinatorial algorithm that uses the Karush-Kuhn-Tucker (KKT) conditions, and produces a closed formula for the solution. Hence, we will obtain a strongly polynomial time algorithm for finding best responses.

Our algorithm can be applied on L_2^2 penalty functions and any value d_r, but for notation simplicity we describe our method for $d_r = 1$. Furthermore, we define $\alpha_i := R_i\mathbf{y} + 2\mathbf{p}_i$ and we call α_i as the payoff of pure strategy i. Then, the utility for the row player can be written as $T_r(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \mathbf{x}_i \cdot \alpha_i - \sum_{i=1}^{n} \mathbf{x}_i^2 - \mathbf{p}^T\mathbf{p}$. Notice that the term $\mathbf{p}^T\mathbf{p}$ is a constant and it does not affect the solution of the best response; so we can exclude it from our computations. Thus, a best response for the row player against strategy $\mathbf{y}$ is the solution of the following quadratic program

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} \mathbf{x}_i \cdot \alpha_i - \sum_{i=1}^{n} \mathbf{x}_i^2 \\ \text{subject to} \quad & \sum_{i=1}^{n} \mathbf{x}_i = 1 \\ & \mathbf{x}_i \geq 0 \quad \text{for all } i \in [n]. \end{aligned}$$

The Lagrangian function for this problem is

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \lambda, \mathbf{u}) = \sum_{i=1}^{n} \mathbf{x}_i \cdot \alpha_i - \sum_{i=1}^{n} \mathbf{x}_i^2 - \lambda(\sum_{i=1}^{n} \mathbf{x}_i - 1) - \sum_{i=1}^{n} u_i\mathbf{x}_i$$

and the corresponding KKT conditions

$$\alpha_i - \lambda - 2\mathbf{x}_i - \mathbf{u}_i = 0 \quad \text{for all } i \in [n] \tag{6.3}$$

$$\sum_{i=1}^{n} \mathbf{x}_i = 1 \tag{6.4}$$

$$\mathbf{x}_i \geq 0 \quad \text{for all } i \in [n] \tag{6.5}$$

$$\mathbf{x}_i \cdot \mathbf{u}_i = 0 \quad \text{for all } i \in [n]. \tag{6.6}$$

Constraints (6.3)-(6.5) are the stationarity conditions and (6.6) are the complementarity slackness conditions. We say that strategy $\mathbf{x}$ is a *feasible response* if it satisfies the KKT conditions. The obvious way to compute a best response is by exhaustively checking all 2^n possible combinations for the complementarity conditions and choose the feasible response that maximizes the utility for a player. Next we prove how we can bypass the brute force technique and compute all best responses in polynomial time.

In what follows, without loss of generality, we assume that $\alpha_1 \geq \ldots \geq \alpha_n$. That is, the pure strategies are ordered according to their payoffs. In the next lemma we prove that in every best response, if a player plays the pure strategy l with positive probability, then he must play every pure strategy k with $k < l$ with positive probability.

Lemma 39. *In every best response* $\mathbf{x}^*$*, if* $\mathbf{x}_l^* > 0$ *then* $\mathbf{x}_k^* > 0$ *for all* $k < l$*.*

Proof. For the sake of contradiction suppose that there is a best response $\mathbf{x}^*$ and a $k < l$ such that $\mathbf{x}_l^* > 0$ and $\mathbf{x}_k^* = 0$. Let us denote $M = \sum_{i \neq \{l,k\}} \alpha_i \cdot \mathbf{x}_i^* - \sum_{i \neq \{l,k\}} {\mathbf{x}_i^*}^2$. Suppose now that we shift some probability, denoted by δ, from pure strategy l to pure strategy k. Then the utility for the row player is $T_r(\mathbf{x}^*, \mathbf{y}) = M + \alpha_l \cdot (\mathbf{x}_l^* - \delta) - (\mathbf{x}_l^* - \delta)^2 + \alpha_k \cdot \delta - \delta^2$, which is maximized for $\delta = \frac{\alpha_k - \alpha_l + 2\mathbf{x}_l^*}{4}$. Notice that $\delta > 0$ since $\alpha_k \geq \alpha_l$ and $\mathbf{x}_l^* > 0$. Thus the row player can increase his utility by assigning positive probability to pure strategy k, which contradicts the fact that $\mathbf{x}^*$ is a best response. □

Lemma 39 implies that there are only n possible supports that a best response can use. Indeed, we can exploit the KKT conditions to derive, for each candidate support, the exact probability that each pure strategy would be played. We derive the probability as a function of α_i's and of the support size. Suppose that the KKT conditions produce a feasible response when we set the support to have size k. From condition (6.3) we get that $\mathbf{x}_i = \frac{1}{2}(\alpha_i - \lambda)$ for all $1 \leq i \leq k$ and zero else. But we know that $\sum_j^k \mathbf{x}_j = 1$. Thus we get that $\sum_{j=1}^k \frac{1}{2}(\alpha_j - \lambda) = 1$ and if we solve for λ, we get that $\lambda = \frac{\sum_{j=1}^k \alpha_j - 2}{k}$. This means that for all $i \in [k]$ we get

$$\mathbf{x}_i = \frac{1}{2}\left(\alpha_i - \frac{\sum_{j=1}^k \alpha_j - 2}{k}\right). \tag{6.7}$$

So, our algorithm does the following. It iterates through all n candidate supports for a best response. For each one, it uses Equation (6.7) to determine the probabilities, and then checks whether these satisfy the KKT conditions, and thus, if this is a feasible response. If it is, then it is saved in a list of feasible responses, otherwise it is discarded. After all n possibilities have been checked, a feasible response with the highest payoff is then returned.

Algorithm 10. Best Response Algorithm for L_2^2 penalty

1. For $i = 1 \ldots n$
   (a) Set $\mathbf{x}_1 \geq \ldots \geq \mathbf{x}_i > 0$ and $\mathbf{x}_{i+1} = \ldots = \mathbf{x}_n = 0$.
   (b) Check if there is a feasible response under these constraints.
   (c) If so, add it to the list of feasible responses.
2. Among the feasible responses choose one with the highest utility.

We now show that the base algorithm gives a 5/7-approximation when applied to L_2^2-penalty games. For the row player's regret, we can use Lemma 37 to show

that the regret is bounded by δ. However, for the column player's regret, things are more involved. We will show that the regret of the column player is at most $2.5 - 2.5\delta$. That analysis depends on the maximum entry of the base strategy $\mathbf{q}$ and more specifically on whether $\max_k\{\mathbf{q}_k\} \leq 1/2$ or not.

Lemma 40. *If* $\max_k\{\mathbf{q}_k\} \leq 1/2$*, then the regret the column player suffers under strategy profile* $(\mathbf{x}^*, \mathbf{y}^*)$ *is at most* $2.5 - 2.5\delta$*.*

Proof. Note that when $\max_k\{\mathbf{q}_k\} \leq 1/2$, then $\mathfrak{b}_c = \|\mathbf{y} - \mathbf{p}\|_2^2 \leq 1.5$ for all possible $\mathbf{y}$. Then, using the analysis from Lemma 38, along with the fact that $d_c \cdot \mathfrak{b}_c(\mathbf{y}^*, \mathbf{q}) \leq 2$ for L_2^2 penalties, and since by assumption $d_c = 1$, the claim follows. □

For the case where there is a k such that $\mathbf{q}_k > 1/2$ a more involved analysis is needed. The first goal is to prove that under *any* strategy $\mathbf{y}^*$ that is a best response against $\mathbf{p}$ the pure strategy k is played with positive probability. In order to prove that, first it is proven that there is a feasible response against strategy $\mathbf{p}$ where pure strategy k is played with positive probability. In what follows we denote $\alpha_i := C_i^T\mathbf{p} + 2\mathbf{q}_i$.

Lemma 41. *Let* $\mathbf{q}_k > 1/2$ *for some* $k \in [n]$*. Then there is a feasible response where pure strategy* k *is played with positive probability.*

Proof. Note that $\alpha_k > 1$ since by assumption $\mathbf{q}_k > 1/2$. Recall from Equation (6.7) that, in a feasible response $\mathbf{y}$, it holds that $\mathbf{y}_i = \frac{1}{2}\left(\alpha_i - \frac{\sum_{j=1}^k \alpha_j - 2}{k}\right)$.

In order to prove the claim it is sufficient to show that $\mathbf{y}_k > 0$ when in the KKT conditions is set $\mathbf{y}_i > 0$ for all $i \in [k]$. Or equivalently, to show that $\alpha_k - \frac{\sum_{j=1}^k \alpha_j - 2}{k} = \frac{1}{k}\big((k-1)\alpha_k + 2 - \sum_{j=1}^{k-1}\alpha_j\big) > 0$. But,

$$\begin{aligned}
(k-1)\alpha_k + 2 - \sum_{j=1}^{k-1}\alpha_j &> k + 1 - \sum_{j=1}^{k-1}\left(C^T\mathbf{x} + 2\mathbf{q}_i\right) \quad (\text{since } \alpha_k > 1)\\
&\geq k + 1 - (k-1) - \sum_{j=1}^{k-1} 2\mathbf{q}_i\\
&\geq 1 + \mathbf{q}_k \quad (\text{since } \mathbf{q} \in \Delta^n)\\
&> 0.
\end{aligned}$$

The claim follows. □

Next it is proven that the utility of the column player is increasing when he adds pure strategies i in his support such that $\alpha_i > 1$.

Lemma 42. *Let $\mathbf{y}^k$ and $\mathbf{y}^{k+1}$ be two feasible responses with support size k and $k+1$ respectively, where $\alpha_{k+1} > 1$. Then $T_c(\mathbf{x}, \mathbf{y}^{k+1}) > T_c(\mathbf{x}, \mathbf{y}^k)$.*

Proof. Let $\mathbf{y}^k$ be a feasible response with support size k for the column player against strategy $\mathbf{p}$ and let $\lambda(k) := \frac{\sum_{j=1}^{k} \alpha_j - 2}{2k}$. Then the utility of the column player when he plays $\mathbf{y}^k$ can be written as

$$\begin{aligned}
T_c(\mathbf{x}, \mathbf{y}^k) &= \sum_{i=1}^{n} \mathbf{y}_i^k \cdot \alpha_i - \sum_{i=1}^{n} (\mathbf{x}_i^k)^2 - \mathbf{q}^T\mathbf{q} \\
&= \sum_{i=1}^{k} \mathbf{y}_i^k \big(\alpha_i - \mathbf{y}_i^k\big) - \mathbf{q}^T\mathbf{q} \\
&= \sum_{i=1}^{k} \left(\frac{\alpha_i}{2} - \lambda(k)\right)\left(\frac{\alpha_i}{2} + \lambda(k)\right) - \mathbf{q}^T\mathbf{q} \\
&= \frac{1}{4}\sum_{i=1}^{k} \alpha_i^2 - k \cdot \big(\lambda(k)\big)^2 - \mathbf{q}^T\mathbf{q}.
\end{aligned}$$

The goal now is to prove that $T_c(\mathbf{x}, \mathbf{y}^{k+1}) - T_c(\mathbf{x}, \mathbf{y}^k) > 0$. By the previous analysis for $T_c(\mathbf{x}, \mathbf{y}^k)$ and if $A := \sum_{i=1}^{k} \alpha_i - 2$, then

$$\begin{aligned}
T_c(\mathbf{x}, \mathbf{y}^{k+1}) - T_c(\mathbf{x}, \mathbf{y}^k) &= \frac{1}{4}\sum_{i=1}^{k+1} \alpha_i^2 - (k+1)\big(\lambda(k+1)\big)^2 - \frac{1}{4}\sum_{i=1}^{k} \alpha_i^2 + k \cdot \big(\lambda(k)\big)^2 \\
&= \frac{1}{4}\left(\alpha_{k+1}^2 + \frac{A^2}{k} - \frac{(A+\alpha_{k+1})^2}{k+1}\right) \\
&= \frac{1}{4}\left(\alpha_{k+1}^2 + \frac{1}{k+1}(A^2 - \alpha_{k+1}^2 - 2A\alpha_{k+1})\right) \\
&= \frac{1}{4(k+1)}\big(k\alpha_{k+1}^2 + A^2 - 2A\alpha_{k+1}\big) \\
&> \frac{1}{4(k+1)}\big(k + A^2 - 2A\big) \quad (\text{since } 1 < \alpha_{k+1} \leq 2 \text{ and } A > k-2) \\
&> \frac{1}{4(k+1)}\big(k^2 - 5k + 8\big) \quad (\text{since } A > k-2) \\
&> 0.
\end{aligned}$$

□

Notice that $\alpha_k \geq 2\mathbf{p}_k > 1$. Thus, the utility of the feasible response that assigns positive probability to pure strategy k is strictly greater than the utility of any feasible response that does not assign probability to k. Thus, strategy k is always played in a best response. Hence, the next lemma follows.

Lemma 43. *If there is a $k \in [n]$ such that $\mathbf{q}_k > 1/2$, then in* every *best response $\mathbf{y}^*$ the pure strategy k is played with positive probability.*

Using now Lemma 43 we can provide a better bound for the regret the column player suffers, since in every best response $\mathbf{y}^*$ the pure strategy k is played with positive probability.

Lemma 44. *Let $\mathbf{y}^*$ be a best response when there is a pure strategy k with $\mathbf{q}_k > 1/2$. Then the regret for the column player under strategy profile $(\mathbf{x}^*, \mathbf{y}^*)$ is bounded by $2 - 2\delta$.*

Proof. Recall from the analysis for the Algorithm 1 that the regret for the column player is

$$\begin{aligned}\mathcal{R}^c(\mathbf{x}^*, \mathbf{y}^*) &\leq (1-\delta)\Big(\max_{\tilde{\mathbf{y}} \in \Delta}\{\hat{\mathbf{x}}^T C \tilde{\mathbf{y}}\} + 2\tilde{\mathbf{y}}^T \mathbf{q}_k - 2\mathbf{y}^{*^T}\mathbf{q} + \mathbf{y}^{*^T}\mathbf{y}^*\Big) \\ &\leq (1-\delta)\big(1 + 2\mathbf{q}_k - 2\mathbf{y}^{*^T}\mathbf{q} + \mathbf{y}^{*^T}\mathbf{y}^*\big). \qquad (6.8)\end{aligned}$$

We focus now on the term $\mathbf{y}^{*^T}\mathbf{y}^* - 2\mathbf{y}^{*^T}\mathbf{q}$. It can be proven [1] that $\mathbf{y}^{*^T}\mathbf{y}^* - 2\mathbf{y}^{*^T}\mathbf{q} \leq 1 - 2\mathbf{q}_k$. Thus, from (6.8) we get that $\mathcal{R}^c(\mathbf{x}^*, \mathbf{y}^*) \leq 2 - 2\delta$. □

Recall now that the regret for the row player is bounded by δ, so if we optimize with respect to δ the regrets are equal for $\delta = 2/3$. Thus, the next theorem follows, since when there is a k with $\mathbf{q}_k > 1/2$ the Algorithm 1 produces a 2/3-equilibrium. Hence, combining this with Lemma 40, the Theorem 18 follows for $\delta = 5/7$.

Theorem 18. *In biased games with L_2^2 penalties a 5/7-equilibrium can be computed in polynomial time.*

6.6.3 A 13/21-Approximation for Inner-Product Penalty Games

We observe that we can also tackle the case where the penalty function is the inner product of the strategy played, i.e. when the players have L_2^2 penalties and $\mathbf{p} = \mathbf{q} = \mathbf{0}$. For these games, that we call "inner product penalty games", we replace $\mathbf{p}$ as the starting point of the base algorithm with the fully mixed strategy $\mathbf{x}^n$. Hence, for that case $\mathbf{x}^* = \delta \cdot \mathbf{x}^n + (1-\delta) \cdot \mathbf{x}$ for some $\delta \in [0, 1]$.

Again, the regret the row player suffers under strategy profile $(\mathbf{x}^*, \mathbf{y}^*)$ is bounded by δ.

[1] Section 6.6.5

Lemma 45. *When the penalty function is the inner product of the strategy played, then the regret for the row player under strategy profile* $(\mathbf{x}^*, \mathbf{y}^*)$ *is bounded by* δ*.*

Furthermore, using similar analysis as in Lemma 38, it can be proven that the regret for the column player under strategy profile $(\mathbf{x}^*, \mathbf{y}^*)$ is bounded by $(1-\delta)(1 + d_c \cdot \mathbf{y}^{*T}\mathbf{y}^*)$. For the column player we will distinguish between the cases where $d_c \leq 1/2$ and $d_c > 1/2$. For the first case where $d_c \leq 1/2$ it is easy see that the algorithm produces a 0.6-equilibrium. For the other case, when $d_c > 1/2$, first it is proven that there is no pure best response.

Lemma 46. *If the penalty for the column player is equal to* $\mathbf{y}^T\mathbf{y}$ *and* $d_c > \frac{1}{2}$*, then there is no pure best response against any strategy of the row player.*

Proof. Let C_j to denote the payoff of the column player from his j-th pure strategy against some strategy $\mathbf{x}$ played by the row player. For the sake of contradiction, assume that there is a pure best response for the column player where, without loss of generality, he plays only his first pure strategy. Suppose now that he shifts some probability to his second strategy, that is, he plays the first pure strategy with probability x and the second pure strategy with probability $1-x$. The utility for the column player under this mixed strategy is $x \cdot C_1 + (1-x) \cdot C_2 - d_c \cdot (x^2 + (1-x)^2)$, which is maximized for $x = \frac{2d_c + C_1 - C_2}{4d_c}$, where C_1 and C_2 stand for the first and the second column respectively of the matrix C. Notice that $x > 0$, which means that the column player can deviate from the pure strategy and increase his utility. The claim follows. □

With Lemma 46 in hand, it can be proven that when $d_c > 1/2$ the column player does not play any pure strategy with probability greater than 3/4.

Lemma 47. *If* $d_c > 1/2$*, then in* $\mathbf{y}^*$ *no pure strategy is played with probability greater than 3/4.*

Proof. For the sake of contradiction suppose that there is a pure strategy i in $\mathbf{y}^*$ that is played with probability greater than 3/4. Furthermore, let k be the support size of $\mathbf{y}^*$. From Lemma 46, since $d_c > 1/2$, we know that there is no pure best response, thus $k \geq 2$. Then using Equation (6.7) we get that $\frac{3}{4} < \frac{1}{2}\big(\alpha_i - \frac{\sum_{j=1}^{k} \alpha_j - 2}{k}\big)$. If we solve for α_j we get that $\alpha_i > \frac{3k-4}{2k-2} > 1$ which is a contradiction since, when $\mathbf{q} = \mathbf{0}$, it holds that $\alpha_i = C_i^T\mathbf{x} \leq 1$. □

A direct corollary from Lemma 47 is that $\mathbf{y}^{*T}\mathbf{y}^* \leq 5/8$. Hence, we can prove the following lemma.

Lemma 48. *Under strategy profile* $(\mathbf{x}^*, \mathbf{y}^*)$ *the regret for the column player is bounded by* $\frac{13}{8}(1-\delta)$.

Proof. Firstly, note that $T_c(\mathbf{x}^*, \mathbf{y}^*) = \delta {\mathbf{x}^n}^T C\mathbf{y}^* + (1-\delta)\mathbf{x}^T C\mathbf{y}^* - {\mathbf{y}^*}^T\mathbf{y}^*$. Moreover, $\max_{\tilde{\mathbf{y}}\in\Delta}\{{\mathbf{x}^n}^T C\tilde{\mathbf{y}} - \tilde{\mathbf{y}}^T\tilde{\mathbf{y}}\} - T_c(\mathbf{x}^n, \mathbf{y}^*) = 0$, since $\mathbf{y}^*$ is a best response against $\mathbf{x}^n$. Finally, notice that $0 \leq \mathbf{y}^T\mathbf{y} \leq 1$ for all $\mathbf{y}$. Thus, the regret for the column player is

$$\begin{aligned}\mathcal{R}^c(\mathbf{x}^*, \mathbf{y}^*) &= (1-\delta)\Big(\max_{\tilde{\mathbf{y}}\in\Delta}\{\mathbf{x}^T C\tilde{\mathbf{y}} - \tilde{\mathbf{y}}^T\tilde{\mathbf{y}}\} - \mathbf{x}^T C\mathbf{y}^* + {\mathbf{y}^*}^T\mathbf{y}^*\Big) \\ &< (1-\delta)\big(1+\frac{5}{8}\big).\end{aligned}$$

which matches the claimed result. □

If we combine Lemmas 45 and 48 and solve for δ we can see that the regrets are equal for $\delta = \frac{13}{21}$. Thus, we get the following theorem for biased games where $\mathbf{q} = \mathbf{0}$.

Theorem 19. *The strategy profile* $(\mathbf{x}^*, \mathbf{y}^*)$ *is a* $\frac{13}{21}$*-equilibrium for biased games with* $\mathbf{q} = \boldsymbol{0}$.

6.6.4 A 2/3-Approximation for L_∞-Biased Games

Finally, we turn our attention to the L_∞ penalty. We start by giving a combinatorial algorithm for finding best responses. Similar to the best response Algorithm for the L_1 penalty, the intuition is to start from the base strategy $\mathbf{p}$ of the row player and shift probability from pure strategies with low payoff to pure strategies with higher payoff. This time though, the shifted probability will be distributed between the pure strategies with higher payoff.

Without loss of generality assume that $R_1\mathbf{y} \geq \ldots \geq R_n\mathbf{y}$, ie., that the strategies are ordered according to their payoff in the unbiased bimatrix game. The set of pure strategies of the row player can be partitioned into three disjoint sets according to the payoff they yield:

$$\begin{aligned}\mathcal{H} &:= \{i \in [n] : R_i\mathbf{y} = R_1\mathbf{y}\} \\ \mathcal{M} &:= \{i \in ([n] \setminus \mathcal{H}) : R_1\mathbf{y} - R_i\mathbf{y} - d_r < 0\} \\ \mathcal{L} &:= \{i \in [n] : R_1\mathbf{y} - R_i\mathbf{y} - d_r > 0\}.\end{aligned}$$

Next we give an algorithm that computes a best response for L_∞ penalty.

Algorithm 11. Best Response Algorithm for L_∞ penalty

1. For all $i \in \mathcal{L}$, set $x_i = 0$.
2. If $\mathcal{P} \leq |\mathcal{H}| \cdot p_{\max}$, then set $x_i = \mathbf{p}_i + \frac{\mathcal{P}}{|\mathcal{H}|}$ for all $i \in \mathcal{H}$ and $x_j = \mathbf{p}_j$ for $j \in \mathcal{M}$.
3. Else if $\mathcal{P} < |\mathcal{H} \cup \mathcal{M}| \cdot p_{\max}$, then
   - Set $x_i = \mathbf{p}_i + p_{\max}$ for all $i \in \mathcal{H}$.
   - Set $k = \lfloor \frac{\mathcal{P} - |\mathcal{H}| \cdot p_{\max}}{p_{\max}} \rfloor$.
   - Set $x_i = \mathbf{p}_i + p_{\max}$ for all $i \leq |\mathcal{H}| + k$.
   - Set $x_{|\mathcal{H}|+k+1} = \mathbf{p}_{|\mathcal{H}|+k+1} + \mathcal{P} - (|\mathcal{H}| + k) \cdot p_{\max}$.
   - Set $x_j = \mathbf{p}_j$ for all $|\mathcal{H}| + k + 2 \leq j \leq |\mathcal{H}| + |\mathcal{M}|$.
4. Else set $x_i = \mathbf{p}_i + \frac{\mathcal{P}}{|\mathcal{H} \cup \mathcal{M}|}$ for all $i \in \mathcal{H} \cup \mathcal{M}$.

Let $p_{\max} := \max_{i \in \mathcal{L}} \mathbf{p}_i$ and let $\mathcal{P} := \sum_{i \in \mathcal{L}} \mathbf{p}_i$. Then for every best response the following lemma holds.

Lemma 49. *If $\mathcal{L} \neq \emptyset$, then for any best response $\mathbf{x}$ of the row player against strategy $\mathbf{y}$ it holds that $\|\mathbf{x} - \mathbf{p}\|_\infty \geq p_{\max}$. Else $\mathbf{p}$ is the best response.*

Proof. Using similar arguments as in Lemma 36, it can be proven that if there are no pure strategies i and k such that $R_k\mathbf{y} - R_i\mathbf{y} - d_r < 0$ then any shifting of probability decreases the utility of the row player. Thus, the best response of the player is $\mathbf{p}$. On the other hand, if there are strategies i and k such that $R_k\mathbf{y} - R_i\mathbf{y} - d_r > 0$, then the utility of the row player increases if all the probability from strategy i is shifted to pure strategy k. The set $\mathcal{L}$ contains all these pure strategies. Let $\mathsf{j} \in \mathcal{L}$ be the pure strategy that defines $p_{\max}$. Then, all the $p_{\max}$ probability can be shifted from j to the a pure strategy in $\mathcal{H}$, i.e. a pure strategy that yields the highest payoff, and strictly increase the utility of the player. Thus, the strategy j is played with zero probability and the claim follows. □

In what follows assume that $\mathcal{L} \neq \emptyset$, hence $p_{\max} > 0$. From Lemma 49 follows that there is a best response where the strategy with the highest payoff is played with probability $\mathbf{p}_1 + p_{\max}$. Hence, it can be shifted up to $p_{\max}$ probability from pure strategies with lower payoff to each pure strategy with higher payoff, starting from the second pure strategy etc. After this shift of probabilities there will be a set of pure strategies that where each one is played with probability $\mathbf{p}_i + p_{\max}$ and possibly one pure strategy j that is played with probability less or equal to $\mathbf{p}_j$.

The question is whether more probability should be shifted from the low payoff strategies to strategies that yield higher payoff. The next lemma establishes that no pure strategy form $\mathcal{L}$ is played with positive probability in any best response against $\mathbf{y}$.

Lemma 50. *In every best response against strategy* $\mathbf{y}$ *all pure strategies* $i \in \mathcal{L}$ *are played with zero probability.*

Proof. Let K denote the set of pure strategies that are played with positive probability after the first shifting of probabilities. Without loss of generality assume that each strategy $i \in K$ is played with probability $\mathbf{p}_i + p_{\max}$. Then the utility of the player under this strategy is equal to $U = \sum_{i \in K}(\mathbf{p}_i + p_{\max}) \cdot R_i\mathbf{y} - d_r \cdot p_{\max}$. For the sake of contradiction, assume that there is one strategy j from $\mathcal{L}$ that belongs to K. Suppose that probability δ is shifted from the strategy j to the first pure strategy. Then the utility for the player is equal to $U + \delta(R_1\mathbf{y} - R_{\mathrm{j}}\mathbf{y} - d_r) > U$, since by the definition of $\mathcal{L}$ we have that $R_1\mathbf{y} - R_{\mathrm{j}}\mathbf{y} - d_r > 0$. Thus, the utility of the player is increasing if probability is shifted. Notice that the analysis holds even if the penalty is $p_{\max} + \delta$ instead of $p_{\max}$, thus the claim follows. □

Thus, all the probability $\mathcal{P}$ from strategies from $\mathcal{L}$ should be shifted to strategies that yield higher payoff. The question now is what is the optimal way to distribute that probability over the strategies with the higher payoff. It is not hard to see that distributing this probability uniformly over the strategies in $\mathcal{H}$ minimizes the penalty the player suffers. Furthermore, it is easy to see that the maximum amount of probability is shifted to strategies in $\mathcal{H}$. Next we prove that if $\mathcal{P} \geq p_{\max} \cdot (|\mathcal{H}| + |\mathcal{M}|)$ then $\mathcal{P}$ is uniformly distributed over the pure strategies in $\mathcal{H} \cup \mathcal{M}$.

Lemma 51. *If* $\mathcal{P} \geq p_{\max} \cdot (|\mathcal{H}| + |\mathcal{M}|)$ *then there is a best response where the probability* $\mathcal{P}$ *is uniformly distributed over the pure strategies in* $\mathcal{H} \cup \mathcal{M}$.

Proof. Let $|\mathcal{H}| + |\mathcal{M}| = k$ and $S = \mathcal{P} - k \cdot p_{\max}$. Let

$$U = \sum_{i \in \mathcal{H} \cup \mathcal{M}} (\mathbf{p}_i + p_{\max} + \frac{S}{k}) R_i\mathbf{y} - d_r(p_{\max} + \frac{S}{k})$$

be the utility when the probability S is distributed uniformly over all pure strategies in $\mathcal{H} \cup \mathcal{M}$. Furthermore, let U' be the utility when $\delta > 0$ probability is shifted from a pure strategy j to the first pure strategy that yields the highest payoff. Then $U' = U + \delta(R_1\mathbf{y} - R_j\mathbf{y} - d_r)$, but $R_1\mathbf{y} - R_j\mathbf{y} - d_r \leq 0$ since $j \in \mathcal{H} \cup \mathcal{M}$. The claim follows. □

Approximation Algorithm

Using the previous analysis the correctness of the algorithm follows.

Note that, using similar arguments as in Lemma 36 the next lemma can be proved.

Lemma 52. *If $d_r \geq 1$, then* $\mathbf{p}$ *is a dominant strategy.*

Furthermore, the combination of Lemma 52 with the fact that best responses can be computed in polynomial time gives the next theorem.

Theorem 20. *In biased games with L_∞ penalty functions and $\max\{d_r, d_c\} \geq 1$, an equilibrium can be computed in polynomial time.*

Again, we can see that there is a connection between the equilibria of the distance biased game and the well supported Nash equilibria (WSNE) of the underlying bimatrix game. The following Theorem can be proven in a similar way as the Theorem 16.

Lemma 53. *Let $\mathcal{B} = \big(R, C, \mathfrak{b}_r(\mathbf{x}, \mathbf{p}), \mathfrak{b}_c(\mathbf{y}, \mathbf{q}), d_r, d_c\big)$ be a distance biased game with L_∞ penalties and let $d := \max\{d_r, d_c\}$. Any equilirbium of $\mathcal{B}$ is a d-WSNE for the bimatrix game (R, C).*

Approximation Algorithm

For the quality of approximation, we can reuse the results that we proved for the L_1 penalty. Lemma 37 applies unchanged. For Lemma 38, we observe that $d_c \cdot \mathfrak{b}_c(\mathbf{y}^*, \mathbf{q}) \leq 1$ when the penalty $\mathfrak{b}_c(\mathbf{y}^*, \mathbf{q})$ is the L_∞ norm, since for this case it holds $\|\mathbf{y}^* - \mathbf{q}\|_\infty \leq 1$ and it is assumed that $d_c \leq 1$. Thus, we have the following theorem.

Theorem 21. *In biased games with L_∞ penalties a 2/3-equilibrium can be computed in polynomial time.*

6.6.5 Proof that $\mathbf{y}^{*^T}\mathbf{y}^* - 2\mathbf{y}^*_k\mathbf{q}_k \leq 1 - 2\mathbf{q}_k$.

Proof. Notice from the Equation (6.7) that for all i we get $\mathbf{y}_i = \mathbf{y}_k + \frac{1}{2}(\alpha_i - \alpha_k)$. Using that, we can write the term $\mathbf{y}^T\mathbf{y} = \sum_i \mathbf{y}_i^2$ as follows for a strategy $\mathbf{y}$ with

support size s.

$$\begin{aligned}\sum_{i=1}^{s}\mathbf{y}_i^2 &= \mathbf{y}_i^2 + \sum_{i\neq k}\mathbf{y}_i^2 \\ &= \mathbf{y}_k^2 + \sum_{i\neq k}\left(\mathbf{y}_k + \frac{1}{2}(\alpha_i - \alpha_k)\right)^2 \\ &= s\mathbf{y}_k^2 + \Big(\sum_{i\neq k}(\alpha_i - \alpha_k)\Big)\mathbf{y}_k + \frac{1}{4}\sum_{i\neq k}(\alpha_k - \alpha_i)^2.\end{aligned}$$

Then, we can see that ${\mathbf{y}^*}^T\mathbf{y} - 2{\mathbf{y}_k^*}^T\mathbf{q}_k$ is increasing as $\mathbf{y}_k^*$ increases, since we know from Lemma 43 that $\mathbf{y}_k^* > 0$. This becomes clear if we take the partial derivative of ${\mathbf{y}^*}^T\mathbf{y}^* - 2\mathbf{y}_k^*\mathbf{q}_k$ with respect to $\mathbf{y}_k^*$ which is equal to

$$\begin{aligned}2s\mathbf{y}_k^* + \sum_{i\neq k}(\alpha_i - \alpha_k) - 2\mathbf{q}_k &= 2s\mathbf{y}_k^* + \sum_{i\neq k}2(\mathbf{y}_i^* - \mathbf{y}_k^*) - 2\mathbf{q}_k \quad \Big(\text{since } \mathbf{y}_i = \mathbf{y}_k + \frac{1}{2}(\alpha_i - \alpha_k)\Big) \\ &= 2s\mathbf{y}_k^* + 2\sum_{i\neq k}\mathbf{y}_i^* - 2(s-1)\mathbf{y}_k^* - 2\mathbf{q}_k \\ &= 2\sum_{i=1}^{s}\mathbf{y}_i^* - 2\mathbf{q}_k \\ &= 2 - 2\mathbf{q}_k \\ &\geq 0 \quad (\text{since } \mathbf{y}_k^* > 0).\end{aligned}$$

Thus, the value of ${\mathbf{y}^*}^T\mathbf{y}^* - 2\mathbf{y}_k^*\mathbf{q}_k$ is maximized when $\mathbf{y}_k^* = 1$ and our claim follows. □

6.7 Open Questions

Several open questions stem from this chapter. The most important one is to understand the exact computational complexity of equilibrium computation in Lipschitz and penalty games. Another interesting feature is that we cannot verify efficiently in all penalty games whether a given strategy profile is an equilibrium, and so it seems questionable whether PPAD can capture the full complexity of penalty games. On the other side, for the distance biased games that we studied in this chapter, we have shown that we can decide in polynomial time if a strategy profile is an equilibrium. Is the equilibrium computation problem PPAD-complete for the classes of games we studied? Are there any subclasses of penalty games, e.g. when the underlying normal form game is zero sum, that are easy to solve?

Another obvious direction is to derive better polynomial time approximation guarantees for distance biased games. We believe that the optimization approach

used by [66] and [24] might tackle this problem. Under the L_1 penalties the analysis of the steepest descent algorithm may be similar to [24] and therefore we may be able to obtain a constant approximation guarantee similar to the bound of 0.5 that was established in Chapter 4. The other known techniques that compute approximate Nash equilibria [7] and approximate well supported Nash equilibria [15, 30, 53] solve a zero sum bimatrix game in order to derive the approximate equilibrium, and there is no obvious way to generalise this approach in penalty games.

Chapter 7

Conclusions

In this thesis we studied algorithms for computing approximate equilibria in several classes of games.

In Chapter 3 we have developed a new technique for computing approximate Nash equilibria, and approximate well-supported Nash equilibria. This new technique has allowed us to improve upon the best known results in multiple settings. For well-supported Nash equilibria, we have presented a polynomial-time algorithm for finding a 0.6528-WSNE, and we have shown how to implement it in a communication efficient manner, and a query efficient manner, improving upon the best known results in those settings. For approximate Nash equilibria, our techniques obtain a 0.382-NE and, again, we showed how this can be carried out in a communication efficient manner, improving the best known results in that setting. Several open questions stem from our paper. The most obvious one is to improve the derived bounds. Another important question is to derive lower bounds for all the problems studied in this chapter.

In Chapter 4 we presented a polynomial time algorithm that, for every δ in the range $0 < \delta \leq 0.5$, finds a $(0.5+\delta)$-Nash equilibrium of a polymatrix game in time polynomial in the input size and $\frac{1}{\delta}$. Note that our approximation guarantee *does not depend on the number of players,* which is a property that was not previously known to be achievable for polymatrix games, and still cannot be achieved for general strategic form games. As it was explained at the end of this Chapter, there are several interesting open questions. Among the most interesting ones, given the result of [63], is to derive a lower bound on the approximability of the computation of approximate equilibria in polymatrix games.

In Chapter 6 we studied games with infinite action spaces, and non-linear payoff functions. We have shown that Lipschitz continuity of the payoff function can be exploited to provide approximation algorithms. For Lipschitz games, Lipschitz

continuity of the payoff function allows us to provide an efficient algorithm for finding approximate equilibria. For penalty games, Lipschitz continuity of the penalty function allows us to provide a QPTAS. Finally, we provided strongly polynomial algorithms for computing approximate equilibria for L_1, L_2^2, and L_∞ distance biased games. Among all the open questions described in the end of the Chapter, maybe the most interesting question is to pin down exactly the computational complexity of these games.

Bibliography

[1] Eleftherios Anastasiadis and Argyrios Deligkas. Minmax heterogeneous facility location games. 2016. Working paper.

[2] Yaron Azrieli and Eran Shmaya. Lipschitz games. *Mathematics of Operations Research*, 38(2):350–357, 2013.

[3] Yakov Babichenko. Best-reply dynamics in large binary-choice anonymous games. *Games and Economic Behavior*, 81:130–144, 2013.

[4] Yakov Babichenko, Siddharth Barman, and Ron Peretz. Simple approximate equilibria in large games. In *ACM Conference on Economics and Computation, EC '14*, pages 753–770, 2014.

[5] Siddharth Barman. Approximating Nash equilibria and dense bipartite subgraphs via an approximate version of caratheodory's theorem. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 361–369, 2015.

[6] Emile Borel. La theorie du jeu at les equations integrales a noyau symmetrique. *Comptes Rendus de l'Academie des Sciences*, 173:1304–1308, 1921.

[7] Hartwig Bosse, Jaroslaw Byrka, and Evangelos Markakis. New algorithms for approximate Nash equilibria in bimatrix games. *Theoretical Computer Science*, 411(1):164–173, 2010.

[8] Patrick Briest, Paul W. Goldberg, and Heiko Röglin. Approximate equilibria in games with few players. *CoRR*, abs/0804.4524, 2008.

[9] Yang Cai and Constantinos Daskalakis. On minmax theorems for multiplayer games. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 217–234, 2011.

[10] Ioannis Caragiannis, David Kurokawa, and Ariel D. Procaccia. Biased games. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 609–615, 2014.

[11] Gretchen B. Chapman and Eric J. Johnson. Anchoring, activation, and the construction of values. *Organizational Behavior and Human Decision Processes*, 79(2):115 – 153, 1999.

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

[13] Xi Chen, Dimitris Paparas, and Mihalis Yannakakis. The complexity of non-monotone markets. In *Symposium on Theory of Computing Conference, STOC'13*, pages 181–190, 2013.

[14] Vincent Conitzer and Tuomas Sandholm. Communication complexity as a lower bound for learning in games. In *Proceedings of the Twenty-first International Conference on Machine Learning ICML*, pages 185–192, 2004.

[15] Artur Czumaj, Argyrios Deligkas, Michail Fasoulakis, John Fearnley, Marcin Jurdzinski, and Rahul Savani. Distributed methods for computing approximate equilibria. In *Web and Internet Economics - 10th International Conference, WINE*, 2016.

[16] Artur Czumaj, Michail Fasoulakis, and Marcin Jurdzinski. Approximate well-supported nash equilibria in symmetric bimatrix games. In *Algorithmic Game Theory - 7th International Symposium, SAGT*, pages 244–254, 2014.

[17] George B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

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

[19] Constantinos Daskalakis, Aranyak Mehta, and Christos H. Papadimitriou. Progress in approximate nash equilibria. In *Proceedings 8th ACM Conference on Electronic Commerce (EC-2007)*, pages 355–358, 2007.

[20] Constantinos Daskalakis, Aranyak Mehta, and Christos H. Papadimitriou. A note on approximate Nash equilibria. *Theoretical Computer Science*, 410(17):1581–1588, 2009.

[21] Constantinos Daskalakis and Christos H. Papadimitriou. Approximate nash equilibria in anonymous games. *Journal of Economic Theory*, 156:207–245, 2015.

[22] Joyee Deb and Ehud Kalai. Stability in large Bayesian games with heterogeneous players. *Journal of Economic Theory*, 157(C):1041–1055, 2015.

[23] Argyrios Deligkas, John Fearnley, Tobenna Peter Igwe, and Rahul Savani. An empirical study on computing equilibria in polymatrix games. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 186–195, 2016.

[24] Argyrios Deligkas, John Fearnley, Rahul Savani, and Paul Spirakis. Computing approximate Nash equilibria in polymatrix games. In *Web and Internet Economics - 10th International Conference, WINE*, pages 58–71, 2014.

[25] Argyrios Deligkas, John Fearnley, and Paul G. Spirakis. Lipschitz continuity and approximate equilibria. In *9th International Symposium on Algorithmic Game Theory, SAGT*, pages 15–26, 2016.

[26] Argyrios Deligkas, George B. Mertzios, and Paul G. Spirakis. On the complexity of weighted greedy matchings. *CoRR*, abs/1602.05909, 2016.

[27] Kousha Etessami and Mihalis Yannakakis. On the complexity of nash equilibria and other fixed points. *SIAM J. Comput.*, 39(6):2531–2597, 2010.

[28] Argyrios Deligkas John Fearnley and Rahul Savani. Inapproximability results for approximate nash equilibria. In *Web and Internet Economics - 10th International Conference, WINE*, 2016.

[29] John Fearnley, Martin Gairing, Paul W. Goldberg, and Rahul Savani. Learning equilibria of games via payoff queries. In *ACM Conference on Electronic Commerce, EC '13*, pages 397–414, 2013.

[30] John Fearnley, Paul W. Goldberg, Rahul Savani, and Troels Bjerre Sørensen. Approximate well-supported Nash equilibria below two-thirds. In *International Symposium on Algorithmic Game Theory, SAGT*, pages 108–119, 2012.

[31] John Fearnley, Tobenna Peter Igwe, and Rahul Savani. An empirical study of finding approximate equilibria in bimatrix games. In *Symposium on Experimental Algorithms, SEA*, 2015.

[32] John Fearnley and Rahul Savani. Finding approximate Nash equilibria of bimatrix games via payoff queries. In *ACM Conference on Electronic Commerce, EC '14*, pages 657–674, 2014.

[33] Uriel Feige and Inbal Talgam-Cohen. A direct reduction from k-player to 2-player approximate Nash equilibrium. In *International Symposium on Algorithmic Game Theory, SAGT*, pages 138–149, 2010.

[34] Amos Fiat and Christos H. Papadimitriou. When the players are not expectation maximizers. In *International Symposium on Algorithmic Game Theory, SAGT*, pages 1–14, 2010.

[35] Martin Gairing and Rahul Savani. Computing stable outcomes in hedonic games. In *Symposium on Algorithmic Game Theory, SAGT*, pages 174–185, 2010.

[36] Sam Ganzfried and Tuomas Sandholm. Game theory-based opponent modeling in large imperfect-information games. In *10th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '11, pages 533–540, 2011.

[37] Sam Ganzfried and Tuomas Sandholm. Safe opponent exploitation. In *ACM Conference on Electronic Commerce*, EC '12, pages 587–604, 2012.

[38] Paul W. Goldberg and Arnoud Pastink. On the communication complexity of approximate Nash equilibria. *Games and Economic Behavior*, 85:19–31, 2014.

[39] Paul W. Goldberg and Aaron Roth. Bounds for the query complexity of approximate equilibria. In *ACM Conference on Electronic Commerce, EC '14*, pages 639–656, 2014.

[40] Srihari Govindan and Robert Wilson. Computing Nash equilibria by iterated polymatrix approximation. *Journal of Economic Dynamics and Control*, 28(7):1229–1241, 2004.

[41] Srihari Govindan and Robert Wilson. A decomposition algorithm for n-player games. *Economic Theory*, 42(1):97–117, 2010.

[42] Mingyu Guo and Argyrios Deligkas. Revenue maximization via hiding item attributes. In *IJCAI, Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 157–163, 2013.

[43] Mingyu Guo, Argyrios Deligkas, and Rahul Savani. Increasing VCG revenue by decreasing the quality of items. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 705–711, 2014.

[44] Sergiu Hart and Yishay Mansour. How long to equilibrium? the communication complexity of uncoupled equilibrium procedures. *Games and Economic Behavior*, 69(1):107–126, 2010.

[45] Sébastien Hémon, Michel de Rougemont, and Miklos Santha. Approximate nash equilibria for multi-player games. In *International Symposium on Algorithmic Game Theory, SAGT*, pages 267–278, 2008.

[46] Joseph T. Howson. Equilibria of polymatrix games. *Management Science*, 18(5):pp. 312–318, 1972.

[47] Michael Johanson, Martin Zinkevich, and Michael Bowling. Computing robust counter-strategies. In J.C. Platt, D. Koller, Y. Singer, and S.T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 721–728. Curran Associates, Inc., 2008.

[48] Howson Joseph and Rosenthal Robert. Bayesian equilibria of finite two-person games with incomplete information. *Management Science*, 21(3):pp. 313–315, 1974.

[49] Daniel Kahneman. Reference points, anchors, norms, and mixed feelings. *Organizational Behavior and Human Decision Processes*, 51(2):296–312, 1992.

[50] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, pages 302–311, 1984.

[51] Leonid Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademiia Nauk SSSR*, 244:1093 – 1096, 1979.

[52] Spyros C. Kontogiannis and Paul G. Spirakis. Well supported approximate equilibria in bimatrix games. *Algorithmica*, 57(4):653–667, 2010.

[53] Spyros C. Kontogiannis and Paul G. Spirakis. Well supported approximate equilibria in bimatrix games. *Algorithmica*, 57(4):653–667, 2010.

[54] M.K. Kozlov, Sergei Tarasov, and Leonid Khachiyan. The polynomial solvability of convex quadratic programming. {*USSR*} *Computational Mathematics and Mathematical Physics*, 20(5):223 – 228, 1980.

[55] Carlton Lemke. Bimatrix equilibrium points and mathematical programming. *Management Science*, 11(7):pp. 681–689, 1965.

[56] Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Playing large games using simple strategies. In *Conference on Electronic Commerce (EC'03)*, pages 36–41, 2003.

[57] Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Playing large games using simple strategies. In *EC*, pages 36–41, 2003.

[58] Marios Mavronicolas and Burkhard Monien. The complexity of equilibria for risk-modeling valuations. *Theoretical Computer Science*, 634:67–96, 2016.

[59] Roger Myerson. *Game Theory Analysis of Conflict*. Princeton University Press, 1997.

[60] John Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

[61] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994.

[62] J. B. Rosen. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica*, 33(3):pp. 520–534, 1965.

[63] Aviad Rubinstein. Inapproximability of nash equilibrium. pages 409–418, 2015.

[64] Aviad Rubinstein. Settling the complexity of computing approximate two-player nash equilibria. In *Symposium on Foundations of Computer Science FOCS*, 2016. To appear.

[65] Rahul Savani and Bernhard von Stengel. Exponentially many steps for finding a nash equilibrium in a bimatrix game. In *45th Symposium on Foundations of Computer Science FOCS*, pages 258–267, 2004.

[66] Haralampos Tsaknakis and Paul G. Spirakis. An optimization approach for approximate Nash equilibria. *Internet Mathematics*, 5(4):365–382, 2008.

[67] Amos Tversky and Daniel Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131, 1974.

[68] John von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[69] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.