Equilibrium Computation in Games and Strategic Aspects of Bitcoin Mining

Francisco Javier Marmolejo Cossío

Hertford College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2019

This thesis is dedicated to the memory of my grandfathers:
Juan Cossío Acosta and Manuel Marmolejo Dávila

Acknowledgements

First and foremost, I would like to thank my advisor Paul Goldberg for his dedicated guidance over the past couple of years. It has been a privilege to work with him, and this thesis would not exist without his encouragement and support.

I am overwhelmed with a sense of gratitude towards my family, who have always supported me in all my endeavours. Thank you to my parents, Olivia and Francisco, for teaching me to always push for my goals while making time for loved ones. To my brothers, José and Juan, thank you for always finding the time to make me laugh and remind me we'll always be the same kids from many years ago.

I want to thank my wonderful wife Rhea for her unyielding love and support. Even while continents away, you have always made me feel loved and at home. I feel truly blessed to have you around Oxford at last, and to be a part of your family.

A big thank you to all of the people who have made my time at Oxford unforgettable. To Zach – for being not just a housemate but a truly great friend; Jericho will always be home with you around. To all the regular denizens of the CS cave, room 224: Ninad, Philip, Alex, Giorgos, Aris, Edwin and Andrius – for all the laughs, support, and of course, your friendship over the years. I could not have picked a better group of friends to share an office with. To Citlali, Poncho and Alberto – for all the unforgettable breakfasts and the feeling of family away from home. To Varun – for always being a great friend and fellow Beatles fan. To Vincenzo – for being one of my oldest friends, and somehow always being around the corner, wherever that corner may be in the world. To Ryan – for the much needed occasional lazy, wholesome Sunday. And finally, to Andrea, Mike, Swaraj, Akshay, and Radhika for your friendship over the years.

I also want to thank Jonathan Katz for hosting me for a wonderful summer at UMD, as well as everyone who made my visit memorable. In particular,

I want to thank the CP towers crew: Aaron, Kyle and Cameron for coming along with me on Finn and Jake's final adventure.

Finally, I am grateful to the following institutions for their generous financial support during my graduate studies: the Mexican National Council of Science and Technology (CONACyT), the San Luis Potosí Council of Science and Technology (COPOCyT), The Autonomous University of San Luis Potosí (UASLP), and the Computer Science department at the University of Oxford.

Abstract

The focus of this thesis is twofold: on one hand we study the query complexity of equilibrium computation in games, and on the other hand, we use equilibrium concepts from game theory as a tool to understand miner incentives in Bitcoin.

In terms of query complexity, we mostly focus on algorithms that have access to utility queries in large games and best response queries in bimatrix games. For the former, we demonstrate query-efficient *completely uncoupled dynamics* that achieve non-trivial approximate equilibria. For the latter, we reduce the problem of query-efficient approximate equilibrium computation to a natural geometric learning problem: approximately learning partitions of an n-dimensional simplex into disjoint convex polytopes via membership queries. Given this reduction we show query-efficient algorithms for the geometric problem, and ultimately provide an algorithm for computing ε-well-supported Nash equilibria in $m \times n$ bimatrix games with a query cost that is polynomial in $\log\left(\frac{1}{\varepsilon}\right)$ and $\max(m, n)$ provided that $\min(m, n)$ is constant. This leads to a polynomial query complexity algorithm for 2-player games, provided that one of the players has a constant number of strategies.

As for incentives in Bitcoin, we shed some light into how robust honest mining protocols are to the presence of strategic agents. Our focus is on the strategic aspects of both solo mining and pool mining in Bitcoin. For the former, we take a multiplayer approach and exhibit specific strategy profiles of multiple strategic miners that outperform honest mining, even if said miners would not be incentivised to be dishonest individually. This effectively renders the Bitcoin protocol less secure than previously thought. As for the latter, we propose a new mining pool protocol that is a randomised variant of the already-ubiquitous pay-per-last-N-shares (PPLNS) mining pool scheme in Bitcoin. Our pool protocol, randomised pay-per-last-N-shares (RPPLNS), enjoys the same desirable properties of PPLNS, but with the added benefit of an exponentially reduced state space required to maintain the protocol. More importantly, this reduced state space also allows us to prove robust guarantees against a richer class of strategic pool mining than before.

Contents

List of Figures

Chapter 1

Introduction

1.1 Games, Equilibria and Bitcoin

Game theory is the technical language of conflict and cooperation between interdependent agents, and although its historical roots lie in mathematics and economics, over the past few decades it has become fruitfully intertwined with the field of computer science. Indeed, with the advent of the internet and other large-scale decentralised technologies, game theory has informed the evolution of such systems and given computer scientists a lens with which to study such vast computational objects that are created and propagated by strategic agents. Conversely, computer science has also shed some light on the computational resources needed to find solution concepts in game theory. This thesis lies precisely in this dialectic between computer science and game theory. On one hand we use techniques from algorithmic complexity to study equilibrium concepts in game theory, and on the other hand, we study Bitcoin by explicitly computing equilibria pertaining to miner incentives in both the Bitcoin ecosystem at large and within mining pools.

One of the cornerstones of game theory is Nash's theorem [57], which establishes the existence of Nash equilibria, a stalemate of sorts, in all finite games. Though Nash equilibria have been used as a model for possible outcomes in strategic scenarios between players, explicitly finding an equilibrium presents many computational challenges, most notably the intractability of computing an equilibrium even in games with two players, as proven in [15] and [17]. In this thesis we also delve into the intricacies of computing equilibria, but rather than only focusing on time as a resource, we place an emphasis on the amount of relevant information about a game an algorithm needs in order to compute different notions of approximate Nash equilibria. The way we formalise how an algorithm obtains information as a resource is via the notion of a *query oracle* which returns a specific pre-defined piece of information about a game upon being queried by an algorithm. Since we treat information as a resource, we are interested in algorithms that make efficient use of an oracle, or rather have a low *query complexity*.

Precisely understanding the number of queries of a specific kind that are needed to obtain different solution concepts is not just an esoteric computational endeavour. There are further practical justifications for why it is reasonable to impose that algorithms be query-efficient. First of all, it is well-known that the full representation of a game can grow exponentially in the number of players involved, hence it can be infeasible to demand that an algorithm have the full representation of the game. In addition, agents themselves may not have a clear idea of how to quantify the utility they gain from specific scenarios of play. Finally, assuming that agents maximise utility, an algorithm may only witness indirect information about utilities in a game, such as best-response behaviour where agents myopically take their best actions in a given strategic scenario. Either way, it is an important and reasonable imposition that an algorithm succeed with this limited information.

Beyond the computational intricacies of equilibrium computation, we previously mentioned how game theory and in particular equilibria have proven to be fruitful tools in understanding the vast decentralised nature of the internet. These same tools are currently being used to understand miner behaviour in Bitcoin and other similar cryptocurrencies that have exploded in popularity over the past decade. Bitcoin is a decentralised, semi-anonymous and tamper-proof digital currency that maintains a public ledger via distributed consensus algorithms powered by blockchain technology. End users of the currency post transactions to the P2P network sustaining the protocol and said transactions are bundled into blocks by *miners*: agents tasked with the upkeep of the ledger. With respect to Bitcoin, the prescribed *longest chain rule* dictates that miners must bundle pending transactions into a block that also includes a single hash pointer to the end of the longest chain seen by the miner in their local view of the ledger. Furthermore, in order for a block to be valid, its hash must lie below a dynamically adjusted threshold. Hence, miners must expend computational resources to find valid blocks.

Indeed, in Bitcoin's proof-of-work consensus protocol the miner is essential. These agents take upon themselves the task of maintaining the ledger of transactions that form the foundation of Bitcoin. This upkeep is not done out of good will however, as the judicious design of Bitcoin pays miners for their efforts, with the aim of incentivising honest behaviour. Ultimately the incentives in Bitcoin are complex, and more importantly the security of the system itself lies in completely understanding agent incentives. In this thesis we shed some light into how robust honest mining protocols are to the presence of strategic agents.

1.2 Outline of the Thesis and Main Results

The thesis is composed of two major parts. In Chapters 2-4 we focus on the query complexity of computing equilibria in games with different query oracles, and in Chapters 5-7 we use equilibria as a tool to study strategic incentives in Bitcoin.

- In Chapter 2 we provide a formal introduction to game theory and query complexity. In particular, we delve into the subtleties of using different query oracles in equilibrium computation and we also give a brief overview of important previous results in the field and background literature.

- In Chapter 3 we study the computation of approximate equilibria in large games via utility queries. In a γ-large game, deviations of a single agent change other players' utilities by at most γ. For large, n-player binary action games we show that for $\gamma = \frac{1}{n}$ we can obtain a non-trivial $\varepsilon = \frac{1}{8}$ approximate Nash equilibrium by a novel completely uncoupled continuous-time dynamic using $O(\log n)$ rounds/queries. We further discretise this dynamic and extend our results to multiple actions and different values of γ to obtain non-trivial approximation guarantees.

- In Chapter 4 we study the computation of approximate well-supported Nash equilibria in bimatrix games via best response queries. The first main contribution is a reduction of approximate equilibrium computation in this setting to the related geometric objective of using a membership query oracle to approximately learn a partition of an m-dimensional simplex into a disjoint covering of n convex polytopes. With this reduction in hand, we provide query efficient algorithms for the aforementioned geometric objective, and ultimately show that in $m \times n$ bimatrix games, we can efficiently compute an ε well-supported Nash equilibrium with a query cost that is polynomial in $\log\left(\frac{1}{\varepsilon}\right)$ and $\max(m, n)$, provided that $\min(m, n)$ is constant. This leads to a polynomial query complexity algorithm for bimatrix player games, provided that one of the players has a constant number of strategies. Furthermore, we show that if we use a more powerful pairwise comparison oracle instead of best response oracles, we can in fact compute an ε-WSNE with a query usage polynomial in m, n, and $\log\left(\frac{1}{\varepsilon}\right)$. In addition, we also partially extend our results to the multiplayer setting.

- In Chapter 5 we give an overview of the Bitcoin protocol and related work in the area of strategic incentives in Bitcoin.

- In Chapter 6 we study strategic mining within the Bitcoin protocol at large by exhibiting a specific truncation of the celebrated selfish mining (SM) strategy of

[22], which we call semi-selfish mining (SSM). SSM not only outperforms honest mining as in SM, but its reduced state space also allows us to compute full utilities if multiple (non-colluding) miners are employing SSM to various degrees. The analysis of SM by Eyal and Sirer, as well as in follow-up work, considers a *single* deviating miner (who may control a large fraction of the hash power in the network) interacting with a remaining pool of honest miners. With SSM we extend this analysis to the case where there are *multiple* (non-colluding) strategic miners. We find that in this setting, specific deviations from honest mining by multiple strategic agents can outperform honest mining, even if, individually, miners would not be incentivised to deviate from the honest protocol. This previous point effectively renders the Bitcoin protocol to be less secure than previously thought.

- In Chapter 7 we study incentives in pool mining and present a novel twist to the already popular "Pay-per-last-N-shares" (PPLNS) mining pool scheme used by a majority of the Bitcoin network. By suitably randomising PPLNS we are able to maintain its strengths (fairness, variance reduction, robustness to pool hopping) while reducing the underlying memory usage of the protocol and proving robustness guarantees against a richer class of strategic mining than before.

1.3 List of Papers

The majority of results in this thesis are from the following series of papers:

- Paul W. Goldberg, Francisco J. Marmolejo-Cossío, and Zhiwei Steven Wu
  Logarithmic Query Complexity for Approximate Nash Computation in Large Games [28]
  Theory of Computing Systems, 63(1): 26-53 (2019).
  Full version in CoRR: abs/1807.06170.

- Paul W. Goldberg, and Francisco J. Marmolejo-Cossío
  Learning Convex Partitions and Computing Game-theoretic Equilibria from Best Response Queries [27]
  Proceedings of *14th Conference on Web and Internet Economics (WINE 2018)*.
  To appear in special issue of *ACM Transactions on Economics and Computation*.
  Full version in CoRR: abs/1807.06170.

- Francisco J. Marmolejo-Cossío, Eric Brigham, Benjamin Sela, and Jonathan Katz
  Multiple (Semi)-Selfish Miners in Bitcoin[51]
  Proceedings of *1st Advances in Financial Technology (AFT 2019)*.
  Full version in CoRR: abs/1906.04502.

- Philip Lazos, Francisco J. Marmolejo-Cossío, Xinyu Zhou, and Jonathan Katz
  RPPLNS: Pay-per-last-N-shares with a Twist
  In submission.

In addition, for completeness we also mention the following work whose results are not contained in this thesis:

- Georgios Birmpas, Philip Lazos, Francisco J. Marmolejo-Cossío, Elias Koutsoupias
  Fairness and Efficiency in Dag-based Cryptocurrencies [11]
  To appear in proceedings of *24th International Conference on Financial Cryptography and Data Security (FC 2020)*.
  Full version in CoRR: abs/1910.02059.

Chapter 2

Query Oracle Models

2.1 Game Theory Preliminaries

For the entirety of this thesis we focus on normalised finite games. A normalised finite game, G, consists of n agents, identified by elements in $[n] = \{1, ..., n\}$, where each agent has a finite action set $\mathcal{A}_i$ of cardinality k_i and a utility function $U_i : \prod_{i=1}^{n} \mathcal{A}_i \to [0, 1]$. We often use $\mathcal{A}_i = \{a_0, ..., a_{k_i-1}\}$ or $\mathcal{A}_i = \{0, ..., k_i - 1\}$ interchangeably to refer to the actions of the i-th player. In general, we let $\mathcal{A} = \prod_{i=1}^{n} \mathcal{A}_i$ be the space of all pure strategy profiles of all players, and for the i-th player we let $\mathcal{A}_{-i} = \prod_{j \neq i} \mathcal{A}_j$ be the space of all pure strategies of agents other than i. For any $a \in \mathcal{A}$, we let $a_{-i} \in \mathcal{A}_{-i}$ denote a specific pure strategy profile of all players other than i within a. This latter notation is of great convenience, for then we can specify $(a', a_{-i}) \in \mathcal{A}$ as the pure strategy profile of all players where player i deviates from a by playing a' and all other players maintain a_{-i}.

For the i-th player we let $\Delta(\mathcal{A}_i)$ denote the space of all probability distributions over the elements of $\mathcal{A}_i$. It is straightforward to see that if $|\mathcal{A}_i| = k_i$, then $\Delta(\mathcal{A}_i) = \{(p_1, ..., p_{k_i-1}) \mid p_1, ..., p_{k_i-1} \geq 0, \sum_{i=1}^{k_i-1} p_i \leq 1\}$, where p_i represents the probability mass placed on a_i and $p_0 = 1 - \sum_{i=1}^{k_i-1}$ represents the probability mass placed on a_0. This latter set can in fact be identified with the $(k_i - 1)$-simplex, which we denote by $\Delta^{k_i-1} \subseteq \mathbb{R}^{k_i-1}$. With this in hand we let $\Delta(\mathcal{A}) = \prod_{i=1}^{n} \Delta(\mathcal{A}_i) \cong \prod_{i=1}^{n} \Delta^{k_i-1}$ be the space of all product probability distributions over action profiles of agents. We call any $x_i \in \Delta(\mathcal{A}_i)$ a *mixed strategy* for player i and note that x_i is in fact a vector $(x_{ij})_{j=1}^{k_i-1} \in \Delta^{k_i-1}$ such that $x_{ij} \geq 0$ for all j and $\sum_{j=1}^{k_i-1} x_{ij} \leq 1$. Finally, we let any $x = (x_i)_{i=1}^{n} \in \Delta(\mathcal{A})$ be a *mixed strategy profile* of all players where for an arbitrary player $i \in [n]$, we can write x as (x_i, x_{-i}) with $x_i \in \Delta(\mathcal{A}_i)$ and $x_{-i} \in \Delta(\mathcal{A}_{-i}) = \prod_{j \neq i} \Delta(\mathcal{A}_j)$. With the aforementioned definitions of mixed strategy profiles, we extend the domain of utility functions from $\mathcal{A}$ to $\Delta(\mathcal{A})$ by letting utility represent expected utility, i.e. $U_i(x) = \mathbb{E}_{a \sim x}(U_i(a))$ for any $x \in \Delta(\mathcal{A})$ when treated as a product distribution over $\mathcal{A}$.

We are ultimately interested in computing Nash equilibria and suitable approximations with query access to a given game G.

Definition 2.1 (Nash Equilibrium)**.** *Suppose that G is a finite n-person game. We say that a mixed profile $x \in \Delta(\mathcal{A})$ is a Nash equilibrium (NE) if for every player $i \in [n]$, it holds that for all $x'_i \in \Delta(\mathcal{A}_i)$, $U_i(x) \geq U_i(x'_i, x_{-i})$. Namely, no agent has any incentive to deviate from x.*

In what follows we also let $E_i : \Delta(\mathcal{A}_{-i}) \to [0,1]$ be the upper envelope function of any player $i \in [n]$ that consists of $E_i(x_{-i}) = \max_{x' \in \Delta(\mathcal{A}_i)} U_i(x', x_{-i})$, representing the maximal utility player i can earn against x_{-i}. Notice that since any $x \in \Delta(\mathcal{A}_i)$ is in fact a mixture over pure strategies in $\mathcal{A}_i$, we can write the upper envelope function as $E_i(x_{-i}) = \max_{a' \in \mathcal{A}_i} U_i(a', x_{-i})$. With this in hand, we can define the concept of *Best Responses* for a given player. We let $BR_i(x_{-i}) = \{a \in \mathcal{A}_i \mid U_i(a, x_{-i}) = E_i(x_{-i})\} \subseteq \mathcal{A}_i$ be the set of best responses player i has against mixed strategy x_{-i}. For a given $x_i \in \Delta(\mathcal{A}_i)$, we also say that the support of x_i is the set of strategies in $\mathcal{A}_i$ that have non-zero probability in x_i, and we denote this set by $S_i(x_i) \subseteq \mathcal{A}_i$. With this notation in hand, we can formulate a useful equivalent combinatorial formulation of a Nash equilibrium

Observation 1. *$x \in \Delta(\mathcal{A})$ is a NE if and only if for every player $i \in [n]$, whenever $x = (x_i, x_{-i})$, it follows that $S_i(x_i) \subseteq BR_i(x_{-i})$.*

As mentioned before, we also study suitable approximations to the concept of a Nash equilibrium. We begin by relaxing the condition that no agent have any incentive to deviate in a NE to agents having a bounded incentive to deviate.

Definition 2.2 (ε-approximate Nash equilibrium)**.** *Suppose that G is a finite n-person game. We say that a mixed profile $x \in \Delta(\mathcal{A})$ is an ε-approximate Nash equilibrium (ε-ANE) if for every agent $i \in [n]$, it holds that for all $x'_i \in \Delta(\mathcal{A}_i)$, $U_i(x) \geq U_i(x'_i, x_{-i}) - \varepsilon$. Namely, no agent has more than ε incentive to deviate from x.*

We also modify the notion of best responses to account for an additive approximation of ε. For any ε we let $\varepsilon BR_i(x_{-i}) = \{a \in \mathcal{A}_i \mid U_i(a, x_{-i}) \geq E_i(x_{-i}) - \varepsilon\}$. This set consists of all actions $a \in \mathcal{A}_i$ that are at most ε from optimal for player i when faced against x_{-i}.

Definition 2.3 (ε-well-supported Nash equilibrium)**.** *Suppose that G is a finite n-person game. We say that a mixed profile $x \in \Delta(\mathcal{A})$ is an ε-well-supported Nash equilibrium (ε-WSNE) if for every agent $i \in [n]$, we can decompose $x = (x_i, x_{-i})$ and have $S_i(x_i) \subseteq \varepsilon BR_i(x_{-i})$.*

It is straightforward to see that the concept of an ε-WSNE is more stringent than that of an ε-ANE: if a mixed strategy profile $x \in \Delta(\mathcal{A})$ is an ε-WSNE, then it is forcibly an ε-ANE, whereas the converse statement does not hold in general.

We also briefly mention the notion of a correlated equilibrium and its relevant approximation. As we have seen before, a Nash equilibrium consists of a product distribution over player strategy profiles where no agent has an incentive to deviate. For correlated equilibria we generalise the definition to allow joint distributions over $\mathcal{A}$ that may not decompose into product distributions as in the case of a Nash equilibrium. The joint distribution can be thought of as a trusted randomised signal to act from a centralised source, and the equilibrium condition simply means that an agent does not have an incentive to deviate upon learning their signal. In the case of ε-correlated equilibria we simply impose that an agent has at most ε incentive to deviate upon learning their signal.

Definition 2.4 (Correlated Equilibria)**.** *Suppose that G is a finite n-person game and that x is a joint distribution over $\mathcal{A}$ which places mass $x(a)$ on each strategy profile $a \in \mathcal{A}$. We say that x is a* correlated equilibrium (CE) *if for every player $i \in [n]$, and all pure strategies, $j, k \in \mathcal{A}_i$,*

$$\sum_{a \in \mathcal{A} : a_i = j} x(a) \left(U_i(k, a_{-i}) - U_i(a) \right) \leq 0.$$

We say that it is an ε-correlated equilibrium (ε-CE) *if for every player $i \in [n]$ and any function $f : \mathcal{A}_i \to \mathcal{A}_i$,*

$$\sum_{a \in \mathcal{A} : a_i = j} x(a) \left(U_i(f(a_i), a_{-i}) - U_i(a) \right) \leq \varepsilon.$$

2.2 Query Complexity

The first half of the dissertation is composed of Chapters 3 and 4, and focuses on algorithms that compute equilibria in games by accessing limited information from the games themselves. Not only does this allow us to differentiate between different solution concepts, but it is also a practical assumption given the fact that the description of a game is in general exponential in the number of players. Ultimately, we treat information as a resource and aim to quantify how much information is necessary to compute an equilibrium in different settings.

The way we formalise how an algorithm accesses information about a given game is by *Query oracles*. We assume our algorithms have no prior knowledge of the game but can access information via queries to an oracle. For a given oracle, $\mathcal{Q}$, and family of games $\mathcal{G}$, we are interested in algorithms that compute a solution concept $\mathcal{S}$ over $G \in \mathcal{G}$ (e.g NE, ε-ANE, or ε-WSNE) with access to $\mathcal{Q}$. For any such algorithm, we say that

its *query complexity* is the number of calls to $\mathcal{Q}$ it makes during its execution. In the following paragraph we make this notion rigorous:

Suppose that $\mathcal{G}$ is a class of finite games where $\mathcal{G}(n,k)$ is the set of games $G \in \mathcal{G}$ that have at most n and players and such that $|\mathcal{A}_i| \leq k$ for all $i \in [n]$. Furthermore, let $\mathcal{A}_{\mathcal{S},\mathcal{Q}}$ be all algorithms that compute solution concept $\mathcal{S}$ for games in $\mathcal{G}$ with access to $\mathcal{Q}$. Finally, let $C_{\mathcal{Q}}(A,G)$ be the number of calls to $\mathcal{Q}$ an algorithm $A \in \mathcal{A}_{\mathcal{S},\mathcal{Q}}$ makes in its execution on a given $G \in \mathcal{G}$.

Definition 2.5 (Query Complexity)**.** *For any algorithm $A \in \mathcal{A}_{\mathcal{S},\mathcal{Q}}$ we let $QC_A(n,k) = \max_{G \in \mathcal{G}(n,k)} C_{\mathcal{Q}}(A,G)$. This is the worst case number of queries A makes in solving for $\mathcal{S}$ on a game in $\mathcal{G}(n,k)$ and it is denoted as the* query complexity *of A. In addition, we let $QC_{\mathcal{S},\mathcal{G},\mathcal{Q}}(n,k) = \min_{A \in \mathcal{A}_{\mathcal{S},\mathcal{Q}}} QC_A(n,k)$ denote the smallest query complexity of any $A \in \mathcal{A}_{\mathcal{S},\mathcal{Q}}$ over games in $\mathcal{G}(n,k)$ and call this function the* query complexity *of the solution concept $\mathcal{S}$ over games in $\mathcal{G}$ with query acces to $\mathcal{Q}$.*

In the definition of query complexity we have implicitly made an emphasis on the query usage as a function of the number of players in a game n, and their maximum number of actions k. Ultimately, we are interested in algorithms with a query complexity that has a low dependence on these parameters. When the solution concept involves an approximation of ε, we also aim for query complexities with a benign dependence on $1/\varepsilon$. What kind of information an oracle provides is crucial to the query complexity of a solution concept. In this section we give a brief overview of different types of natural query oracles.

Utility Queries

Given that we have defined games as a collection of action sets and utility functions for n agents, it is natural to consider a query oracle that returns specific utilities at a given strategy profile of all agents. For a *utility query*, specified by an action profile $a \in \mathcal{A}^n$, the query oracle returns $(U_i(a))_{i=1}^n$, the n-dimensional vector of payoffs to each player. Notice that we could have instead specified that the query oracle return the utility of a single specified agent, however it is trivial to notice that simulating the "all utilities" answer with this restricted oracle can be done by repeating the same single utility query for all agents. Ultimately, this means that distinguishing between an "all utilities" and "single utility" oracle introduces at most a multiplicative factor of n in the query complexity of an algorithm, hence we ignore this distinction and focus on the "all utilities" oracle.

Definition 2.6 (Utility Query Oracles)**.** *Suppose that G is a finite n player game. We say that $\mathcal{Q}_U : \mathcal{A} \to [0,1]^n$ is the pure strategy utility query oracle if for any $a \in \mathcal{A}$,*

$\mathcal{Q}_U(a) = (U_i(a))_{i=1}^n$. *We also say that* $\mathcal{Q}_M : \Delta(\mathcal{A}) \to [0,1]^n$ *is the mixed strategy utility oracle if for any* $x \in \Delta(\mathcal{A})$, $\mathcal{Q}_M(x) = (U_i(x))_{i=1}^n$.

It is important to note the distinction between $\mathcal{Q}_U$ and $\mathcal{Q}_M$. We will mainly focus on describing query-efficient algorithms that use $\mathcal{Q}_U$ rather than $\mathcal{Q}_M$. The reason for this is that as noted in [3], $\mathcal{Q}_M$ is not an interesting oracle model due to the fact that it is "too powerful". This latter point stems from the fact that inputs to the mixed strategy oracle can be of arbitrary precision, and for pure strategy utilities that arise from a finite precision discrete set (say bit strings of a fixed length), a single mixed strategy utility query can in fact recover all payoff information of a given game.

Of course we would like to recover information regarding payoffs to mixed strategies with $\mathcal{Q}_U$, as this is fundamental in the definition of equilibria. We naturally do so by sampling from $\mathcal{Q}_U$ according to mixed product distributions in $\Delta(\mathcal{A})$ and taking an empirical average of utilities. Such sampling is used extensively in Chapter 3, and the details of how to perform this sampling along with approximation guarantees can be found in Section 3.6.3. In more detail, for a given game G, let $k = \max_{i=1,\dots,n}(|\mathcal{A}_i|)$. We show that approximating $\mathcal{Q}_M(x)$ up to an additive error of β with a success probability of at least $1-\delta$ can be done with $\frac{64k^2}{\beta^3}\log(8n/\delta)$ samples from $\mathcal{Q}_U(a)$ where a is sampled according to $x \in \Delta(\mathcal{A})$.

Best Response Queries

In spite of being explicitly represented in the mathematical definition of a game, one could argue that in practice actions do not always give rise to clear-cut utilities, let alone the fact that agents may have other incentives at play, thus making it difficult to encapsulate utility into a single number. For this reason, we introduce the natural concept of best response queries. Using a best response oracle, an algorithm gains information regarding what best responses agents may have to a specific mixed strategy profile of all other agents in the game. Since this is a set-valued function, we distinguish between three different styles of best response query oracle each of differing strength:

Definition 2.7 (Best Response Query Oracles)**.** *Let* G *be an* n*-player game.*

- *There are* n *strong best response oracles,* $BR_1, \dots, BR_n$ *as per Section 2.1.*
- *Suppose each* $\mathcal{A}_i$ *has a strict ordering denoted by* $\prec_i$*. Then there are* n *lexicographic best response oracles,* $BR_i^\ell : \Delta(\mathcal{A}_{-i}) \to \mathcal{A}_i$ *for* $i = 1, \dots, n$*. Each of these is defined by* $BR_i^\ell(x) = \min_{\prec_i}(BR_i(x))$.
- *A family of adversarial best response oracles is a collection of* n *functions denoted* $BR_i^A : \Delta(\mathcal{A}_{-i}) \to \mathcal{A}_i$ *for* $i = 1, \dots, n$*. Each of these satisfies* $BR_i^A(x) \in BR_i(x)$.

It is straightforward to see that the aforementioned oracles are in decreasing order of strength, for the strong best response oracle can simulate any given lexicographic or adversarial oracle, and a lexicographic oracle is simply an example of an adversarial oracle. Therefore algorithms that successfully compute equilibria with an adversarial oracle for example, trivially extend to the other two.

Furthermore, it is straightforward to see that $\mathcal{Q}_M$ can trivially efficiently simulate BR_i for any $i \in [n]$. For a given $x_{-i} \in \Delta(\mathcal{A}_{-i})$, this is done by $k_i = |\mathcal{A}_i|$ calls to $\mathcal{Q}_M$ at (a', x_{-i}) for all $a' \in \mathcal{A}_i$ to determine which actions maximise expected utility for player i against x_{-i}. Thus any algorithm that computes equilibria via any family of best response oracles can be simulated with $\mathcal{Q}_M$ with at most a multiplicative overhead of $\max_{i\in[n]} k_i$ queries.

On the other hand, at a first glance it may seem that $\mathcal{Q}_U$ can simulate best response oracles in a similar fashion, but this is not the case. $\mathcal{Q}_U$ can in fact not even be used to efficiently simulate an adversarial best response oracle with high probability in the worst case. Intuitively, there are n-player games where an agent may be indifferent to all strategy profiles but one (out of the exponentially many pure strategy profiles of $\mathcal{A}$). In this scenario, distinguishing expected payoffs to mixed strategy profiles between such a game and one where the given agent is in fact indifferent to all profiles is impossible to do with high probability. We refer the interested reader to Appendix A for a full proof of this result.

Pairwise Comparison Queries

We briefly introduce an oracle that is intermediate between the strong best response oracle from above and $\mathcal{Q}_M$. A pairwise comparison oracle takes as input a player $i \in [n]$, two actions $a_j, a_k \in \mathcal{A}_i$ where $a_j \neq a_k$ and a mixed strategy profile $x_{-i} \in \Delta(\mathcal{A}_{-i})$ to answer whether action a_j weakly dominates action a_k for player i when all other players use x_{-i}.

Definition 2.8 (Pairwise Comparison Queries)**.** *Suppose that G is a finite n player game. We let $\mathcal{Q}_{PW}$ be the pairwise comparison query oracle. Suppose that $i \in [n]$, $a_j, a_k \in \mathcal{A}_i$ such that $a_j \neq a_k$, and $x_{-i} \in \Delta(\mathcal{A}_{-i})$:*

$$\mathcal{Q}_{PW}(i, a_j, a_k, x_{-i}) = \begin{cases} 1, & \text{if } U_i(a_j, x_{-i}) \geq U_i(a_k, x_{-i}), \\ 0 & \text{otherwise.} \end{cases}$$

Although it seems that we have imposed lexicographic tie-breaking, we can in fact ascertain whether $U_i(a_j, x_{-i})) = U_i(a_k, x_{-i})$ by making the queries $\mathcal{Q}_{PW}(i, a_j, a_k, x_{-i})$ and $\mathcal{Q}_{PW}(i, a_k, a_j, x_{-i})$, and noting whether they both return 1. Once again, it is not

difficult to see that we can efficiently simulate $\mathcal{Q}_{PW}$ with $\mathcal{Q}_M$ but not with $\mathcal{Q}_U$, since the same adversarial example presented in Appendix A holds. Furthermore, we also notice that simulating BR_i for any player $i \in [n]$ can be done with $2\binom{k_i}{2}$ queries to $\mathcal{Q}_{PW}$ by establishing a complete ranking of all actions in $\mathcal{A}_i$ against a given mixed strategy. As a consequence, any query-efficient algorithms in the best response oracle setting translate to query-efficient algorithms in the pairwise comparison oracle setting.

ε-Best Response Queries

We have already introduced the notion of approximate best responses, we simply formalise this into oracle models akin to those regarding best responses.

Definition 2.9 (ε-Best Response Query Oracles)**.** *Let G be an n-player game.*

- *There are n strong ε-best response oracles, $\varepsilon BR_1, ..., \varepsilon BR_n$ as per Section 2.1.*
- *Suppose each $\mathcal{A}_i$ has a strict ordering denoted by $\prec_i$. Then there are n lexicographic ε-best response oracles, $\varepsilon BR_i^\ell : \Delta(\mathcal{A}_{-i}) \to \mathcal{A}_i$ for $i = 1, ..., n$. Each of these is defined by $\varepsilon BR_i^\ell(x) = \min_{\prec_i}(\varepsilon BR_i(x))$.*
- *A family of adversarial ε-best response oracles is a collection of n functions denoted $\varepsilon BR_i^A : \Delta(\mathcal{A}_{-i}) \to \mathcal{A}_i$ for $i = 1, ..., n$. Each of these satisfies $\varepsilon BR_i^A(x) \in \varepsilon BR_i(x)$.*

Once again it is clearly the case that we can efficiently simulate any εBR_i with $\mathcal{Q}_M$. In addition, our example from Appendix A still holds if we replace $BR_1(x_{-1})$ with $\varepsilon BR_1(x_{-i})$ for $\varepsilon < 2^n$. Hence, for sufficiently small ε, we can also not hope to efficiently simulate any ε-best response query with $\mathcal{Q}_U$. On the other hand, in in Section 3.6.3 we show that for an arbitrary $x \in \Delta(\mathcal{A})$, we can approximate $\mathcal{Q}_M(x)$ up to an additive error of β with a success probability of at least $1 - \delta$ using $\frac{64k^2}{\beta^3}\log(8n/\delta)$ samples from $\mathcal{Q}_U(a)$, where a is sampled according to $x \in \Delta(\mathcal{A})$ and k is the largest strategy set cardinality amongst players. This in turn means that if $\varepsilon^{-1} = poly(k, n)$, then approximating $\mathcal{Q}_M$ up to $\varepsilon/3$ can be done efficiently and with high probability by sampling via $\mathcal{Q}_U$. This allows us to compute subsets of εBR_i for arbitrary m_i. To see why this suffices, suppose that m_i has k_i actions, and that for a given x_{-i}, the quantities $\widehat{U}_0, ..., \widehat{U}_{k_i-1}$ are such that $|\widehat{U}_j - U_i(a_j, x_{-i})|$ for each $a_j \in \mathcal{A}_i$. Then it follows that if $\widehat{U}_r \geq \max_j(\widehat{U}_j) - \frac{\varepsilon}{3}$, then $a_r \in \varepsilon BR_i(x_{-i})$. Furthermore, the set of $a_r \in \mathcal{A}_i$ that satisfy these conditions is non-empty, since at least the index $\text{argmax}_j(\widehat{U}_j)$ satisfies the lower bound.

We end our discussion on ε-best response oracles by noting that in general we cannot simulate εBR_i with either BR_i or $\mathcal{Q}_{PW}$. However, we can simulate some adversarial oracles of the form εBR_i^A by using BR_i or $\mathcal{Q}_{PW}$. This is because BR_i returns best

responses, which are in turn trivially ε-best responses, and we can simulate BR_i with $\mathcal{Q}_{PW}$. However, there may be some actions that are ε-best responses and not strict best responses, and this information is impossible to recover with best response information alone, or even a ranking of actions ascertained from repeated calls to $\mathcal{Q}_{PW}$.

Bounded Rationality Response Queries

A further justification for the paradigm of best response query oracles is that in actual play, if agents are utility maximisers, an algorithm may observe best responses. However, it is often beneficial to relax the assumption that agents are utility maximisers by instead assuming that they behave according to a prespecified notion of bounded rationality. For example, agents may instead respond according to an ε-best response (for a suitable paramater of ε). Another common assumption is that agents take stochastic actions with probability proportional to the utility of the action at hand. One mathematical formalisation of this concept is that of a quantal response (QR) agent.

Definition 2.10 (Quantal Response Agent)**.** *A bounded rationality agent is said to behave according to quantal response with rationality parameter $\lambda \geq 0$ if whenever the agent has to choose between actions $a_1, .., a_k$ with utilities $U_1, ..., U_k$, the agent takes action a_i with probability $\frac{e^{\lambda U_i}}{\sum_{j=1}^{k} e^{\lambda U_j}}$. Note that when $\lambda = 0$ an agent uniformly randomises over all actions and as $\lambda \to \infty$ they uniformly randomise over best actions.*

If we take the perspective of a query oracle being a glimpse into action that may happen in play between agents, we encapsulate the notion of a QR agent via a randomised query oracle: $\mathcal{Q}_{QR}^{\lambda}$.

Definition 2.11 (Quantal Response Query)**.** *Suppose that G is an n player game. For a given $i \in [n]$ and $x_{-i} \in \Delta(\mathcal{A}_{-i})$ we let $\mathcal{Q}_{QR}^{\lambda}(i, x_{-i})$ be a distribution over $\mathcal{A}_i$ dictated by quantal response with parameter λ for the actions player i may take against x_{-i}. I.e. the probability on an action $a_j \in \mathcal{A}_i$ is precisely $\frac{e^{\lambda U_i(a_j, x_{-i})}}{\sum_{r=1}^{n} e^{\lambda U_i(a_r, x_{-i})}}$.*

If one is given access to the exact probabilities that underly the quantal response distribution of an agent, it is possible to reconstruct $\mathcal{Q}_{PW}$, and consequently any BR_i. However, since $\mathcal{Q}_{QR}^{\lambda}$ only gives sample access to the QR distribution, it is in general not possible to discern Q_{PW} let alone BR_i from an efficient number of samples from the QR distribution of m_i, since the underlying QR distribution could have such small differences in probability between actions, that a prohibitive amount of samples are necessary to discern this order with high probability. As an extreme example, we can look at the scenario where $\lambda = 0$. In this case a QR agent simply responds according to a uniformly

random action, and it is impossible to discern any information about best responses or ε-best responses for that matter.

On the other hand, one can use concentration inequalities, in the spirit of sampling approximations to $\mathcal{Q}_U$ in Section 3.6.3, to also show that approximating probabilities in a given QR distribution up to polynomial additive factors can be done efficiently via sampling. Depending on the λ value used in $\mathcal{Q}_{QR}^{\lambda}$, this ultimately makes it possible to compute a subset of ε-best responses, as long as ε is not too small, with an efficient number of samples from $\mathcal{Q}_{QR}^{\lambda}$.

2.3 Literature Review

As mentioned in the introduction, the Nash equilibrium (NE) is a fundamental concept in game theory. They are guaranteed to exist in finite games, as proven in [57], yet computational challenges in finding one abound, most notably, the PPAD-completeness of computing an exact equilibrium even for two-player normal form games as proven in [17] and [15]. For this reason, query complexity has been extensively used as a tool to differentiate hardness of equilibrium concepts in games.

Bimatrix Games

Bimatrix games are a natural starting point to study query complexity of equilibria. For example, it is straightforward to prove that computing an exact Nash equilibrium requires full knowledge of players' payoffs. This is shown in [23] whereby the authors take a generalisation of matching pennies and argue that perturbations in payoffs lead to distinct exact equilibria, hence any algorithm must query all payoffs.

ε-ANE however, can be computed in a query efficient manner for different values of ε. For example, the authors of [18] show that in an arbitrary $m \times n$ bimatrix game, a $\frac{1}{2}$-ANE can be computed with two best response queries, and consequently, two utility queries. The row player picks any pure strategy, $i_1 \in [m]$, the column player computes $j \in [n]$, a best response to i_1, and finally the row player computes $i_2 \in [m]$, a best response to j. If the row player uniformly randomises between i_1 and i_2 and the column player plays j, then the mixed strategy profile is the desired $\frac{1}{2}$-ANE.

The same approach is generalised in [23] where the authors prove that computing a $(1 - \frac{1}{i})$-ANE for any $i \geq 2$ requires at most $2k - i + 1$ utility queries in a $k \times k$ bimatrix game. This turns out to be asymptotically optimal in k for deterministic algorithms, as the authors also prove that computing such equilibria requires at least $k - i + 1$ utility queries in the worst case. The proof of the lower bound relies on the fact that with fewer than $k - i + 1$ queries, some column must remained unqueried, and hence it could either

contain maximal payoff (all 1) or minimal payoff (all 0) for the column player, and this discrepancy disallows computing our desired approximate Nash equilibrium.

The results of [23] are extended and strengthened in [24]. The authors begin by proving an $\Omega(k^2)$ lower bound for deterministic algorithms in computing ε-ANE for $\varepsilon < \frac{1}{2}$ in constant sum bimatrix games (and consequently for bimatrix games in general). The proof relies on a similar but more subtle analysis of the lower bound from [23], where with $o(k^2)$ queries, multiple columns remain largely unqueried and adversarial responses to unqueried strategy profiles can force one such column to need significant mass for the column player in the desired ε-ANE. The fact that an adversary has multiple choices of such "good" columns results in the lower bound. This same approach is also extended to lower bounds against randomised algorithms via the use of Yao's minimax principle, yielding a $\Omega(k^2)$ query complexity for computing $\frac{1}{6k}$-WSNE in $k \times k$ bimatrix games. With this in hand, the utility query complexity of ε-ANE in $k \times k$ bimatrix games is completely characterised for deterministic algorithms: $\varepsilon < \frac{1}{2}$ requires $\Omega(k^2)$ queries, $\frac{1}{2} \leq \varepsilon \leq 1 - \frac{1}{k}$ requires $\Theta(k)$ queries and $\varepsilon > 1 - \frac{1}{k}$ requires no queries (uniformly randomising over all actions for each player suffices).

Randomisation can achieve better query efficiency. This is demonstrated in [24] where the authors present specific randomised algorithms that achieves $(\frac{3-\sqrt{5}}{2} + \alpha)$-ANE for any $\alpha > 0$ using $O(k \log(k))$ utility queries. This breaks the $\frac{1}{2}$ barrier of deterministic algorithms expressed above and is done in 3 steps: providing a query-efficient method of approximating payoffs to mixed strategy profiles of a bimatrix game, using said approximate payoffs to provide a query-efficient algorithm for computing ε-ANE in zero-sum bimatrix games (via an extension of an existing algortihm for the computation of ε-ANE for zero-sum bimatrix games in [29]), and finally, using this aforementioned algorithm as a subroutine in computing the desired ANE for general bimatrix games. This final step is done by adapting the approximate equilibrium computation algorithm of [12].

In terms of well-supported equilibria, an overall similar approach also yields an algorithm for computing $(\frac{2}{3} + \alpha)$-WSNE for any $\alpha > 0$. The main difference lies in converting $\frac{\varepsilon^2}{8}$-ANE to ε-WSNE via the methods of [15] (i.e. shifting mass from suboptimal strategy profiles to best responses and bounding the change in best response utility after said shift). Ultimately, the same query-efficient algorithm for computing ε-ANE in zero-sum games is adapted for ε-WSNE and this is used as a subroutine in an algorithm that adapts that of [46]. As a final note, [24] also contains a linear lower bound for the deterministic query complexity of computing ε-WSNE for any $\varepsilon > 0$. This is in contrast to the fact that computing a $(1 - \frac{1}{k})$-ANE requires no queries at all as a pair of uniformly random strategy profiles achieves such an approximation guarantee.

Multiplayer Games

The authors in [33] use techniques from communication complexity to show that in n-player games, $2^{\Omega(n)}$ queries are needed in the worst case to compute: pure Nash equilibria, pure Nash equilibria in a Bayesian setting, and mixed Nash equilibria. More specifically, the authors assume that each player in the game has access to their own utility function, and that an equilibrium computation protocol consists of multiple rounds where agents communicate individual actions in a pure strategy profile and observe corresponding utilities. This communication model translates well to restrictive *uncoupled* equilibrium computation amongst agents.

For the first lower bound, the authors produce two reductions. In the first they reduce the set disjointness communication problem to that of deciding whether a game has a pure Nash equilibrium. This is done by taking an instance of set disjointness with base set $S = \{0,1\}^n$, and subsets $S_1, S_2 \subseteq S$, and constructing an n-player, binary-action game, G, where each player has action set $\mathcal{A}_i = \{0,1\}$, and hence the space of pure strategy profiles, $\mathcal{A}$, can be identified with S. The judicious construction of the game is such that the utilities of action profiles, $a \in S_1 \cap S_2$, dominate all others for each player. To tie things together, two final players are added to the game who either prefer such a dominating $a \in S_1 \cap S_2$ if it exists, or otherwise engage in a game of matching pennies, thus giving G the property that either $S_1 \cap S_2 \neq \emptyset$, and hence there is a unique pure Nash equilibrium, or otherwise there is a unique mixed Nash equilibrium arising from the matching pennies agents. In the second reduction, the authors reduce set disjointness to computing pure Nash equilibria in games with a finite improvement property (and hence games that forcibly have a pure Nash equilibrium). The reduction uses a similar but more involved construction, but most importantly, it rules out the possibility that restricting algorithms to games that have pure Nash equilibria may lower the communication complexity of computing a Nash equilibrium.

The second lower bound involves computing Nash equilibria in the Bayesian setting, where a common distribution of games is chosen, and algorithms are faced with computing an equilibrium from a game chosen from this distribution. The communication cost an algorithm pays is the expected cost from this distribution of games. Indeed, the difficult family of games from the first lower bound does not imply the existence of difficult distributions in terms of expected communication cost. As a consequence, the authors proceed to explicitly describe a distribution of binary-action, n-player, games whereby for each agent $i \in [n]$, and each opponent strategy profile a_{-i}, a best response from the agent action set, $A = \{0,1\}$ is chosen uniformly randomly (with utility 1, as opposed 0 for the worse response). Under this distribution, the authors prove that the probability a game from the sample has no pure Nash equilibria is bounded away from 0 by a constant

α. Furthermore, using classical communication complexity techniques, they focus on combinatorial rectangles over the set of possible player inputs, where each game in the rectangle has no pure Nash equilibrium. From here, they show that each such rectangle has at most probability mass $2^{-2^{n-1}}$ in their given distribution. These two results prove that the number of such monochromatic combinatorial rectangles must in fact be doubly exponential in n, from which the exponential communication complexity lower bound holds.

As for the third lower bound, the authors rightfully note that high communication complexity in mixed equilibria can be an artifice of arbitrary precision representations of probability distributions over action profiles or of payoffs themselves. In spite of this, they create a family of games that are succinctly representable in a well-defined sense and yet, where it can be shown that computing a mixed Nash equilibrium over them has exponential communication complexity in n (the number of players) as well. The construction rests on generalising Jordan's game from [39] to obtain a family of n-player games parametrised by any function $f : \{0,1\}^{n-2} \to \mathbb{R}$. Each choice of f gives a different unique mixed equilibrium of the modified Jordan game, but the authors obtain the aforementioned succinct representation by focusing on a judicious subset of functions from the parameter space (in turn parameterised by the set of all boolean functions $h : \{0,1\}^{n-2} \to \{0,1\}$). Due to the fact that each such choice of h gives rise to a different equilibrium, lower bounds on communication complexity follow.

As a final note on this paper, the authors also briefly delve into upper bounds on the communication complexity of equilibrium concepts. They trivially give upper bounds that match these lower bounds for binary-action games (agents in the upper bound procedure communicate their entire payoff tensor), but most interestingly, for correlated equilibria, they create a communication-efficient protocol by adapting the algorithm of [60], thus obtaining a polynomial communication complexity.

Continuing with the query complexity of correlated equilibria, the authors of [36] prove two important results: any deterministic algorithm that computes an ε-CE in n-player binary action games requires at least $2^{\Omega(n)}$ utility queries in the worst case, and any randomised algorithm that computes an exact correlated equilibrium in n-player binary action games requires $2^{\Omega(n)}$ utility queries in the worst-case.

To show the first result, they consider $\frac{1}{2}$-CE and reduce the problem of computing such an equilibrium to a combinatorial problem on the n-dimensional boolean hypercube which they call the approximate sink problem (AS) . To be precise, if $v \in \{0,1\}^n$ is a vertex of the hypercube and v^i is said vertex with the i-th bit flipped, the authors give each directed edge of the hypercube, (v, v^i), a sign $R(v, v^i) \in \{-1, 1\}$ such that $R(v, v^i) = -R(v^i, v)$. An algorithm is allowed to query any vertex $v \in \{0,1\}^n$, and consequently learns the value

of all $R(v, v^i)$ for $i = 1, ..., n$, i.e. the weights of edges from v. Finally, the goal of an algorithm which solves AS is to find vertex with in-degree no less than $\frac{n}{4}$, or equivalently, such that $\sum_i R(v, v^{(i)}) \leq \frac{n}{2}$. Given an instance of AS, it is straightforward to construct an n-player binary action game such that $U_i(v^i) - U_i(v) = R(v, v^i)$, with utilities being either 0 or 1, and such that queries to the game's payoff matrix correspond to queries in the AS model. Finally, the game's construction is also such that a $\frac{1}{2}$-correlated equilibrium can be immediately transformed to an approximate sink of the original problem instance. As a consequence, lower bounds in the query complexity of AS carry over to lower bounds in the query complexity of $\frac{1}{2}$-correlated equilibria.

The main idea behind the lower bound on AS is that any algorithm for AS which uses less than $2^{n/8-1}$ queries can be turned into a "polite" one, i.e. one where any query v is made to a vertex for which at least $3n/4$ of its neighbours have not been queried. The details follow from involved combinatorial arguments, but intuitively there is enough room in the n-dimensional hypercube to fix non-politeness of an algorithm solving AS. Ultimately, politeness gives an adversary all the power they need to fool an algorithm solving AS, as whatever answer is returned, $\frac{3n}{4}$ of its edge orientations can be changed to make such an answer incorrect.

For lower bounds against randomised algorithms for computing exact correlated equilibria, the authors consider a weighted variant of the AS problem called the "non-positive vertex" problem (NPV). A similar reduction to before shows that the ability to compute an exact correlated equilibrium gives rise to algorithms solving NPV, hence lower bounds for the latter imply lower bounds for the former. To construct lower bounds for NPV in the setting of randomised algorithms, the authors exhibit a probability distribution of difficult NPV instances from rapidly mixing random walks on the boolean hypercube, which are appended to a fixed hamiltonian cycle of the hypercube. In their construction, solving NPV amounts to finding the end of said paths, but due to the fact that they mix rapidly, a subexponential number of queries cannot succeed with high probability, and thus lower bounds for randomised algorithms follow from Yao's minimax principle.

The author in [3] improves upon the previous two papers by proving exponential in n utility query lower bounds for randomised algorithms computing ε-WSNE when ε is a small enough constant. The crux of the proof is in three steps: reducing the computation of WSNE in a suitable $2n$-player game to that of finding an approximate fixed point of an n-dimensional Lipschitz continuous mapping, reducing the problem of finding such an approximate fixed point to that of finding the end of a simple path on the n-dimensional hypercube, and finally, a proof of this latter problem being hard even via randomised algorithms. The first step in this reduction follows a proof of Brower's fixed point theorem from Nash's theorem in [68], and the second is an application of a

reduction described in [37]. As for the third step in the proof, the problem is related to finding the end points of a random path on the n-dimensional hypercube, which given the fact that random walks on the hypercube mix rapidly, is difficult to do. Random walks have self-intersection, so care must be taken in making the reduction, since cycles can occur. However, the authors shows that excising said cycles still maintains a path that is long enough for hardness to creep in.

The authors of [29] also focus on the query complexity of computing correlated equilibria and relevant approximations in the utility query model for n-player games. For ε-correlated equilibria, they prove a logarithmic upper bound in n by adapting well-known multiplicative weights algorithms which compute approximate correlated equilibria to the utility query setting. This is done by a novel query-efficient method of approximating mixed strategy utilities via the utility oracle, whereby mixed strategies are perturbed enough to ensure that all actions have non-trivial support, whence concentration inequalities can be used to ensure that empirical averages are close to true expected utilities with high probability. This upper bound is shown to be asymptotically optimal, as the authors demonstrate via an adversarial distribution of binary action games and Yao's minimax principle that the number of queries needed to find a $(\frac{1}{2} - \frac{1}{k})$-correlated equilibrium with high probability is at least $\log_{k(k-1)}(n)$.

Interestingly, the authors also present a query-efficient algorithm for computing ε-WSNE in succintly representable games. This is done by a "halving" algorithm, whereby a family of candidate games is maintained, and at any given iteration, a game G' is constructed with payoffs at each strategy profile arising from median payoffs of candidate games. An $\frac{\varepsilon}{2}$-WSNE, x, is computed for G', and a large enough sample is taken from x to ensure that if x is not an ε-WSNE of the intended game G, then with high probability the sample will be inconsistent with a constant fraction of candidate games. It is important to note that this algorithm is not computationally efficient.

As a brief aside on the related topic of communication complexity of equilibrium computation, the authors in [5] have also proved lower bounds in communication complexity for computing ε-WSNE for small enough ε in both bimatrix and multiplayer games. As the authors mention in their paper, this result is obtained in four steps: proving lower bounds in terms of the query complexity of end of line instances of specific graphs (of constant degree), lifting these query complexity lower bounds to communication complexity lower bounds, embedding the specific line as a Lipschitz function, and finally constructing a game that imitates said Lipschitz function.

Constrained Classes of Games

Given the aforementioned lower bounds in query complexity of equilibrium computation, it is natural to focus on constrained families of games with hopes of obtaining algorithms that perform better than such worst-case lower bounds. In [31], the authors focus on approximate equilibrium computation in anonymous games with n players, each with k strategies (where the only relevant information for a player's payoff is a partition of opposing players into the strategies they play). In this setting they begin by proving that computing an exact Nash equilibrium requires querying all payoffs exhaustively as well as exhibiting a two-player, three-strategy, anonymous game where the unique equilibrium is irrational, thus further motivating the study of approximate equilibria in the anonymous setting. The main result of the paper is in the context of computing ε-ANE of n-player anonymous games with binary action sets. The authors give a randomised algorithm that computes a $O(n^{-1/4})$-ANE with $\widetilde{O}(n^{3/2})$ payoff queries and a $\widetilde{O}(n^{3/2})$ run time. In addition, they prove a lower bound of $\Omega(k \log(k))$ utility queries needed to compute an ε-WSNE by randomised algorithms.

Returning to [23], the authors also present query efficient algorithms for graphical games of constant degree and various forms of congestion games. For the former, an exact Nash equilibrium can be found with a polynomial number of payoff queries by discovering every payoff in the game, which given the structure of the game, can be done without querying every pure strategy profile. For congestion games with m parallel links and n players, at least $\log(n) + m$ utility queries are needed to compute an exact Nash equilibrium. The $\log(n)$ term arises from considering $m = 2$ links and noticing that in the worst case equilibrium computation involves performing binary search over which agents take which link. As for the m term in the lower bound, this arises from simply considering an m-link game with 1 player and noticing that all utilities of links need to be known to compute an exact Nash equilibrium. On the other hand, the authors also provide an upper bound of $O\left(\log(n) \cdot \frac{\log^2(m)}{\log\log(m)}\right)$ utility queries in computing exact Nash equilibria in this family of congestion games. They provide an algorithm which works by iteratively grouping players into blocks, where all players in block must play on same link. In each round of the algorithm, an invariant is maintained that blocks are in equilibrium (i.e. no block of players can collectively deviate in order to reduce their latency). Initially players are placed in a single block, and in each round, blocks are split into smaller blocks and a new equilibrium is computed for smaller blocks. Eventually block sizes are reduced to 1 and the result is a Nash equilibrium.

Fictitious Play and Best Response Queries

Best response queries are a weaker but natural query model which is powerful enough to implement fictitious play (FP), a dynamic first proposed in [13], and proven to converge in [63] for two-player zero-sum games to an approximate NE. In FP, the row player and column player each maintain a multiset of pure strategy profiles, S_r and S_c respectively. Furthermore, at each iteration of FP, the row-player computes a best-response to the empirical distribution of S_c (i.e. that of sampling uniformly randomly from S_c), and adds this strategy to S_r. The column player does the same, and this process continues, whereby the empirical distributions of each player eventually converge to an ε-ANE.

The convergence of FP has also been extended beyond two-player zero-sum games to also include games solvable by iterated elimination of strictly dominated strategy sets in [55], potential games in [54] and [53], as well as $2 \times n$ games with generic payoffs in [10] . On the other hand, [67] contains a simple non-zero-zum variant of rock-paper-scissors along with initial conditions that guarantee that FP cycles indefinitely around the unique equilibrium and thus does not converge. The authors of [19] provide an explicit construction to show that if ties amongst best responses are broken arbitrarily, then the rate of convergence of FP is quite slow in the worst case (with $\left(\Omega(t^{-1/n})\right)$-ANE for $n \times n$ games after t rounds). In addition, beyond issues of non-convergence, FP can have a poor approximation value for general games, as shown in [30].

Quantal Response

Quantal response was initially introduced in [52] as a statistical approximation to rational play. The authors introduce the notion of a quantal response equilibrium as a solution concept in games, whereby agents do not deterministcally employ best responses in play, but rather act stochastically with probability proportional to the expected utility of their actions. Subsequent work in behavioural economics such as [32] and [69] has further backed the plausability of quantal response as a model for human behaviour in games. In particular, there has been much work in the multi-agent systems community around using quantal response as a model of human behaviour as in [71], especially in the context of Stackelberg security games. Some notable examples include: efficient algorithms for computing optimal defender strategies against QR adversaries in the unconstrained defender setting and efficient approximation of optimal defender strategies against QR adversaries in the constrained defender setting in [72]; Extensions to QR adversaries whereby an exponentially weighted average of subjective utilities is used to model defender behaviour in [59]; and an in-depth analysis of the relative merits of stochastic bounded rationality models of attacker play in repeated Stackelberg security games in [42].

Chapter 3

Computing ε-ANE in Large Games with Utility Queries

In this chapter we investigate the problem of equilibrium computation for "large" n-player games. Large games have a Lipschitz-type property that no single player's utility is greatly affected by any other individual player's actions. In this chapter, we mostly focus on the case where any change of strategy by a player causes other players' payoffs to change by at most $\frac{1}{n}$. We study algorithms having query access to the game's utility function, aiming to find ε-approximate Nash equilibria. We seek algorithms that obtain ε as small as possible, with time polynomial in n.

Our main result is a randomised algorithm that achieves ε approaching $\frac{1}{8}$ for binary-action games in a *completely uncoupled* setting, where each player observes their own payoff to a query, and adjusts their behaviour independently of other players' payoffs/actions. $O(\log n)$ rounds/queries are required. We also show how to obtain a slight improvement over $\frac{1}{8}$, by introducing a small amount of communication between the players. Finally, we give extension of our results to large games with more than two strategies per player, and alternative largeness parameters.

3.1 Introduction

In studying the computation of solutions of multi-player games, we encounter the well-known problem that a game's payoff function has description length exponential in the number of players. One approach is to assume that the game comes from a concisely-represented class (for example, graphical games, anonymous games, or congestion games), and another one is to consider algorithms that have query access to the game's payoff function.

In this chapter, we study the computation of approximate Nash equilibria of multi-player games having the feature that if a player changes their behaviour, it only has a small

effect on the payoffs that result to any other player. These games, sometimes called *large* games, or *Lipschitz* games, have recently been studied in the literature, since they model various real-world economic interactions; for example, an individual's choice of what items to buy may have a small effect on prices, where other individuals are not strongly affected. Note that these games do not have concisely-represented payoff functions, which makes them a natural class of games to consider from the query-complexity perspective. It is already known how to compute approximate *correlated equilibria* for unrestricted n-player games. Here we study the more demanding solution concept of approximate Nash equilibrium.

Large games (equivalently, small-influence games) are studied in [1] and [41]. In these papers, the existence of pure ε-approximate Nash equilibria for $\varepsilon = \gamma\sqrt{8n\log(2kn)}$ is established, where γ is the largeness/Lipschitz parameter of the game, and k is the number of pure strategies for each player. In particular, since we assume that $\gamma = \frac{1}{n}$ and $k = 2$ we notice that $\varepsilon = O(\sqrt{\log(n)/n})$ so that there exist arbitrarily accurate pure Nash equilibria in large games as the number of players increases. The authors of [43] study this class of games from the mechanism design perspective of mediators who aim to achieve a good outcome to such a game via recommending actions to players. The author of [2] studies large binary-action in *anonymous* games where anonymity is exploited to create a randomised dynamic on pure strategy profiles that with high probability converges to a pure approximate equilibrium in $O(n\log n)$ steps.

Our main result applies in the setting of *completely uncoupled dynamics* in equilibria computation. These dynamics have been studied extensively: the authors of [34] show that there exist finite-memory uncoupled strategies that lead to pure Nash equilibria in every game where they exist. Also, there exist finite memory uncoupled strategies that lead to ε-ANE in every game. Young's interactive trial and error from [75] outlines completely uncoupled strategies that lead to pure Nash equilibria with high probability when they exist. Regret testing from [61] and its n-player extension in [25] show that there exist completely uncoupled strategies that lead to an ε-Nash equilibrium with high probability. Randomisation is essential in all of these approaches, as the authors of [35] show that it is impossible to achieve convergence to Nash equilibria for all games if one is restricted to deterministic uncoupled strategies. This prior work is not concerned with rate of convergence; by contrast here we obtain efficient bounds on runtime. Convergence in adaptive dynamics for exact Nash equilibria is also studied in [33] where the authors provide provide exponential lower bounds via communication complexity results. The author of [3] also proves an exponential lower bound on the rate of convergence of adaptive dynamics to an approximate Nash equilibrium for general binary games. Specifically, he proves that there is no k-queries dynamic that converges to an ε-WSNE in $\frac{2^{\Omega(n)}}{k}$ steps with

probability of at least $2^{-\Omega(n)}$ in all n-player binary games. Both of these results motivate the study of specific subclasses of these games, such as the "large" games studied here.

3.2 Preliminaries

We recall some notation from Section 2.1. We consider games with n players where each player, $i \in [n]$, has k actions, $\mathcal{A}_i = \{0, 1, ..., k-1\}$. Given the common action set, we say that $\mathcal{A}_i = \mathcal{A}^*$ for all $i \in [n]$ and we recall that $\mathcal{A} = \prod_{i=1}^{n} \mathcal{A}_i = (\mathcal{A}^*)^n$. Let $a = (a_i, a_{-i})$ denote an *action profile* in which player i plays action a_i and the remaining players play action profile a_{-i}. We also consider *mixed strategies*, which are defined by the probability distributions over the action set $\mathcal{A}^*$. We write $x = (x_i, x_{-i})$ to denote a *mixed-strategy profile* where x_i is a distribution over $\mathcal{A}^*$ corresponding to the i-th player's mixed strategy. To be more precise, x_i is a vector $(x_{ij})_{j=1}^{k-1}$ such that $\sum_{j=1}^{k-1} x_{ij} \leq 1$ where x_{ij} denotes the i-th player's probability mass on his j-th strategy. Furthermore, we denote $x_{i0} = 1 - \sum_{j=1}^{k-1} x_{ij}$ to be the implicit probability mass the i-th player places on his 0-th pure strategy. Once again we recall that $\Delta(\mathcal{A}^*)$ denotes the space of all mixed strategies over the action set $\mathcal{A}^*$. Each player i has a payoff function $U_i \colon \mathcal{A} \to [0, 1]$ mapping an action profile to some value in $[0, 1]$. We will at times write $U_i(x) = \mathbb{E}_{a \sim x}[U_i(a)]$ to denote the expected payoff of player i under mixed strategy x. An action a is player i's *best response* to mixed strategy profile x if $a \in \operatorname{argmax}_{j \in \mathcal{A}^*} U_i(j, x_{-i})$.

We assume our algorithms and players have no other prior knowledge of the game but can access payoff information through querying the *utility oracle* $\mathcal{Q}_U$. For each *utility query* specified by an action profile $a \in \mathcal{A}$, the query oracle will return $(U_i(a))_{i=1}^{n}$, the n-dimensional vector of payoffs to each player. Our goal is to compute an *approximate Nash equilibrium* with a small number of queries. In the completely uncoupled setting, a query works as follows: each player i chooses his own action a_i independently of the other players, and learns his own payoff $U_i(a)$ but no other payoffs or actions.

In the rest of this chapter it is useful to quantify the degree of suboptimality of a specific player in a mixed strategy profile. We call this quantity the regret of a player and define it as follows:

Definition 3.1 (Regret)**.** *Let x be a mixed strategy profile, the* regret *for player i at x is*

$$reg(x, i) = \max_{j \in \mathcal{A}^*} \mathop{\mathbb{E}}_{a_{-i} \sim x_{-i}} [U_i(j, a_{-i})] - \mathop{\mathbb{E}}_{a \sim x} [U_i(a)] .$$

Note that a mixed strategy profile x is an *ε-approximate Nash equilibrium* (ε-ANE) if for each player i, the regret satisfies $reg(x, i) \leq \varepsilon$. In Section 3.6.1 we will address the stronger notion of a *well-supported* approximate Nash equilibrium as per Definition 2.3.

We recall that such an equilibrium is a mixed-strategy profile where players only place positive probability on actions that are approximately optimal.

Observation 2. *To find an exact Nash (or even, correlated) equilibrium of a large game, in the worst case it is necessary to query the game's utilities exhaustively, even with randomised algorithms. This can be done shown using a similar negative result for general games due to [36], and noting that we can obtain a strategically equivalent γ-large game (Def. 3.2), by scaling down the payoffs into the interval $[0, \gamma]$.*

We will assume the following *largeness* condition in our games. Informally, such largeness condition implies that no single player has a large influence on any other player's utility function.

Definition 3.2 (Large Games)**.** *A game is* γ-large *if for any two distinct players $i \neq j$, any two distinct actions a_j and a'_j for player j, and any tuple of actions a_{-j} for everyone else:*

$$|U_i(a_j, a_{-j}) - U_i(a'_j, a_{-j})| \leq \gamma \in [0, 1].$$

We will call γ the *largeness parameter* of the game; in [1] this quantity is called the Lipschitz value of the game. One immediate implication of the largeness assumption is the following Lipschitz property of the utility functions.

Lemma 3.1. *For any player $i \in [n]$, and any action $j \in \mathcal{A}^*$, the fixed utility function $U_i(j, x_{-i}) : \Delta(\mathcal{A}^*)^{(n-1)} \to [0, 1]$ is a γ-Lipschitz function of the second argument $x_{-i} \in \Delta(\mathcal{A}^*)^{n-1} \subseteq \mathbb{R}^{(n-1)\times(k-1)}$ w.r.t. the ℓ_1 norm.*

Proof. Without loss of generality consider $i = 1$ and $j = 0$. Let $q = x_{-1}$ and $q' = x'_{-1}$ be two mixed strategy profiles for the other players. For $i \geq 2$ and $j \in \mathcal{A}^* \setminus \{0\}$, let $\delta_{ij} = q'_{ij} - q_{ij}$. Note that $\|q - q'\|_1 = \sum_{ij} |\delta_{ij}|$.

Let e_{ij} be the unit vector that has a 1 in the (ij)-th entry and 0 elsewhere. We first show that there exists an ordering of the discrete set $\{(ij) \mid 2 \leq i \leq n,\ 1 \leq j \leq k\}$ denoted by $\{\alpha_1, \alpha_2, ..., \alpha_{(n-1)(k-1)}\}$ such that for all $\ell = 1, ..., (n-1)(k-1)$, the vector $q_\ell = q + \sum_{i=1}^{\ell} \delta_{\alpha_i} e_{\alpha_i}$ represents valid mixed strategy profiles for players $i \geq 2$.

Suppose that we fix i, and consider q_i and q'_i as the mixed strategies of player i arising in q and q'. We recall that these are vectors in $[0, 1]^{k-1}$ whose components sum is less than 1. We consider two cases. In the first, suppose that there exists a j such that $\delta_{ij} < 0$ by definition, $\delta_{ij} < q_{ij}$, hence $q_i + \delta_{ij} e_j$ is a valid mixed strategy for player i.

In the second, suppose that $\delta_{ij} > 0$ for all j. Now suppose that $\delta_{ij} > q_{i0} = 1 - \sum_{j=1}^{k-1} q_{ij}$ for all j. If this is the case then q'_i cannot possibly be a valid mixed strategy for player i,

hence it must be the case that for some j, $\delta_{ij} < q_{i0}$, hence once again $q_i + \delta_{ij} e_j$ is a valid mixed strategy for player i.

Since such a choice of valid updates by δ_{ij} can always be found for valid q_i and q_i', we can recursively find valid shifts by δ_{ij} in a specific coordinate to reach q_i' from q_i. If this is applied in order for all players $i \geq 2$, the aforementioned claim holds and indeed $q_\ell = q + \sum_{i=1}^{\ell} \delta_{\alpha_i} e_{\alpha_i}$ for some ordering $\{\alpha_1, ..., \alpha_{(n-1)(k-1)}\}$.

With this in hand, we can use telescoping sums and the largeness condition to prove our lemma. For simplicity of notation, in what follows we assume that $q_0 = q$, and we recall that by definition $q_{(n-1)(k-1)} = q'$.

$$
\begin{aligned}
|U_i(j, q') - U_i(j, q)| &= \left| \sum_{\ell=1}^{(n-1)(k-1)} U_i(j, q_\ell) - U_i(j, q_{\ell-1}) \right|, \\
\text{(Triangle Inequality)} \qquad &\leq \sum_{\ell=1}^{(n-1)(k-1)} |U_i(j, q_\ell) - U_i(j, q_{\ell-1})|, \\
\text{(Definition of Largeness)} \qquad &\leq \sum_{\ell=1}^{(n-1)(k-1)} \gamma |\delta_{\alpha_\ell}| = \gamma \|q' - q\|_1.
\end{aligned}
$$

which proves our claim. □

From now on until Section 3.6 we will focus on $\frac{1}{n}$-large binary action games where $\mathcal{A}^* = \{0, 1\}$ and $\gamma = \frac{1}{n}$. The reason for this is that the techniques we introduce can be more conveniently conveyed in the special case of $\gamma = \frac{1}{n}$, and subsequently extended to general γ.

Recall that x_i denotes a mixed strategy of player i. In the special case of binary-action games, we slightly abuse the notation to let x_i denote the probability that player i plays 1 (as opposed to 0), since in the binary-action case, this single probability describes player i's mixed strategy.

The following notion of *discrepancy* will be useful.

Definition 3.3 (Discrepancy)**.** *Letting x be a mixed strategy profile, the* discrepancy *for player i at x is*

$$
disc(x, i) = \left| \mathop{\mathbb{E}}_{a_{-i} \sim x_{-i}} [U_i(0, a_{-i})] - \mathop{\mathbb{E}}_{a_{-i} \sim x_{-i}} [U_i(1, a_{-i})] \right|.
$$

Estimating payoffs for mixed profiles We can approximate the expected payoffs for any mixed strategy profile by repeated calls to the oracle $\mathcal{Q}_U$. In particular, for any target accuracy parameter β and confidence parameter δ, consider the following procedure to implement an oracle $\mathcal{Q}_{\beta,\delta}$:

- For any input mixed strategy profile x, compute a new mixed strategy profile $x' = (1 - \frac{\beta}{2})x + (\frac{\beta}{2})\mathbf{1}$ such that each player i is playing uniform distribution with probability $\frac{\beta}{2}$ and playing distribution x_i with probability $1 - \frac{\beta}{2}$.
- Let $N = \frac{64}{\beta^3} \log(8n/\delta)$, and sample N payoff queries randomly from x', and call the oracle $\mathcal{Q}_U$ with each query as input to obtain a payoff vector.
- Let $\widehat{U}_{i,j}$ be the average sampled payoff to player i for playing action j.[1] Output the payoff vector $(\widehat{U}_{ij})_{i\in[n],j\in\{0,1\}}$.

Lemma 3.2. *For any $\beta, \delta \in (0,1)$ and any mixed strategy profile x, the oracle $\mathcal{Q}_{\beta,\delta}$ with probability at least $1-\delta$ outputs a payoff vector $(\widehat{U}_{i,j})_{i\in[n],j\in\{0,1\}}$ that has an additive error of at most β, that is for each player i, and each action $j \in \{0,1\}$,*

$$|U_i(j, x_{-i}) - \widehat{U}_{i,j}| \leq \beta.$$

The lemma follows from Proposition 1 of [29] and the largeness property.

Extension to Stochastic Utilities. We consider a generalisation where the utility to player i of any pure profile a may consist of a probability distribution $D_{a,i}$ over $[0,1]$, and if a is played, i receives a sample from $D_{a,i}$. The player wants to maximise his expected utility with respect to sampling from a (possibly mixed) profile, together with sampling from any $D_{a,i}$ that results from a being chosen. If we extend the definition of $\mathcal{Q}$ to output samples of the $D_{a,i}$ for any queried profile a, then $\mathcal{Q}_{\beta,\delta}$ can be defined in a similar way as before, and simulated as above using samples from $\mathcal{Q}$. Our algorithmic results extend to this setting.

3.3 Warm-up: 0.25-Approximate Equilibrium

In this section, we exhibit some simple procedures whose general approach is to query a constant number of mixed strategies (for which additive approximations to the payoffs can be obtained by sampling). Observation 3 notes that a $\frac{1}{2}$-approximate Nash equilibrium can be found without using any payoff queries:

Observation 3. *Consider the following "uniform" mixed strategy profile. Each player puts $\frac{1}{2}$ probability mass on each action: for all i, $x_i = \frac{1}{2}$. Such a mixed strategy profile is a $\frac{1}{2}$-approximate Nash equilibrium.*

[1] If the player i never plays an action j in any query, set $\widehat{U}_{i,j} = 0$.

We present two algorithms that build on Observation 3 to obtain better approximations than $\frac{1}{2}$. For simplicity of presentation, we assume that we have access to a mixed strategy query oracle $\mathcal{Q}_M$ that returns exact expected payoff values for any input mixed strategy p. Our results continue to hold if we replace $\mathcal{Q}_M$ by $\mathcal{Q}_{\beta,\delta}$. [2]

Obtaining $\varepsilon = 0.272$. First, we show that having each player making small adjustment from the "uniform" strategy can improve ε from $\frac{1}{2}$ to around 0.27. We simply let players with large regret shift more probability weight towards their best responses. More formally, consider the following algorithm **OneStep** with two parameters $\alpha, \Delta \in [0, 1]$:

- Let the players play the "uniform" mixed strategy. Call the oracle $\mathcal{Q}_M$ to obtain the payoff values of $U_i(0, x_{-i})$ and $U_i(1, x_{-i})$ for each player i.
- For each player i, if $U_i(0, x_{-i}) - U_i(1, x_{-i}) > \alpha$, then set $x_i = \frac{1}{2} - \Delta$; if $U_i(1, x_{-i}) - U_i(0, x_{-i}) > \alpha$, set $x_i = \frac{1}{2} + \Delta$; otherwise keep playing $x_i = \frac{1}{2}$.

Theorem 3.1. *If we use algorithm* **OneStep** *with parameters* $\alpha = 2 - \sqrt{\frac{11}{3}}$ *and* $\Delta = \sqrt{\frac{11}{48}} - \frac{1}{4}$*, then the resulting mixed strategy profile is an ε-approximate Nash equilibrium with* $\varepsilon \leq 0.272$.

Proof. Let x denote the "uniform" mixed strategy, and x' denote the output strategy by **OneStep**. We know that $\|x - x'\|_1 \leq n\Delta$. By Lemma 3.1, we know that for any player i and action j, $|U_i(j, x_{-i}) - U_i(j, x'_{-j})| \leq \Delta$.

Consider a player i whose discrepancy in x satisfies $disc(x, i) \leq \alpha$. Then such player's discrepancy in x' is at most $disc(x', i) \leq \alpha + 2\Delta$, so his regret in x' is bounded by

$$reg(x', i) = x'_i \, disc(x', i) = disc(x', i)/2 \leq \alpha/2 + \Delta. \tag{3.1}$$

Consider a player i such that $disc(x, i) > \alpha$. Then we consider two different cases. In the first case, the best response of player i remains the same in both profiles x and x'. Since $disc(x', i) \leq 1$, we can bound the regret by

$$reg(x', i) = x'_i \, disc(x', i) = \left(\frac{1}{2} - \Delta\right). \tag{3.2}$$

In the second case, the best response of player i changes when the profile x changes to x'. In this case, the discrepancy is at most $2\Delta - \alpha$, and so the regret is bounded by

$$reg(x', i) = x'_i \, disc(x', i) = \left(\frac{1}{2} + \Delta\right)(2\Delta - \alpha). \tag{3.3}$$

[2] In particular, if we use $\mathcal{Q}_{\beta,\delta}$ for our query access, then with probability at least $1 - \delta$ we will get $(\varepsilon + O(\beta))$-approximate equilibrium, where ε is the approximation performance obtainable via access to $\mathcal{Q}_M$.

By combining all cases from Equations (3.1) to (3.3), we know the regret is upper-bounded by

$$reg(x', i) \leq \max\left(\frac{\alpha}{2} + \Delta, \frac{1}{2} - \Delta, \frac{1}{2}(1 + 2\Delta)(2\Delta - \alpha)\right). \tag{3.4}$$

By choosing values

$$(\alpha^*, \Delta^*) = \left(2 - \sqrt{\frac{11}{3}}, \sqrt{\frac{11}{48}} - \frac{1}{4}\right) \approx (0.085, 0.229),$$

the right hand side of Equation (3.4) is bounded by 0.272. Thus if we use the optimal α^* and Δ^* in our algorithm, we can attain an $\varepsilon = 0.272$ approximate Nash equilibrium. □

Obtaining $\varepsilon = 0.25$. We now give a slightly more sophisticated algorithm than the previous one. We will again have the players starting with the "uniform" mixed strategy, then let players shift more weights toward their best responses, and finally let some of the players switch back to the uniform strategy if their best responses change in the adjustment. Formally, the algorithm **TwoStep** proceeds as:

- Start with the "uniform" mixed strategy profile, and query the oracle $\mathcal{Q}_M$ for the payoff values. Let b_i be player i's best response.
- For each player i, set the probability of playing their best response b_i to be $\frac{3}{4}$. Call $\mathcal{Q}_M$ to obtain payoff values for this mixed strategy profile, and let b_i' be each player i's best response in the new profile.
- For each player i, if $b_i \neq b_i'$, then resume playing $x_i = \frac{1}{2}$. Otherwise maintain the same mixed strategy from the previous step.

Theorem 3.2. *The mixed strategy profile output by* **TwoStep** *is an ε-approximate Nash equilibrium with $\varepsilon \leq 0.25$.*

Proof. Let x denote the "uniform" strategy profile, x' denote the strategy profile after the first adjustment, and x'' denote the output strategy profile by **TwoStep**.

For any player i, there are three cases regarding the discrepancy $disc(x, i)$.

1. The discrepancy $disc(x, i) > \frac{1}{2}$;
2. The discrepancy $disc(x, i) \leq \frac{1}{2}$ and player i returns to the uniform mixed strategy at the end;
3. The discrepancy $disc(x, i) \leq \frac{1}{2}$ and player i does not return to the uniform mixed strategy in the end.

Before we go through all the cases, the following facts are useful. Observe that $\|x - x'\|, \|x - x''\|, \|x' - x''\| \leq n/4$, so for any action j,

$$\max\{|U_i(j, x'_{-i}) - U_i(j, x''_{-i})|, |U_i(j, x_{-i}) - U_i(j, x'_{-i}), |U_i(j, x_{-i}) - U_i(j, x''_{-i})|\} \leq \frac{1}{4}. \quad (3.5)$$

It follows that

$$\max\{|disc(x', i) - disc(x'', i)|, |disc(x, i) - disc(x', i)|, |disc(x, i) - disc(x'', i)|\} \leq \frac{1}{2}.$$

We will now bound the regret of player i in the first case. Since in the mixed strategy profile x, the best response of player i is better than the other action by more than $\frac{1}{2}$. This means the best response action will remain the same in x' and x'' for this player, and they will play this action with probability $\frac{3}{4}$ in the end, so his regret is bounded by $\frac{1}{4}$.

Let us now focus on the second case where discrepancy $disc(x, i) \leq \frac{1}{2}$ and player i returns to the uniform strategy of part 1. It is sufficient to show that the discrepancy at the end satisfies $disc(x'', i) \leq \frac{1}{2}$. Without loss generality, assume that the player best response in the "uniform" strategy profile is action $b_i = 1$, and the best response after the first adjustment is action $b_i = 0$. This means

$$U_i(1, x_{-i}) - U_i(0, x_{-i}) \geq 0 \quad \text{and,} \quad U_i(0, x'_{-i}) - U_i(1, x'_{-i}) \geq 0.$$

By combining with Equation (3.5), we have

$$U_i(1, x''_{-i}) - U_i(0, x''_{-i}) \leq U_i(1, x'_{-i}) - U_i(0, x'_{-i}) + \frac{1}{2} \leq \frac{1}{2},$$
$$U_i(0, x''_{-i}) - U_i(1, x''_{-i}) \leq U_i(0, x_{-i}) - U_i(1, x_{-i}) + \frac{1}{2} \leq \frac{1}{2}.$$

Therefore, we know $disc(x'', i) \leq \frac{1}{2}$, and hence the regret $reg(x'', i) \leq \frac{1}{4}$.

Finally, we consider the third case where $disc(x, i) \leq \frac{1}{2}$ and player i does not return to a uniform strategy. Without loss generality, assume that action 1 is best response for player i in both x and x', and so $U_i(1, x'_{-i}) \geq U_i(0, x'_{-i})$. By Equation (3.5), we also have

$$U_i(0, x''_{-i}) - U_i(1, x''_{-i}) \leq \frac{1}{2}.$$

If their best response changes to 0, then regret is bounded by $reg(x'', i) \leq \frac{1}{8}$. Otherwise, if 1 remains as the best response, then regret is again bounded by $reg(x'', i) \leq \frac{1}{4}$. Hence, in all of the cases above we could bound the player's regret by $\frac{1}{4}$. □

3.4 $\frac{1}{8}$-Approximate Equilibrium via Uncoupled Dynamics

In this section, we present our main algorithm that achieves approximate equilibria with $\varepsilon \approx \frac{1}{8}$ in a completely uncoupled setting. In order to arrive at this we first model game dynamics as an uncoupled continuous-time dynamical system where a player's strategy profile updates depend only on their own mixed strategy and payoffs. Afterwards we present a discrete-time approximation to these continuous dynamics to arrive at a query-based algorithm for computing $(\frac{1}{8} + \alpha)$-Nash equilibrium with query complexity logarithmic in the number of players. Here, $\alpha > 0$ is a parameter that can be chosen, and the number of mixed-strategy profiles that need to be tested is inversely proportional to α. Finally, as mentioned in Section 3.2, we recall that these algorithms carry over to games with stochastic utilities, and in this latter setting we can show that our algorithm uses an essentially optimal number of queries.

Throughout the section, we will rely on the following notion of a *strategy/payoff state*, capturing the information available to a player at any moment of time.

Definition 3.4 (Strategy-payoff state)**.** *For any player i, the* strategy/payoff state *for player i is defined as the ordered triple $s_i = (v_{i1}, v_{i0}, x_i) \in [0,1]^3$, where v_{i1} and v_{i0} are the player's utilities for playing pure actions 1 and 0 respectively, and x_i denotes the player's probability of playing action 1. Furthermore, we denote the player's discrepancy by $D_i = |v_{i1} - v_{i0}|$ and we let x_i^* denote the probability mass on the best response, that is if $v_{i1} \geq v_{i0}$, $x_i^* = x_i$, otherwise $x_i^* = 1 - x_i$.*

3.4.1 Continuous-Time Dynamics

First, we will model game dynamics in continuous time, and assume that a player's strategy/payoff state (and thus all variables it contains) is a differentiable time-valued function. When we specify these values at a specific time t, we will write $s_i(t) = (v_{i1}(t), v_{i0}(t), x_i(t))$. Furthermore, for any time-differentiable function g, we denote its time derivative by $\dot{g} = \frac{d}{dt}g$. We will consider continuous game dynamics formally defined as follows.

Definition 3.5 (Continuous game dynamic)**.** *A continuous game dynamic consists of an update function f that specifies a player's strategy update at time t. Furthermore, f depends only on $s_i(t)$ and $\dot{s}_i(t)$. In other words, $\dot{x}_i(t) = f(s_i(t), \dot{s}_i(t))$ for all t.*

Observation 4. *We note that in this framework, a specific player's updates do not depend on other players' strategy/payoff states nor their history of play. This will eventually lead us to uncoupled Nash equilibria computation in Section 3.4.2.*

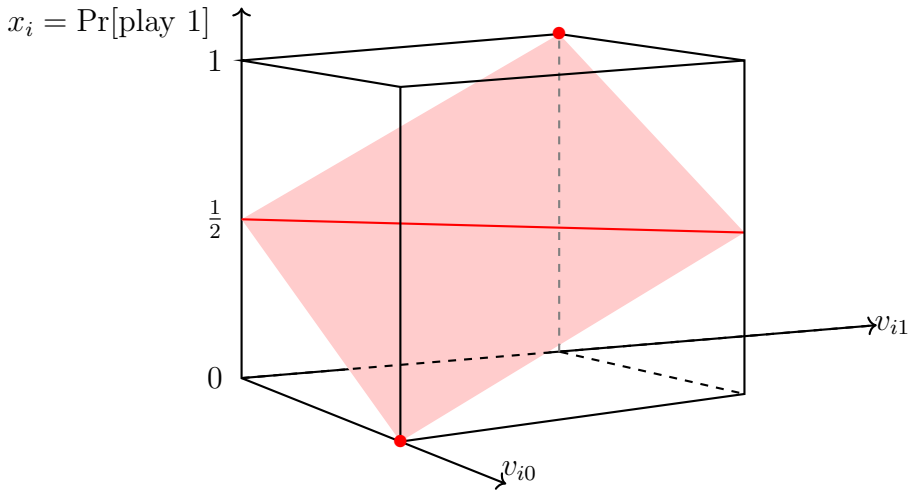

Figure 3.1: Visualisation of $\mathcal{P}$; on the red line, $v_{i0} = v_{i1}$ so the player is indifferent and mixes with equal probabilities; at the red points the player has payoffs of 0 and 1, and makes a pure best response.

A central object of interest in our continuous dynamic is a linear sub-space $\mathcal{P} \subset [0,1]^3$ such that all strategy/payoff states in it incur a bounded regret. Formally, we will define $\mathcal{P}$ via its normal vector $\vec{n} = (-\frac{1}{2}, \frac{1}{2}, 1)$ so that $\mathcal{P} = \{s_i |\ s_i \cdot \vec{n} = \frac{1}{2}\}$. Equivalently, we could also write $\mathcal{P} = \{s_i \mid x_i^* = \frac{1}{2}(1 + D_i)\}$. (See Figure 3.1 for a visualisation.) With this observation, it is straightforward to see that any player with strategy/payoff state in $\mathcal{P}$ has regret at most $\frac{1}{8}$.

Lemma 3.3. *Suppose that a player $i \in [n]$ has a strategy/payoff state, $s_i \in \mathcal{P}$, then their regret is at most $\frac{1}{8}$.*

Proof. This follows from the fact that a player's regret can be expressed as $D_i(1 - x_i^*)$ and the fact that all points on $\mathcal{P}$ also satisfy $x_i^* = \frac{1}{2}(1 + D_i)$. In particular, the maximal regret of $\frac{1}{8}$ is achieved when $D_i = \frac{1}{2}$ and $x_i^* = \frac{3}{4}$. □

Next, we want to show there exists a dynamic that allows all players to eventually reach $\mathcal{P}$ and remain on it over time. We notice that for a specific player, $\dot{v}_{i1}$, $\dot{v}_{i0}$ and subsequently $\dot{D}_i$ measure the cumulative effect of other players shifting their strategies. However, if we limit how much any individual player can change their mixed strategy over time by imposing $|\dot{x}_i| \leq 1$ for all i, Lemma 3.1 guarantees $|\dot{v}_{ij}| \leq 1$ for $j = 0, 1$ and consequently $|\dot{D}_i| \leq 2$. With these quantities bounded, we can consider an adversarial framework where we construct game dynamics by solely assuming that $|\dot{x}_i(t)| \leq 1$, $|\dot{v}_{ij}(t)| \leq 1$ for $j = 0, 1$ and $|\dot{D}_i(t)| \leq 2$ for all times $t \geq 0$.

Now assume an adversary controls $\dot{v}_{i0}$, $\dot{v}_{i1}$ and hence $\dot{D}_i$, one can show that if a player sets $\dot{x}_i(t) = \frac{1}{2}(\dot{v}_{i1}(t) - \dot{v}_{i0}(t))$, then he could stay on $\mathcal{P}$ whenever he reaches the subspace.

Lemma 3.4. *If $s_i(0) \in \mathcal{P}$, and $\dot{x}_i(t) = \frac{1}{2}(\dot{v}_{i1}(t) - \dot{v}_{i0}(t))$, then $s_i(t) \in \mathcal{P}\ \forall\ t \geq 0$.*

Theorem 3.3. *Under the initial conditions $x_i(0) = \frac{1}{2}$ for all i, the following continuous dynamic, **Uncoupled Continuous Nash (UCN)**, has all players reach $\mathcal{P}$ in at most $\frac{1}{2}$ time units. Furthermore, upon reaching $\mathcal{P}$ a player never leaves.*

$$\dot{x}_i(t) = f(s_i(t), \dot{s}_i(t)) = \begin{cases} 1 & \text{if } s_i \notin \mathcal{P} \text{ and } v_{i1} \geq v_{i0}, \\ -1 & \text{if } s_i \notin \mathcal{P} \text{ and } v_{i1} < v_{i0}, \\ \frac{1}{2}(\dot{v}_{i1}(t) - \dot{v}_{i0}(t)) & \text{if } s_i \in \mathcal{P}. \end{cases}$$

Proof. From Lemma 3.4 it is clear that once a player reaches $\mathcal{P}$ they never leave the plane. It remains to show that it takes at most $\frac{1}{2}$ time units to reach $\mathcal{P}$.

Since $x_i(0) = x_i^*(0) = \frac{1}{2}$, it follows that if $s_i(0) \notin \mathcal{P}$ then $x_i^*(0) < \frac{1}{2}(1 + D_i(0))$. On the other hand, if we assume that $\dot{x}_i^*(t) = 1$ for $t \in [0, \frac{1}{2}]$, and that player preferences do not change, then it follows that $x_i^*(\frac{1}{2}) = 1$ and $x_i^*(\frac{1}{2}) \geq \frac{1}{2}(1 + D_i(\frac{1}{2}))$, where equality holds only if $D_i(\frac{1}{2}) = 1$. By continuity of $x_i^*(t)$ and $D_i(t)$ it follows that for some $k \leq \frac{1}{2}$, $s_i(k) \in \mathcal{P}$. It is simple to see that the same holds in the case where preferences change. □

3.4.2 Discrete Time-step Approximation

The continuous-time dynamics of the previous section hinge on obtaining expected payoffs in mixed strategy profiles, thus we will approximate expected payoffs via $\mathcal{Q}_{\beta,\delta}$. Our algorithm will have each player adjusting their mixed strategy over rounds, and in each round query $\mathcal{Q}_{\beta,\delta}$ to obtain the payoff values.

Since we are considering discrete approximations to **UCN**, the dynamics will no longer guarantee that strategy/payoff states stay on the plane $\mathcal{P}$. For this reason we define the following region around $\mathcal{P}$:

Definition 3.6. *Let $\mathcal{P}^\lambda = \{s_i \mid s_i \cdot \vec{n} \in [\frac{1}{2} - \lambda, \frac{1}{2} + \lambda]\}$, with normal vector $\vec{n} = (-\frac{1}{2}, \frac{1}{2}, 1)$. Equivalently, $\mathcal{P}^\lambda = \{s_i \mid x_i^* = \frac{1}{2}(1 + D_i) + c,\ c \in [-\lambda, \lambda]\}$.*

Just as in the proof of Lemma 3.3, we can use the fact that a player's regret is $D_i(1 - x_i^*)$ to bound regret on $\mathcal{P}^\lambda$.

Lemma 3.5. *The worst case regret of any strategy/payoff state in $\mathcal{P}^\lambda$ is $\frac{1}{8}(1 + 2\lambda)^2$. This is attained on the boundary: $\partial\mathcal{P}^\lambda = \{s_i \mid s_i \cdot \vec{n} = \frac{1}{2} \pm \lambda\}$.*

Corollary 1. *For a fixed $\alpha > 0$, if $\lambda = \frac{\sqrt{1+8\alpha}-1}{2}$, then $\mathcal{P}^\lambda$ attains a maximal regret of $\frac{1}{8} + \alpha$.*

We present an algorithm in the completely uncoupled setting, $\mathbf{UN}(\alpha, \eta)$, that for any parameters $\alpha, \eta \in (0, 1]$ computes a $(\frac{1}{8} + \alpha)$-Nash equilibrium with probability at least $1 - \eta$. Our algorithm is essentially discretising **UCN** and using approximate utility information obtained via sampling pure utility queries.

Since $x_i(t) \in [0, 1]$ is the mixed strategy of the i-th player at round t we let $x(t) = (x_i(t))_{i=1}^n$ be the resulting mixed strategy profile of all players at round t. Furthermore, we use the mixed strategy oracle $\mathcal{Q}_{\beta,\delta}$ from Lemma 3.2 that for a given mixed strategy profile x returns the vector of expected payoffs for all players with an additive error of β and a correctness probability of $1 - \delta$.

The following lemma is used to prove the correctness of $\mathbf{UN}(\alpha, \eta)$:

Lemma 3.6. *Suppose that $w \in \mathbb{R}^3$ with $\|w\|_\infty \leq \lambda$ and let function $h(x) = x \cdot \vec{n}$, where $\vec{n}$ is the normal vector of $\mathcal{P}$. Then $h(x + w) - h(x) \in [-2\lambda, 2\lambda]$. Furthermore, if $w_3 = 0$, then $h(x + w) - h(x) \in [-\lambda, \lambda]$.*

Proof. The statement follows from the following expression:

$$h(x + w) - h(x) = w \cdot \vec{n} = \frac{1}{2}(w_2 - w_1) + w_3.$$

□

Theorem 3.4. *With probability $1 - \eta$, $\boldsymbol{UN}(\alpha, \eta)$ correctly returns a $(\frac{1}{8} + \alpha)$-approximate Nash equilibrium by using $O(\frac{1}{\alpha^4} \log\left(\frac{n}{\alpha\eta}\right))$ queries.*

Proof. By Lemma 3.2 and a union bound, we can guarantee that with probability at least $1 - \eta$ all sample approximations to mixed payoff queries have an additive error of at most $\Delta = \frac{\lambda}{4}$. We will condition on this accuracy guarantee in the remainder of our argument. Now we can show that for each player there will be some round $k \leq N$, such that at the beginning of the round their strategy/payoff state lies in $\mathcal{P}^{\lambda/2}$. Furthermore, at the beginning of all subsequent rounds $t \geq k$, it will also be the case that their strategy/payoff state lies in $\mathcal{P}^{\lambda/2}$.

The reason any player generally reaches $\mathcal{P}^{\lambda/2}$ follows from the fact that in the worst case, after increasing x^* by Δ for N rounds, $x^* = 1$, in which case a player is certainly in $\mathcal{P}^{\lambda/2}$. Furthermore, Lemma 3.6 guarantees that each time x^* is increased by Δ, the value of $\widehat{s}_i \cdot \vec{n}$ changes by at most $\frac{\lambda}{2}$ which is why $\widehat{s}_i$ are always steered towards $\mathcal{P}^{\lambda/4}$. Due to inherent noise in sampling, players may at times find that $\widehat{s}_i$ slightly exit $\mathcal{P}^{\lambda/4}$ but since additive errors are at most $\frac{\lambda}{4}$. We are still guaranteed that true s_i lie in $\mathcal{P}^{\lambda/2}$.

The second half of step 4 forces a player to remain in $\mathcal{P}^{\lambda/2}$ at the beginning of any subsequent round $t \geq k$. The argumentation for this is identical to that of Lemma 3.4 in the continuous case.

Algorithm 1 UN(α, η)

Require:

Threshold: $\alpha > 0$

Confidence: $\eta > 0$

Initialisation:

$\lambda \leftarrow \frac{\sqrt{1+8\alpha}-1}{2}$

$\Delta \leftarrow \frac{\lambda}{4}$

$N \leftarrow \lceil \frac{2}{\Delta} \rceil$

$x_i(0) \leftarrow \frac{1}{2}$ for $i \in [n]$

Initial Gradient Estimate:

for $(i,j) \in [n] \times \{0,1\}$ **do**

$\widehat{v}_{ij}(0) = \left(\mathcal{Q}_{(\Delta, \frac{\eta}{N})}(j, x(0)_{-i})\right)_i$

Query Dynamics:

for $t = 1, ..., N$ **do**

for $i \in [n]$ **do**

for $j \in \{0,1\}$ **do**

$\widehat{v}_{ij}(t) \leftarrow \left(\mathcal{Q}_{(\Delta, \frac{\eta}{N})}(j, x(t)_{-i})\right)_i$

$\Delta\widehat{v}_{ij}(t) \leftarrow \widehat{v}_{ij}(t) - \widehat{v}_{ij}(t-1)$

if $\widehat{s}_i(t) = (\widehat{v}_{i1}(t), \widehat{v}_{i0}(t), x_i(t)) \notin \mathcal{P}^{\lambda/4}$ **then**

$x_i^*(t+1) \leftarrow x_i^*(t) + \Delta$

else

$x_i^*(t+1) \leftarrow x_i^*(t) + \frac{1}{2}(\Delta\widehat{v}_{i1}(t) - \Delta\widehat{v}_{i0}(t))$

return $x(N)$

Finally, the reason that individual probability movements are restricted to $\Delta = \frac{\lambda}{4}$ is that at the end of the final round, players will move their probabilities and will not be able to respond to subsequent changes in their strategy/payoff states. From the second part of Lemma 3.6, we can see that in the worst case this can cause a strategy/payoff state to move from the boundary of $\mathcal{P}^{\lambda/2}$ to the boundary of $\mathcal{P}^{\frac{3\lambda}{4}} \subset \mathcal{P}^{\lambda}$. However, λ is chosen in such a way so that the worst-case regret within $\mathcal{P}^{\lambda}$ is at most $\frac{1}{8} + \alpha$, therefore it follows that $\mathbf{UN}(\alpha, \eta)$ returns a $\frac{1}{8} + \alpha$ approximate Nash equilibrium. Furthermore, the number of queries is

$$(N+1)\left(\frac{1024}{\lambda^3}\log\left(\frac{8nN}{\eta}\right)\right) = \left(\frac{1}{\lambda}+1\right)\left(\frac{1024}{\lambda^3}\log\left(\frac{8n}{\lambda\eta}\right)\right).$$

It is not difficult to see that $\frac{1}{\lambda} = O(\frac{1}{\alpha})$ which implies that the number of queries made is $O\left(\frac{1}{\alpha^4}\log\left(\frac{n}{\alpha\eta}\right)\right)$ in the limit. □

3.4.3 Logarithmic Lower Bound

As mentioned in the preliminaries section, all of our previous results extend to stochastic utilities. In particular, if we assume that G is a game with stochastic utilities where expected payoffs are large with parameter $\frac{1}{n}$, then we can apply $\mathbf{UN}(\alpha, \eta)$ with $O(\log(n))$ queries to obtain a mixed strategy profile where no player has more than $\frac{1}{8} + \alpha$ incentive to deviate. Most importantly, for every $\ell > 2$, we can use the same methods as [29] to lower bound the query complexity of computing a mixed strategy profile where no player has more than $(\frac{1}{2} - \frac{1}{\ell})$ incentive to deviate.

Theorem 3.5. *For every $\ell > 2$, the query complexity of computing a mixed strategy profile where no player has more than $(\frac{1}{2} - \frac{1}{\ell})$ incentive to deviate for stochastic utility games is $\Omega(\log_{\ell(\ell-1)}(n))$. Alongside Theorem 3.4 this implies the query complexity of computing mixed strategy profiles where no player has more than $\frac{1}{8}$ incentive to deviate in stochastic utility games is $\Theta(\log(n))$.*

Proof. Suppose that we have n players and that $\ell > 2$. For every $b \in \{0,1\}^n$ we can construct a stochastic utility game G_b as follows: For each player i, the utility of strategy b_i is Bernoulli with bias $\frac{\ell}{\ell-1}$ and the utility of strategy $1 - b_i$ is bernoulli with bias $\frac{1}{\ell}$. Note that this game is trivially $\left(\frac{1}{n}\right)$-Lipschitz, as each player's payoff distributions are completely independent of other players' strategies.

Suppose that $\mathcal{G}$ is the uniform distribution on the set of all G_b, then using the same argumentation as Theorem 3 of [29], we get the following:

Theorem 3.6. *Let $\mathcal{A}$ be a deterministic payoff-query algorithm that makes at most $\log_{\ell(\ell-1)}(n)$ utility queries and outputs a mixed strategy x. If $\mathcal{A}$ performs on $\mathcal{G}$, then*

with probability more than $\frac{1}{2}$*, there will exist a player with a regret greater than* $\frac{1}{2} - \frac{1}{\ell}$ *in* x*.*

We can immediately apply Yao's minimax principle from [73] to this result to complete the proof. □

3.5 Achieving $\varepsilon < \frac{1}{8}$ with Communication

We return to continuous dynamics to show that we can obtain a worst-case regret of slightly less than $\frac{1}{8}$ by using limited communication between players, thus breaking the uncoupled setting we have been studying until now.

First of all, let us suppose that initially $x_i(0) = \frac{1}{2}$ for each player i and that **UCN** is run for $\frac{1}{2}$ time units so that strategy/payoff states for each player lie on $\mathcal{P} = \{s_i \mid x_i^* = \frac{1}{2}(1 + D_i)\}$. We recall from Lemma 3.3 that the worst case regret of $\frac{1}{8}$ on this plane is achieved when $x_i^* = \frac{3}{4}$ and $D_i = \frac{1}{2}$. We say a player is *bad* if they achieve a regret of at least 0.12, which on $\mathcal{P}$ corresponds to having $x_i^* \in [0.7, 0.8]$. Similarly, all other players are *good*. We denote $\theta \in [0, 1]$ as the proportion of players that are bad. Furthermore, as the following lemma shows, we can in a certain sense assume that $\theta \leq \frac{1}{2}$.

Lemma 3.7. *If* $\theta > \frac{1}{2}$*, then for a period of* 0.15 *time units, we can allow each bad player to shift to their best response with unit speed, and have all good players update according to* ***UCN*** *to stay on* $\mathcal{P}$*. After this movement, at most* $1 - \theta$ *players are bad.*

Proof. If i is a bad player, in the worst case scenario, $\dot{D}_i = 2$, which keeps their strategy/payoff state, s_i, on the plane $\mathcal{P}$. However, at the end of 0.15 time units, they will have $x_i^* > 0.85$, hence they will no longer be bad. On the other hand, since the good players follow the dynamic, they stay on $\mathcal{P}$, and at worst, all of them become bad. □

Observation 5. *After this movement, players who were bad are the only players possibly away from* $\mathcal{P}$ *and they have a discrepancy that is greater than* $0 \cdot 1$*. Furthermore, all players who become bad lie on* $\mathcal{P}$*.*

We can now outline a continuous-time dynamic that utilises Lemma 3.7 to obtain a $(\frac{1}{8} - \frac{1}{220})$ maximal regret.

1. Have all players begin with $x_i(0) = \frac{1}{2}$
2. Run **UCN** for $\frac{1}{2}$ time units.
3. Measure, θ, the proportion of bad players. If $\theta > \frac{1}{2}$ apply the dynamics of Lemma 3.7.

4. Let all bad players use $\dot{x}_i^* = 1$ for $\Delta = \frac{1}{220}$ time units.

Theorem 3.7. *If all players follow the aforementioned dynamic, no single player will have a regret greater than* $\frac{1}{8} - \frac{1}{220}$.

In essence one shows that if Δ is a small enough time interval (less than 0.1 to be exact), then all bad players will unilaterally decrease their regret by at least 0.1Δ and good players won't increase their regret by more than Δ. The time step $\Delta = \frac{1}{220}$ is thus chosen optimally.

Proof. We have seen via Lemma 3.7 that after step 3 the proportion of bad players is at most $\theta \leq \frac{1}{2}$, we wish to show that step 4 reduces maximal regret by at least $\frac{1}{220}$ for every bad player while maintaining a low regret for good players.

Since after step 3 all bad players remain on $\mathcal{P}$, we can consider an arbitrary bad player on the plane $\mathcal{P}$ with regret $r = D(1 - x^*)$. Let us suppose that we allow all bad players to unilaterally shift their probabilities to their best response for a time period of $\Delta < 0.4 \leq D$ units (the bound implies bad player preferences do not change). This means that the worst case scenario for their regret is when their discrepancy increases to $D + 2\theta\Delta$. If we let r' be their new regret after this move, we get the following:

$$r' = (D + 2\theta\Delta)(1 - x^* - \Delta) = D(1 - x^*) + 2\theta\Delta(1 - x^*) - D\Delta - 2\theta\Delta^2,$$

$$= r - 2\theta\Delta^2 + (2\theta(1 - x^*) - D)\,\Delta.$$

However, we can use our initial constraints on D and x^* from the fact that the players were bad, along with the fact that $\theta \leq \frac{1}{2}$ to obtain the following:

$$2\theta(1 - x^*) \leq (1 - x^*) \leq 0.3 < 0.4 \leq D.$$

Hence as long as $\Delta < 0.4$, $r' < r$ hence we can better the new bad players, without hurting the good players by choosing a suitably small value of Δ.

To see that we don't hurt good players to much, suppose that we have a good player with discrepancy D and best-response mass, x^*. By definition, their initial regret is $r = D(1 - x^*) < 0.12$. There are two extreme cases to what can happen to their regret after the bad players shift their strategies in step 4. Either their discrepancies increase by $2\theta\Delta$, in which case preferences are maintained, or either discrepancies decrease by $2\theta\Delta$ and preferences change (which can only occur when $2\theta\Delta > D$). For the first case we can calculate the new regret r' as follows:

$$r' = (D + 2\theta\Delta)(1 - x^*) = r + 2\theta(1 - x^*)\Delta \leq r + (1 - x^*)\Delta \leq r + \Delta.$$

This means that the total change in regret is at most Δ. Note that if a player was originally bad and then shifted according to Lemma 3.7 then their discrepancy is at least 0.1. For this reason if we limit ourselves to values of $\Delta < 0.1$, then all such players will always fall in this case since their preferences cannot change.

Now we analyse the second case where preferences switch. Since we are only considering $\Delta < 0.1$, then we can assume that all such profiles must lie on $\mathcal{P}$. In this case we get the following new regret:

$$r' = (2\theta\Delta - D)(x) = r + 2\theta x^*\Delta - D \leq r + x^*\Delta - D \leq r + \Delta.$$

Consequently, in the scenario that preferences change, the change of regret is bounded by Δ as well. This means that for $\Delta < 0.1$, the decrease in regret for bad players is at least:

$$2\theta\Delta^2 + (D - 2\theta(1 - x^*))\Delta > 0.1\Delta.$$

And for such time-steps Δ, the regret for good players increases by at most Δ. Thus under these bounds, the optimal value is $\Delta = \frac{1}{220}$ which gives rise to a maximal regret of $\frac{1}{8} - \frac{1}{220} = \frac{137}{1100}$. □

As a final note, we see that this process requires one round of communication in being able to perform the operations in Lemma 3.7, that is we need to know if $\theta > \frac{1}{2}$ or not to balance player profiles so that there are at most the same number of bad players to good players. Furthermore, in exactly the same fashion as $\mathbf{UN}(\alpha, \eta)$, we can discretise the above process to obtain a query-based algorithm that obtains a regret of $\frac{1}{8} - \frac{1}{220} + \alpha < \frac{1}{8}$ for arbitrary α.

3.6 Extensions

In this section we address three extensions to our previous results:

- (Section 3.6.1) We extend the algorithm **UCN** to large binary-action games with a more general largeness parameter $\gamma = \frac{c}{n} \in [0, 1]$, where c is a constant.
- (Section 3.6.2:) For n-player, k-action games, we consider arbitrary Lipschitz constants $\gamma = \frac{c}{n} \in [0, 1]$ and construct a new continuous dynamic, $\mathbf{QR}(\gamma)$ that utilizes a non-linear subspace of strategy/payoff states.
- (Section 3.6.3) We once again consider n-player large games with k actions, and largeness parameter $\gamma = \frac{c}{n}$. In this setting we describe an algorithm that uses a new uncoupled approach that is substantially different from other continuous dynamics from this chapter.

3.6.1 Continuous Dynamics for Binary-action Games with Arbitrary γ

We recall that for large games, the largeness parameter γ denotes the extent to which players can affect each others' utilities. Instead of assuming that $\gamma = \frac{1}{n}$ we now let $\gamma = \frac{c}{n} \in [0, 1]$ for some constant c. We show that we can extend **UCN** and still ensure a better than $\frac{1}{2}$-equilibrium. We recall that for the original **UCN**, players converge to a linear subspace of strategy/payoff states and achieve a bounded regret. For arbitrary $\gamma = \frac{c}{n}$, we can extend this subspace of strategy/payoff states as follows:

$$\mathcal{P}_\gamma = \left\{ (x^*, D) \mid x^* = \min\left(\frac{1}{2} + \frac{D}{2c}, 1\right) \right\}.$$

where D and x^* represent respectively a player's discrepancy and probability allocated to the best response. For $c = 1$ we recover the subspace $\mathcal{P}$ as in **UCN**. Furthermore, if $|\dot{x^*}| \leq 1$ for each player, then $|\dot{D}| \leq 2c$, which means that we can implement an update as follows:

$$\dot{x^*} = \frac{\dot{D}}{2c}.$$

This leads us to the following natural extension to Theorem 3.3:

Theorem 3.8. *Under the initial conditions $x_i(0) = \frac{1}{2}$ for all i, the following continuous dynamic, **UCN**-γ, has all players reach $\mathcal{P}_\gamma$ in at most $\frac{1}{2}$ time units. Furthermore, upon reaching $\mathcal{P}_\gamma$ a player never leaves, i.e.*

$$\dot{x_i^*}(t) = f(D_i(t), \dot{D}_i(t)) = \begin{cases} 1 & \text{if } s_i \notin \mathcal{P}_\gamma, \\ 0 & \text{if } s_i \in \mathcal{P}_\gamma \text{ and } x_i^* > \frac{1}{2} + \frac{D_i}{2c}, \\ \frac{\dot{D}_i}{2c} & \text{otherwise .} \end{cases}$$

Notice that unlike **UCN**, this dynamic is no longer necessarily a continuously differentiable function with respect to time when $c > 1$. However, it is still continuous.

Once again, we note that for all strategy/payoff states, regret can be expressed as

$$R = (1 - x^*)D,$$

from which we can prove the following:

Theorem 3.9. *Suppose that $\gamma = \frac{c}{n}$ and that a player's strategy/payoff state lies on $\mathcal{P}_\gamma$, then his regret is at most $\frac{c}{8}$ for $c \leq 2$ and his regret is at most $\frac{1}{2} - \frac{1}{2c}$ for $c > 2$. Furthermore, the equilibria obtained are also c-WSNE.*

Proof. If $c \leq 2$, then regret is maximised when $D = \frac{c}{2}$ and consequently when $x^* = \frac{3}{4}$. This results in a regret of $\frac{c}{8}$. On the other hand, if $c > 2$, then regret is maximised when $D = 1$ and consequently $x^* = \frac{1}{2} + \frac{1}{2c}$. This results in a regret of $\frac{1}{2} - \frac{1}{2c}$.

As for the second part of the theorem, from the definition of $\mathcal{P}_\gamma$ and from the definition of ε-WSNE in Section 2.3 it is straightforward to see that when $D \geq c$, $x^* = 1$ which means that no weight is put on the strategy whose utility is at most c from that of the best response. □

Thus we obtain a regret that is better than simply randomizing between both strategies, although as should be expected, the advantage goes to zero as the largeness parameter increases.

Discretisation and Query Complexity

In the same way as **UN**-(α, η), where we discretised **UN**, Theorem 3.9 can be discretised to yield the following result.

Theorem 3.10. *For a given accuracy parameter α and correctness probability η, we can implement a query-based discretisation of **UCN**-γ that with probability $1 - \eta$ correctly computes an ε-approximate Nash equilibrium for*

$$\varepsilon = \begin{cases} \frac{c}{8} + \alpha, & \text{if } c \leq 2, \\ \frac{1}{2} - \frac{1}{2c} + \alpha, & \text{if } c > 2. \end{cases}$$

Furthermore the discretisation uses $O\left(\frac{1}{\alpha^4}\left(\frac{n}{\alpha\eta}\right)\right)$ queries.

3.6.2 Continuous-time Quantal Response Dynamics

In this section we construct a continuous-time dynamic akin to **UCN** from Section 3.4. Our dynamic computes approximate well-supported Nash equilibria for k-action, γ-large games with n players, where each of these parameters is arbitrary. This scenario is more complicated than the binary-action case due to the fact that modifications to a player's strategy can encompass adding or taking away probability mass from multiple different pure strategies. Since we are interested in completely decoupled protocols, we will take the perspective of an arbitrary player and generalise the notion of strategy/payoff states from Section 3.4.

Definition 3.7 (Generalised strategy/payoff state)**.** *For an arbitrary player i, the* strategy/payoff state *for player i is defined as the ordered triple $s_i = (v_{i1}, ..., v_{ik}, x_{i1}, ..., x_{ik}) \in [0, 1]^{2k}$, where v_{ij} are expected utilities for playing pure actions j, and x_{ir} denotes the player's probability of playing action r. When the specific identity of a player is unimportant, we simplify the above notation to $s = (v_1, ..., v_k, x_1, ..., x_k)$.*

In all the analysis that follows, we assume the perspective of a generic player and hence use the latter notation from the definition.

Lemma 3.8. *Suppose that the rate of change of a player's mixed strategy satisfies the condition* $\sum_{i=1}^{k} |\dot{x}_i| \leq 2$. *Then the total effect on other players' utilities is at most* γ.

Proof. Since $\sum x_i = 1$, it follows that $\sum \dot{x}_i = 0$. Let I^+ be the set of all i such that $\dot{x}_i > 0$ and I^- be the rest. It should be straightforward to see that $\sum_{i \in I^+} |\dot{x}_i| \leq 1$ and that the same holds for I^-. Furthermore, $\sum_{i \in I^+} \dot{x}_i + \sum_{i \in I^-} \dot{x}_i = 0$, hence for every increase in probability there is a matching decrease. For this reason, we can consider the total change in probability as occuring in "pairwise" increments and decrements that eventually total out the complete change in probability. Since the pairwise changes don't ever change utilities by more than γ, then the claim follows. □

Corollary 2. *If all players satisfy the above constraints, then any player's utilities changes by at most* $n\gamma$.

From this point onwards, we assume that all players are restricted in their movements of probability by the above rule ($\sum_{i=1}^{k} |\dot{x}_i| \leq 2$), and as before, we will find a subspace of all strategy/payoff states with the property that an arbitrary player can remain on it in an uncoupled scenario via valid updates, and that in addition, regret on the subspace is bounded.

Definition 3.8 (Quantal Response Subspace)**.** *For a given randomness parameter* $\lambda \geq 0$, *we define the following subspace:*

$$\mathcal{QR}_\lambda = \left\{ s = (v_1, ..., v_k, x_1, ..., x_k) \mid x_i = Q_i^\lambda(\vec{v}) = \frac{e^{\lambda v_i}}{\sum_{j=1}^{k} e^{\lambda v_j}} \right\},$$

where we have referenced the family of quantal response functions Q_i^λ *from Chapter 2.*

We will show that for $\lambda = \frac{1}{2n\gamma}$ a player can reach $\mathcal{QR}_\lambda$ relatively quickly and remain on it with valid probability updates such that $\sum_{i=1}^{k} |\dot{x}_i| \leq 2$.

Theorem 3.11. *Under the initial conditions* $\lambda = \frac{1}{2n\gamma}$ *and* $x_{ir} = \frac{1}{k}$, *for any player* i *and all pure strategies* $r = 1, .., k$, *the following continuous dynamic,* $\boldsymbol{QR}(\gamma)$, *has all players reach* $\mathcal{QR}_\lambda$ *in at most* $O(\lambda + \log(k))$ *time units. Furthermore, upon reaching* $\mathcal{QR}_\lambda$, *each player remains on* $\mathcal{QR}_\lambda$.

For $r \geq 2$:

$$\dot{x}_{ir} = f(x_{i2}, ..., x_{ir}, \dot{x}_{i2}, ..., \dot{x}_{ir}) = \begin{cases} x_{ir}, & \text{if } x_{ir} < Q_r(\vec{v}_i), \\ -x_{ir}, & \text{if } x_{ir} > Q_r(\vec{v}_i), \\ \lambda \sum_{j=1}^{k} (\dot{v}_{ir} - \dot{v}_{ij}) x_{ir} x_{ij}, & \text{if } x_{ir} = Q_r(\vec{v}_i). \end{cases}$$

For $r = 1$:

$$\dot{x}_{i1} = -\sum_{j=2}^{k} \dot{x}_{ij}.$$

Proof. First we will show that for $\lambda = \frac{1}{2n\gamma}$ a player can remain on $\mathcal{QR}_\lambda$. If we assume that $x_{ir} = Q_r(\vec{v}_i) = \frac{e^{\lambda v_{ir}}}{\sum_j e^{v_{ij}}}$ as a function of time, then it is straightforward to see that probability time derivatives have the following form:

$$\dot{x}_i = \lambda \sum_{j=1}^{k} (\dot{v}_i - \dot{v}_j) x_i x_j.$$

Since this is precisely the third case for $\mathbf{QR}(\gamma)$, the claim holds.

Now we wish to show that $|\dot{x}_{ir}| \leq x_{ir}$ for $r \geq 2$. This is trivial for the first two cases of the dynamic, but for the third we can bound the magnitude of this derivative:

$$|\dot{x}_{ir}| = \lambda x_{ir} |\sum_{j=1}^{k} (\dot{v}_{ir} - \dot{v}_{ij}) x_{ij}| \leq \lambda x_{ir} \sum_{j=1}^{k} |(\dot{v}_{ir} - \dot{v}_{ij})| x_{ij}.$$

Furthermore, if for the time being we assume that $|\dot{v}_{ir}| \leq n\gamma$ for all r, then we get $|(\dot{v}_{ir} - \dot{v}_{ij})| \leq 2n\gamma$ and we can further bound the above expression to get our desired expression:

$$|\dot{x}_{ir}| \leq \lambda x_{ir} \sum_{j=1}^{k} (2n\gamma) x_{ij} = \lambda (2n\gamma) x_{ir} = x_{ir}.$$

Finally, since we have defined $\dot{x}_{i1} = -\sum_{j=2}^{k} \dot{x}_{ij}$, we obtain the following:

$$|\dot{x}_{i1}| = |\sum_{j=2}^{k} \dot{x}_{ij}| \leq \sum_{j=2}^{k} |\dot{x}_{ij}| \leq \sum_{j=2}^{k} x_{ij} \leq 1.$$

Hence we satisfy $\sum_{j=1}^{k} |\dot{x}_{ij}| \leq 2$, which from Lemma 3.8 and Corollary 2 justifies the assumption that $|\dot{v}_{ir}| \leq n\gamma$ for all r.

Finally, it remains to show that under this dynamic, a player reaches $\mathcal{QR}_\gamma$ in at most $O(\lambda + \log(k))$ time units. To do so, we demonstrate that for an arbitrary $r = 2, ..., k$, $x_{ir} = Q_r(\vec{v}_i)$ after the required time.

We have already shown that once $x_{ir}(t) = Q_r(\vec{v}_i(t))$ for a given time t, then $x_{ir}(t') = Q_r(\vec{v}_i(t'))$ for subsequent $t' > t$. Let t^* be the first time such that $x_{ir}(t^*) = Q_r(\vec{v}_i(t^*))$. From the first two cases of the dynamic, we can use the initial conditions $x_{ir}(0) = \frac{1}{k}$ to solve for $x_{ir}(t)$ for $t < t^*$ as the following:

$$x_{ir}(t) = \begin{cases} \frac{1}{k} e^{t}, & \text{if } \frac{1}{k} = x_{ir}(0) < Q_r(\vec{v}_i(0)), \\ \frac{1}{k} e^{-t}, & \text{if } \frac{1}{k} = x_{ir}(0) > Q_r(\vec{v}_i)(0). \end{cases}$$

As an upper bound to the worst-case t^*, we consider when $\frac{1}{k}e^t = 1$ and when $\frac{1}{k}e^{-t} = \frac{1}{1+(k-1)e^\lambda}$ (the second term is the smallest probability attainable under quantal response with rationality parameter λ). In doing so, we obtain

$$t^* \leq \max\left(\log(k), \log\left(\frac{1+(k-1)e^\lambda}{k}\right)\right) = O(\lambda + \log(k)),$$

which completes the proof of Theorem 3.11.

□

Now for a specific λ, we wish to find bounds on the worst-case regret a player can suffer on $\mathcal{QR}_\lambda$. I.e. a combination of utilities $v_1, ..., v_k$ such that the corresponding quantal response probability masses amount to a large regret. In order to do so, we assume without loss of generality that $v_1 \geq v_i \geq 0$ for all i, so that we can write the regret in terms of v_1 and a player's expected utility, $V = \frac{\sum v_i e^{v_i}}{\sum e^{v_j}}$:

$$R = v_1 - V.$$

In the following analysis, we relax the bounds on v_i to find a global maximum on R that still holds in our restricted case. Since v_1 is fixed, maximising regret is equivalent to minimising $\log(V)$. To do so, we take partial derivatives and equate to 0:

$$\frac{\partial}{\partial v_i}(\log(V)) = \frac{(1+\lambda v_i)e^{\lambda v_i}}{\sum_{j=1}^k v_j e^{\lambda v_j}} - \frac{\lambda e^{\lambda v_i}}{\sum_{j=1}^k e^{\lambda v_j}} = 0.$$

After simplification we get the following expression for v_i which is irrespective of the index i:

$$v_i = V - \frac{1}{\lambda} = \alpha^*.$$

It follows that a globally maximum regret is obtained when all other utilities $v_i = \alpha$ are uniform. Given this, we can reformulate our expected utility as follows:

$$V = \frac{v_1 e^{\lambda v_1} + (k-1)\alpha e^{\lambda\alpha}}{e^{\lambda v_1} + (k-1)e^{\lambda\alpha}}.$$

We take logarithms and differentiate to find the critical value α^*:

$$\frac{\partial V}{\partial \alpha} = \frac{(k-1)(1+\lambda\alpha)e^{\lambda\alpha}}{v_1 e^{\lambda v_1} + (k-1)\alpha e^{\lambda\alpha}} - \frac{(k-1)\lambda e^{\lambda\alpha}}{e^{\lambda v_1} + (k-1)e^{\lambda\alpha}} = 0.$$

If we let $\beta = \lambda(\alpha - v_1)$, then we can simplify the above to

$$1 + (k-1)e^\beta + \beta = 0,$$

for which the solution makes use of the Lambert W function and has the following form:

$$\beta = -W\left(\frac{(k-1)}{e}\right) - 1,$$

which leads to

$$\alpha^* = v_1 - \frac{1 + W\left(\frac{(k-1)}{e}\right)}{\lambda}.$$

However, we recall that we had an expression for α^* in terms of V and λ:

$$\alpha^* = V - \frac{1}{\lambda},$$

which leads to the following:

$$V - \frac{1}{\lambda} = v_1 - \frac{1 + W\left(\frac{(k-1)}{e}\right)}{\lambda}, \text{ and}$$

$$R = v_1 - V = \frac{W\left(\frac{(k-1)}{e}\right)}{\lambda} = 2n\gamma W\left(\frac{(k-1)}{e}\right) = f_R(n, \gamma, k).$$

Here we have defined the function $f_R = 2n\gamma W\left(\frac{(k-1)}{e}\right)$ on the right hand side as the upper bound term on regret that occurs in this analysis.

We recall that for $x > e$ the Lambert W function satisfies the bound $W(x) < \log(x)$, which means that for $k \geq 9$ the following regret bounds hold:

$$R < \frac{\log(k-1) - 1}{\lambda} = 2n\gamma\left(\log(k-1) - 1\right).$$

Hence if a player reaches the $\mathcal{QR}_\lambda$, their regret is at most $O(n\gamma \log k)$.

It is important to note that in the bound above, α can be negative, which can lead to the above upper bound not being tight, as the worst-case strategy/payoff state in the extended regret function is no longer realisable under the bounds $v_i \in [0, v_1]$.

To get around this, we note that $\alpha < 0$ happens precisely when $v_1 - \frac{1}{\lambda} < f_R(n, \gamma, k)$. This means that the above bound is tight up until $f_R(n, \gamma, k)$ reaches $v_1 - \frac{1}{\lambda}$. This term is highest at $1 - \frac{1}{\lambda} = 1 - 2n\gamma$.

So when $f_R(n, \gamma, k) > 1 - \frac{1}{\lambda}$ there is no value of v_1 that will invoke a positive minimal α value, hence the minimum value of V is obtained on the boundary of $(v_2, ..., v_k) \in [0, v_1]^{k-2}$, and it is straightforward to see that the minimum boundary value is when $v_1 = 1$ and all other v_i are 0. This leads to a maximal regret of the following form:

$$R \leq \frac{k-1}{e^\lambda + k - 1}.$$

Theorem 3.12 (Final Bounds)**.** *For a given combination of n, γ and $k \geq 9$, $\boldsymbol{QR}(\gamma)$ obtains the following maximal regret, R, on $\mathcal{QR}_{(2n\gamma)^{-1}}$ after $O(\lambda + \log(k))$ time:*

$$R \leq \begin{cases} 2n\gamma(\log(k-1)) - 2n\gamma, & \text{if } 2n\gamma \log(k-1) \leq 1, \\ \frac{k-1}{e^{\frac{1}{2n\gamma}} + (k-1)} & \text{otherwise.} \end{cases}$$

3.6.3 Block Equilibrium Computation for k-action Games

When the number of pure strategies per player is $k > 2$, the initial "strawman" idea corresponding to Observation 3 is to have all n players randomise uniformly over their k strategies. Notice that the resulting regret may in general be as high as $1 - \frac{1}{k}$. In this section we give a new uncoupled dynamic for computing approximate equilibria in k-action games where (for largeness parameter $\gamma = \frac{1}{n}$) the worst-case regret approaches $\frac{3}{4}$ as k increases, hence improving over uniform randomisation over all strategies. Recall that in general we are considering $\gamma = \frac{c}{n}$ for fixed $c \in [0, n]$. The following is just a simple extension of the payoff oracle $\mathcal{Q}_{\beta,\delta}$ to the setting with k actions: for any input mixed strategy profile x, the oracle will with probability at least $1 - \delta$, output payoff estimates for x with error at most β for all n players.

Estimating payoffs for mixed profiles in k-action games. Given a payoff oracle $\mathcal{Q}_U$ and any target accuracy parameter β and confidence parameter δ, consider the following procedure to implement an oracle $\mathcal{Q}_{\beta,\delta}$:

- For any input mixed strategy profile x, compute a new mixed strategy profile $x' = (1 - \frac{\beta}{2})x + (\frac{\beta}{2k})\mathbf{1}$ such that each player i is playing uniform distribution with probability $\frac{\beta}{2}$ and playing distribution x_i with probability $1 - \frac{\beta}{2}$.
- Let $m = \frac{64k^2}{\beta^3} \log(8n/\delta)$, and sample m payoff queries randomly from x', and call the oracle $\mathcal{Q}_U$ with each query as input to obtain a payoff vector.
- Let $\widehat{U}_{i,j}$ be the average sampled payoff to player i for playing action j.[3] Output the payoff vector $(\widehat{U}_{ij})_{i\in[n],j\in\{0,1\}}$.

As in previous sections, we begin by assuming that our algorithm has access to $\mathcal{Q}_M$, the more powerful query oracle that returns exact expected payoffs with regards to mixed strategies. We will eventually show in section 3.6.3 that this does not result in a loss of generality, as when utilising $\mathcal{Q}_{\beta,\delta}$ we incur a bounded additive loss with regards to the approximate equilibria we obtain.

The general idea of Algorithm 2 is as follows. For a parameter $N \in \mathbb{N}$, every player uses a mixed strategy consisting of a discretised distribution in which a player's probability is divided into N quanta of probability $\frac{1}{N}$, each of which is allocated to a single pure strategy. We refer to these quanta as "blocks" and label them $B_1, \ldots, B_N$. Initially, blocks may be allocated arbitrarily to pure strategies. Then in time step t, for $t = 1, \ldots, N$, block t is reallocated to the player's best response to the other players' current mixed strategies.

[3] If the player i never plays an action j in any query, set $\widehat{U}_{i,j} = 0$.

The general idea of the analysis of Algorithm 2 is the following. In each time step, a player's utilities change by at most $n\gamma/N = c/N$. Hence, at the completion of Algorithm 2, block N is allocated to a nearly-optimal strategy, and generally, block $N - r$ is allocated to a strategy whose closeness to optimality goes down as r increases, but enables us to derive the improved overall performance of each player's mixed strategy.

Algorithm 2 Block Equilibrium Computation **BU** (performed by each player)

Require:
Parameter: $N \in \mathbb{N}$

Initialisation:
Blocks $B_1, \ldots, B_N$; {a block represents $\frac{1}{N}$ of the player's mixed strategy }
Allocate each B_i to arbitrary $j \in [k]$;

Block Updates:
for $t = 1, \ldots, N$ **do**
Observe expected utilities $(U_j)_{j\in[k]}$ to current mixed strategy profile $\vec{x} = (\vec{x}_i)_{i\in[n]}$;
Reallocate B_t to best response to $\vec{x}$;

return $\vec{x}_i = (x_j)_{j\in[k]}$

Theorem 3.13. ***BU*** *returns a mixed strategy profile* $(\vec{x}_i)_{i\in[n]}$ *that is an* ε*-NE where:*

$$\varepsilon = \begin{cases} c\left(1 + \frac{1}{N}\right), & \text{if } c \leq \frac{1}{2}, \\ 1 - \frac{1}{4c} + \frac{3}{2N}, & \text{if } c > \frac{1}{2}. \end{cases}$$

Notice for example that for $\gamma = \frac{1}{n}$ (i.e. putting $c = 1$), each player's regret is at most $\frac{3}{4} + \frac{3}{2N}$, so we can make this approach $\frac{3}{4}$ since N is a parameter of the algorithm.

Proof. For an arbitrary player $i \in [n]$, in each step $t = 1, ..., N$, probability block B_t is re-assigned to i's current best response.

Since every player is doing the same transfer of probability, by the largeness condition of the game, one can see that every block's assigned strategy incurs a regret that increases by at most $\frac{2c}{N}$ at every time step. This means that at the end of N rounds, the j-th most recently updated block will at worst be assigned to a strategy that has $\min\{1, \frac{2cj}{N}\}$ regret. This means we can bound a player's total regret as follows:

$$R \leq \sum_{i=1}^{N} \min\left\{1, \frac{2cj}{N}\right\} \cdot \frac{1}{N}$$

There are two important cases for this sum: when $2c \leq 1$ and when $2c > 1$. In the first case:

$$R \leq \sum_{i=1}^{N} \frac{2ci}{N^2} = n\gamma\left(1 + \frac{1}{N}\right).$$

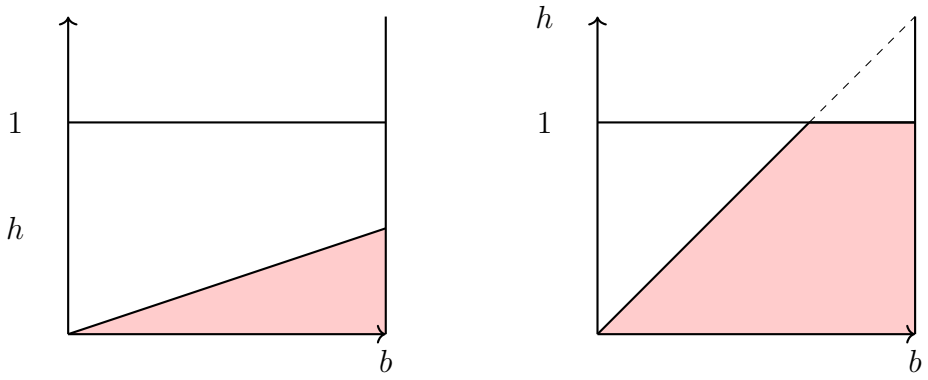

Figure 3.2: Visualisation of $\mathcal{A}^{b,h}$ when $h \leq 1$ (Left) and $h > 1$ (Right).

In order to compute the second, we let $\ell = \lfloor \frac{N}{2c} \rfloor$ and get:

$$R \leq \left(\sum_{i=1}^{\ell} \frac{2ci}{N^2} \right) + (N - \ell) \cdot \frac{1}{N} \leq 1 - \frac{1}{4c} + \frac{3}{2N}.$$

□

In fact, we can slightly improve the bounds in Theorem 3.13 via introducing a dependence on k. In order to do so, we need to introduce some definitions first.

Definition 3.9. *For $b, h > 0$, we construct $\mathcal{A}^{b,h}$ by taking a right triangle of base length, b, and height, h, and truncating it at height 1 if $h > 1$ (resulting in a trapezoid). If $h \leq 1$, the triangle remains untruncated. We call $\mathcal{A}^{b,h}$ the truncated triangle in the Cartesian plane with a base of length, b, and height, h. See Figure 3.2 for a visualisation.*

Definition 3.10. *For a given truncated triangle $\mathcal{A}^{b,h}$ and a partition of the base, $\mathcal{P} = \{x_1, ..., x_r\}$ where $0 \leq x_1 \leq \ldots \leq x_r \leq b$, we denote the left sum of $\mathcal{A}^{b,h}$ under $\mathcal{P}$ by $LS(\mathcal{A}^{b,h}, \mathcal{P})$ (for reference see Figure 3.3) and define it as follows:*

$$LS(\mathcal{A}^{b,h}, \mathcal{P}) = \sum_{i=1}^{|\mathcal{P}|} \left(\min \left\{ \frac{h}{b} x_i, 1 \right\} \right) (x_{i+1} - x_i).$$

With these definitions in hand, we can set up a correspondence between the worst case regret of **BU** and left sums of $\mathcal{A}^{b',2cb'}$ where $b' = 1 + \frac{1}{N}$. Suppose in the process of **BU** a player has blocks $B_1, ..., B_N$ in the queue. Furthermore, without loss of generality, suppose that their k strategies are sorted in ascending order of utility so that $u_1, ..., u_k$ where u_j is the expected utility of the j-th strategy at the end of the process. Furthermore, let $R_j = u_1 - u_j$ (i.e. the regret of strategy j), so that we also have $0 = R_1 \leq R_2 \leq ... \leq R_k \leq 1$. If N is much larger than k, then by the pigeon-hole principle, many blocks will be assigned

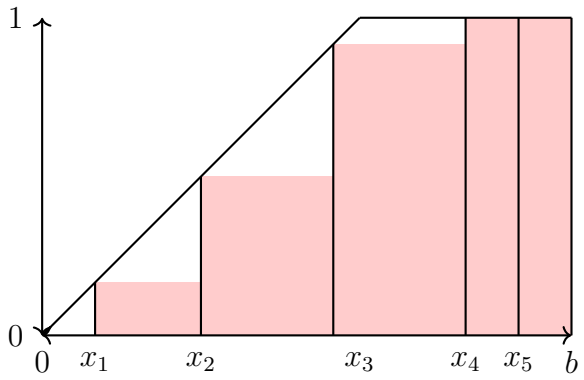

Figure 3.3: Example of left sum of five-element partition of base in the case where $h > 1$.

to the same strategy, and hence will incur the same regret. However, as in the analysis of the previous bounds, each block has restrictions as to how much regret their assigned strategy can incur due to the largeness condition of the game. In particular, the assigned strategy of block B_b (assigned b turns ago) can only be assigned to a strategy j such that $R_j \leq \min\left\{1, \frac{2cb}{N}\right\}$. For such an assignment, since the block has probability mass $\frac{1}{N}$, it contributes a value of $R_j \cdot \left(\frac{1}{N}\right)$ to the overall regret of a player. Hence for fixed regret values $(R_1, .., R_k)$, we can pick a valid assignment of these values to blocks and get an expression for total regret that can be visualised geometrically in Figure 3.4.

The next important question is what valid assignment of blocks to regret values results in the maximal amount of total regret for a player. In Figure 3.4, Block 1 is assigned to strategy 1, Blocks 2,3, and 7 are assigned to strategy 2, blocks 4 and 5 are assigned to strategy 3, block 5 is assigned to strategy 4 and finally blocks 8 and 9 are assigned to strategy 5. One can see that this does not result in maximal regret. Rather it is simple to see that a greedy allotment of blocks to regret values results in maximal total regret. Such a greedy allotment can be described as follows: assign as many possible (their regret constraints permitting) blocks at the end of the queue to R_k, then repeat this process one-by-one for R_i earlier in the queue. This is visualised in Figure 3.5, and naturally leads to the following result:

Lemma 3.9. *Let $b' = 1 + \frac{1}{N}$. For any fixed $R_1, ..., R_k$, the worst case assignment of probability blocks to strategies corresponds to a left sum of $\mathcal{A}^{b',2cb'}$ for some partition of $[0, b']$ with cardinality at most $k - 1$.*

This previous lemma reduces the problem of computing worst case regret to that of computing maximal left sums under arbitrary partitions. To that end, we define the precise worst-case partition value we will be interested in.

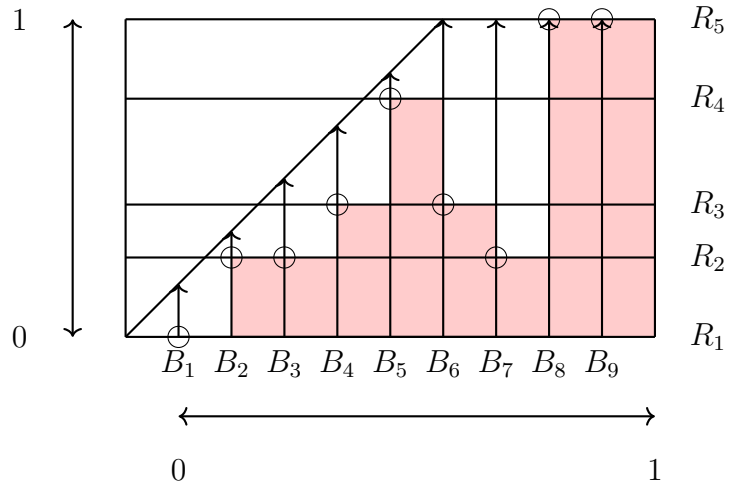

Figure 3.4: For $N = 9$ and $k = 5$, and $c > \frac{1}{2}$, this shows a visualisation of a feasible allotments of regret values to blocks after **BU**. Note that this does not exhibit worst-case regret.

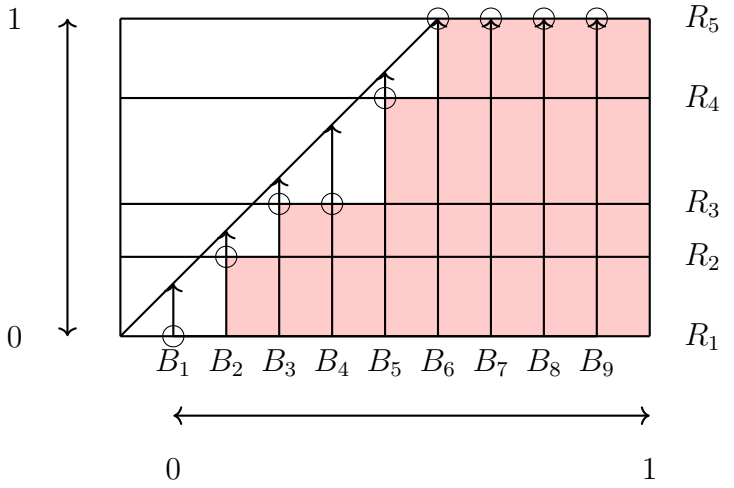

Figure 3.5: For $N = 9$ and $k = 5$, and $c > \frac{1}{2}$, this shows a visualisation of a feasible allotments of regret values to blocks after **BU**. Unlike Figure 3.4, this does exhibit worst-case regret.

Definition 3.11. *For a given $\mathcal{A}^{b,h}$, let us denote the maximal left sum under partitions of cardinality k by $\mathcal{A}_k^{b,h}$. Mathematically, the value is defined as follows:*

$$\mathcal{A}_k^{b,h} = \sup_{|\mathcal{P}|=k} LS(\mathcal{A}^{b,h}, \mathcal{P}).$$

We can explicity compute these values which in turn will bound a player's maximal regret.

Lemma 3.10. $\mathcal{A}_k^{1,1} = \left(\frac{1}{2}\right)\left(\frac{k}{k+1}\right)$ *which is obtained on the partition* $\mathcal{P} = \{\frac{1}{k+1}, \frac{2}{k+1}, ..., \frac{k}{k+1}\}$.

Proof. This result follows from induction on k and self-similarity of the original triangle. For $k = 1$, our partitions consist of a single point $x \in [0, 1]$ hence the area under the triangle will be $\mathcal{A}_1^{1,1}(x) = (1-x)x$ which as a quadratic function of x has a maximum at $x = \frac{1}{2}$. At this point we get $\mathcal{A}_1^{1,1}(x) = \frac{1}{2} \cdot \frac{1}{2}$ as desired.

Now let us assume that the lemma holds for k, we wish to show that it holds for $k+1$. Any $(k+1)$-element partition must have a left-most element, x_1. We let $\mathcal{A}'(x)$ be the maximal truncated area for a $(k+1)$-element partition, given that $x_1 = x$. By fixing x we add an area of $x(1-x)$ under the triangle and we are left with k points to partition $[x, 1]$. We notice however, that we are thus maximising truncated area under a similar triangle to the original, where the scaling factor is $(1-x)$. We can therefore use the inductive assumption and get the following expression:

$$\mathcal{A}'(x) = (1-x)x + (1-x)^2\mathcal{A}_k^{1,1} = (1-x)x + \frac{1}{2}(1-x)^2\left(\frac{k}{k+1}\right).$$

It is straightforward to see that $\mathcal{A}'(x)$ is maximised when $x = \frac{1}{k+2}$. Consequently the maximal truncated area arises from the partition where $x_i = \frac{i}{k+2}$ which in turn proves our claim.

□

Via linear scaling, and a case analysis, one can extend the above result to arbitrary values of b, h and k.

Corollary 3. *For $b, h > 0$ and $k \geq 2$, we obtain the following expressions for $\mathcal{A}_k^{b,h}$:*

$$\mathcal{A}_k^{b,h} = \begin{cases} \left(\frac{bh}{2}\right)\left(\frac{k}{k+1}\right), & \text{if } h < \frac{k+1}{k}, \\ b(1 - \frac{1}{2h} - \frac{1}{2hk}) & \text{otherwise.} \end{cases}$$

Proof. Let us first consider $h \leq 1 < \frac{k+1}{k}$. In this case, the triangle at hand is untruncated, hence we can scale the result of Lemma 3.10 to get the desired expression. Now let us suppose that $1 < h < \frac{k+1}{k}$. For this case, let us consider $\mathcal{B}^{b,h}$ to be the the triangle with base b and height h that, unlike $\mathcal{A}^{b,h}$, is not truncated. As in the previous paragraph, we can scale the result of Lemma 3.10 and see that the largest k-element left sum for

$\mathcal{B}^{b,h}$ occurs for the partition $\mathcal{P} = \{\frac{b}{k+1}, \frac{2b}{k+1}, ..., \frac{kb}{k+1}\}$. The fact that $h < \frac{k+1}{k}$ implies that $\frac{kb}{k+1} < \frac{b}{h}$. Consequently, the left sum at $\mathcal{P}$ for $\mathcal{B}^{b,h}$ is also a left sum for $\mathcal{A}^{b,h}$. Since $\mathcal{A}^{b,h} \subset \mathcal{B}^{b,h}$, it follows that this partition also gives a maximal k-element partition for left sums of $\mathcal{A}^{b,h}$ and thus the claim holds.

As for $h > \frac{k+1}{k}$, let us define $\ell = \frac{b}{h}$ and $b' = \ell\left(\frac{k+1}{k}\right)$. Similar to the proof of Lemma 3.10, we let $\mathcal{A}'(x)$ be the maximal truncated area for a k-element partition, given that $x_k = x$. We begin by showing that $\mathcal{A}'$ achieves its maximum in $[0, \ell]$. To show this, we suppose that $x^* \in (\ell, b]$ is a maximiser of $\mathcal{A}'$. Ultimately x^* is the final element of a k-element partition, $\mathcal{P}$, which gives rise to an optimal left sum for $\mathcal{A}^{b,h}$. Suppose that y is the $(k-1)$-th element of $\mathcal{P}$. We consider two cases: when $\ell \leq y < x^*$, and when $y < \ell < x^*$. In the first scenario, we notice that removing x^* from $\mathcal{P}$ results in the same left sum albeit with a $(k-1)$-element partition. This in turn means that the choice of x^* was not optimal as we had assumed. In the second scenario, we notice that replacing x^* with ℓ in $\mathcal{P}$ results in a larger left sum, once again contradicting the optimality of x^*. Both of these contradictions prove that without loss of generality, we can consider $x^* \in [0, \ell]$.

In what follows we will show that in fact $x^* = \ell$ is an optimiser of $\mathcal{A}'$. Let us recall that we defined $b' = \ell\left(\frac{k+1}{k}\right)$. From the fact that $h > \frac{k+1}{k}$, we know that $\ell < b' < b$. Suppose that $x \in [0, \ell]$ and that $\mathcal{P}$ is the optimal k-element partition that gives rise to a maximal left sum in $\mathcal{A}^{b,h}$ such that $x_k = x$. The left sum that arises from $\mathcal{P}$ can be decomposed into two parts: a left sum of $\mathcal{B}^{b', \frac{k+1}{k}}$, the untrunacated triangle of base b' and hight $\frac{k+1}{k}$, and the remaining portion of the overall left sum. The remaining portion of the left sum (arising from area beyond ℓ) is clearly maximised when $x_k = \ell$. On the other hand, from Lemma 3.10, we also know that the left sum of $\mathcal{B}^{b', \frac{k+1}{k}}$ is maximised at $x_k = b'\left(\frac{k}{k+1}\right) = \ell$. Since both components of $\mathcal{A}'$ are maximised at ℓ, our claim holds.

Using this previous point and Lemma 3.10 we have ultimately shown that the optimal k-element partition for $\mathcal{A}^{b,h}$ is in fact $\mathcal{P} = \{x_i\}_{i=1}^k$ where $x_i = \ell\left(\frac{i}{k}\right)$. Simple algebra thus gives the desired expression: $\mathcal{A}_k^{b,h} = b(1 - \frac{1}{2h} - \frac{1}{2hk})$

□

Finally, we can combine everything above to obtain:

Theorem 3.14. *With access to a query oracle that computes exact expected utilities for mixed strategy profiles,* ***BU*** *returns an ε-approximate Nash equilibrium for*

$$\varepsilon = \begin{cases} c\left(\frac{k-1}{k}\right)\left(1 + \frac{1}{N}\right)^2, & \text{if } \frac{1}{2}\left(\frac{N}{N+1}\right) < c < \frac{1}{2}\left(\frac{k}{k-1}\right)\left(\frac{N}{N+1}\right), \\ 1 - \frac{1}{4c}\left(\frac{k}{k-1}\right) + \frac{1}{N} & \text{otherwise.} \end{cases}$$

Proof. This just a straightforward application of Lemma 3.9 and Corollary 3.

□

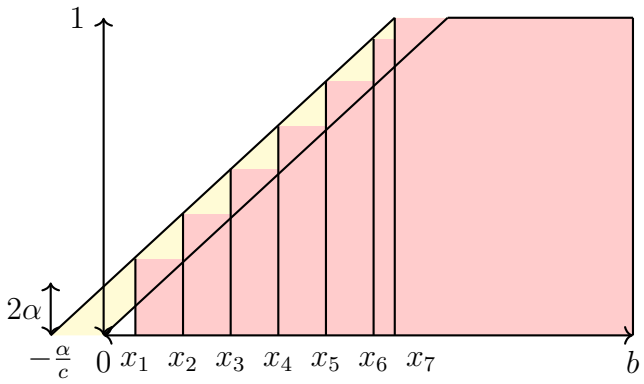

Figure 3.6: Example of α additive error in utility sampling. For this 7 element partition, regret bounds are increased by 2α and we get an augmented truncated triangle.

Query Complexity of Block Method

In the above analysis we assumed access to a mixed strategy oracle as we computed expected payoffs at each time-step for all players. When using $\mathcal{Q}_{\beta,\delta}$ however, there is an additive error and a bounded correctness probability to take into account.

In terms of the additive error, if we assume that there is an additive error of α on each of the N queries in **BU**, then at any time step, the b-th block will be assigned to a strategy that incurs at most $\left(\min\left\{1, \frac{2cb}{N}\right\} + 2\alpha\right)$ regret, which can visualised geometrically in figure 3.6, and which leads to the following extension of Lemma 3.9.

Lemma 3.11. *Suppose that $\beta > 0$ is fixed and that we define $b'' = (1 + \frac{1}{N} + \frac{\beta}{c})$ and $h'' = 2cb''$. In **BU**, if queries incorporate an additive error of β on expected utilities, for any fixed choice of $R_1, ..., R_k$, the worst case assignment of probability blocks B_b to strategies corresponds to a left sum of $\mathcal{A}^{b'',h''}$ for some partition of $[0, b'']$ with cardinality at most $k - 1$.*

Finally, since our approximate query oracle is correct with a bounded probability, in order to assure that the same additive error of β holds on all N queries of **BU**, we need to impose a correctness probability of $\frac{\delta}{N}$ in order to achieve the former with a union bound. This leads to the following query complexity result for **BU**.

Theorem 3.15. *For any $\alpha, \eta > 0$, if we implement **BU** using $\mathcal{Q}_{\beta,\delta}$ with $\beta = \alpha$ and $\delta = \frac{\eta}{N}$, with probability $1 - \eta$, we will obtain an ε-approximate Nash equilibrium for*

$$\varepsilon = \begin{cases} c\left(\frac{k-1}{k}\right)\left(1 + \frac{1}{N} + \frac{\alpha}{c}\right)^2, & \text{if } \frac{1}{2}\left(\frac{N}{N+1}\right) < c < \frac{1}{2}\left(\frac{k}{k-1}\right)\left(\frac{N}{N+1}\right), \\ 1 - \frac{1}{4c}\left(\frac{k}{k-1}\right) + \frac{1}{N} + \frac{\alpha}{c} & \text{otherwise.} \end{cases}$$

The total number of queries used is $\frac{64k^2}{\alpha^3}\log\left(\frac{8nN}{\delta}\right)$

It is important to note that once again, the regret has an extra term of the form $O(\frac{1}{N})$ in the number of probability blocks. Although this can be minimised in the limit, there is a price to be paid in query complexity, as this would involve a larger number of rounds in the computation of approximate equilibria.

3.6.4 Comparison Between Discrete Methods

We can compare the guarantees from our discrete methods from 3.6.1 and 3.6.3 when we let the number of strategies $k = 2$ and we consider largeness parameters $\gamma = \frac{c}{n} \in [0, 1]$. Furthermore, we consider how both methods compare when $N \to \infty$.

	$c \leq 1$	$1 \leq c \leq 2$	$c \geq 2$
UNC	$\frac{c}{8}$	$\frac{c}{8}$	$\frac{1}{2} - \frac{1}{2c}$
BU	$\frac{c}{2}$	$1 - \frac{1}{2c}$	$1 - \frac{1}{2c}$

One can see that **UNC** does better by a multiplicative factor of $\frac{1}{4}$ in the case of small c and better by an additive factor of $\frac{1}{2}$ for large c.

3.7 Conclusion and Future Directions

The obvious question raised by our results is the possible improvement in the additive approximation obtainable. Since *pure* approximate equilibria are known to exist for these games, the search for such equilibria is of interest. A slightly weaker objective (but still stronger than the solutions we obtain here) is the search for *well-supported* approximate equilibria in cases where $c > 1$ and for better *well-supported* approximate equilibria in general.

There is also the question of lower bounds, especially in the completely uncoupled setting. Our algorithms are randomised (estimating the payoffs that result from a mixed strategy profile via random sampling) and one might also ask what can be achieved using deterministic algorithms.

Finally, we also notice that one of our warm-up algorithms, **TwoStep**, as well as **BU** can in fact be implemented via access to best response oracles rather than utility oracles, whereas **UN**, **QR** and **OneStep** fundamentally make use of the utility values. Indeed the author of [2] has shown that if a large game is in addition anonymous, then a best response dynamic converges to an approximate pure Nash equilibrium with high probability. Therefore, it would be interesting to see what approximation guarantees one can get for equilibria in large games that are not anonymous via best response queries.

Chapter 4

Computing ε-WSNE with Best Response Queries

In this chapter we study algorithms that have query access to a player's best-response behaviour: an algorithm may query a mixed-strategy profile (i.e. probability distributions constructed by the algorithm, over each player's pure strategies) and learn a given player's best responses. Our focus is on standard bimatrix games, which is arguably the most natural starting-point for an investigation of a new query model. The solution concept of interest is approximate Nash equilibria (exact equilibria are typically impossible to find using finitely many such queries). A basic challenge is to identify algorithms that achieve this goal with good bounds on their query complexity (and also, ideally, their runtime complexity).

In more detail, we assume an $m \times n$ game G: a row player has m pure strategies and a column player has n pure strategies. G has two unknown $m \times n$ payoff matrices that represent payoffs to the players for all combinations of pure strategy choices they may make. A query consists of a probability distribution over the pure strategies of one of the players, and elicits an answer consisting of a best response for the other player (i.e. a pure strategy that maximises that player's expected payoff). The general question of interest is: how many queries are needed, as a function of m, n, ε to compute an ε-WSNE.

Using Kakutani's fixed point theorem, we reduce this question to a novel and more geometrical challenge in the design of query protocols. Suppose that the m-simplex Δ^m is partitioned into n convex regions having labels in $[n] = \{1, \ldots, n\}$. When we query a point $x \in \Delta^m$ we are told the label of the convex region that contains x. How many queries (in terms of m, n, ε) are needed in order to ensure that all points in Δ^m are within ε of a point whose label we know? We show how to achieve this using time and queries polynomial in $\log \varepsilon$ and $\max(m, n)$ provided that $\min(m, n)$ is constant. This leads to a polynomial query complexity algorithm for 2-player games, provided that one of the players has a constant number of strategies.

Finally, we explore the possibility of using a pairwise comparison oracle which is more powerful than best response oracles. In this setting we find that our geometric objective can be solved with a number of queries that is polynomial in m, n and $\log\left(\frac{1}{\varepsilon}\right)$. This in turn results in a query complexity for computing ε-WSNE in bimatrix games that is also polynomial in m, n and $\log\left(\frac{1}{\varepsilon}\right)$ via pairwise comparison queries.

4.1 Introduction

Earlier work in computational learning theory has studied exact learning of geometrical regions over a discretised domain, where algorithms are sought with query complexity logarithmic in a "resolution" parameter and binary search is repeatedly applied in a systematic way as in [14]. The authors of [26] specifically study the learnability of polytopes in this context, deriving query efficient algorithms, and precisely classifying polytopes learnable in this setting. These algorithms can be adapted to approximately learn a single polytope with membership queries, but the obtained notion of approximation is not directly applicable to computing ε-close labellings.

For a bimatrix game, simple ε-close labellings can be constructed by querying best responses at mixed strategies arising as uniform distributions over sufficiently large multisets of pure strategies. As a consequence of our main theorem, best responses to these multiset distributions contain enough information to compute approximate WSNE. This result is in the spirit of [4] and [49], where the authors aim to quantify specific k such that some approximate equilibrium arises as a uniform mixture over multisets of size k. We note in our scenario that there is also a guaranteed existence of an approximate equilibrium using sufficiently large multisets, however, *verifying* that a *specific* pair of mixed strategies is an approximate WSNE is not straightforward using only best response queries. This is in contrast to the verification of approximate equilibria via utility queries as studied in [4].

Separately, we note that the work in this chapter is possibly relevant to the search for a price equilibrium in certain markets. In [8], the authors study markets consisting of *strong-substitutes* demand functions for N different goods available in multiple discrete units. These markets are a generalisation of the *product-mix auction* of [45]; a basic task is to identify prices at which some desired bundle of the goods is demanded. Consider the space $(\mathbb{R}^+)^N$ of all price vectors. As analysed in [7] and [8], a strong-substitutes demand function partitions this price space into convex polytopes, each of which comprises the prices at which some particular bundle of goods is demanded. Consequently, this chapter relates to a setting where price vectors may be queried, and responses consist of demand bundles. The connection is imperfect, since the main objective in the context of [7] and

[8] would be to learn a price at which some target bundle is demanded, rather than the entire demand function. The ideas here may be useful for learning the values that the market has for various bundles.

Overview of Chapter

We begin by diving headfirst into the underlying geometric learning problem we are trying to solve. As mentioned before, we consider partitions of the unit m-simplex Δ^m into n convex polytopes, $P_1, ..., P_n$, and aim to approximately learn the partition with access to a membership oracle that for a given $x \in \Delta^m$, returns a polytope to which x belongs. The notion of approximation we study is that of *ε-close labellings*, a collection of empirical polytopes, $\{\widehat{P}_i\}_{i=1}^n$, such that $\widehat{P}_i \subseteq P_i$ for $i = 1, ..., n$ and $\cup_{i=1}^n \widehat{P}_i$ is an ε-net of $\Delta^m \subset \mathbb{R}^m$ in the ℓ_2 norm.

Note that in one dimension ($m = 1$) we can use binary search to solve this problem using $n \log(1/\varepsilon)$ queries. We generalise to higher dimension, exploiting convexity of the regions to reduce query usage in computing ε-close labellings. We present two algorithms for this task: Constant-Dimension Generalised Binary Search (CD-GBS), which for constant m uses $poly(n, \log\left(\frac{1}{\varepsilon}\right))$ queries, and Constant-Region Generalised Binary Search (CR-GBS), which uses CD-GBS as a subroutine and for constant n uses $poly(m, \log\left(\frac{1}{\varepsilon}\right))$ queries.

This problem is ultimately linked to the question of how to compute approximate (well-supported) Nash equilibra (ε-WSNE) using only best response information, obtained via queries in which the algorithm selects a mixed strategy profile and a player, and receives a best response for that player to the mixed profile. Via Kakutani's fixed-point theorem [40] we reduce this variant of equilibrium computation to finding ε-close labellings of polytope partitions. For $m \times n$ games where m is constant (or n equivalently, by symmetry), we show that an ε-WSNE can be computed using $poly(n, \log\left(\frac{1}{\varepsilon}\right))$ best response queries. In addition, we also consider using a pairwise comparison oracle, which is more powerful than best-response oracles. In this setting, we show that we can in fact compute an ε-close labelling with a query cost that is polynomial in m, n and $\log\left(\frac{1}{\varepsilon}\right)$. This in turn implies that an ε-WSNE can be computed with $poly(m, n, \frac{1}{\varepsilon})$ pairwise comparison queries.

Finally, we briefly delve into the problem of computing ε-WSNE in multiplayer games with best response queries. Unfortunately, as soon as there are more than two players, the geometric connection between computing ε-WSNE and learning polytope partitions of the simplex breaks down. Nonetheless, fixed-point techniques from Section 4.7 can still be applied in this setting, and we present a simple algorithm that computes an ε-WSNE

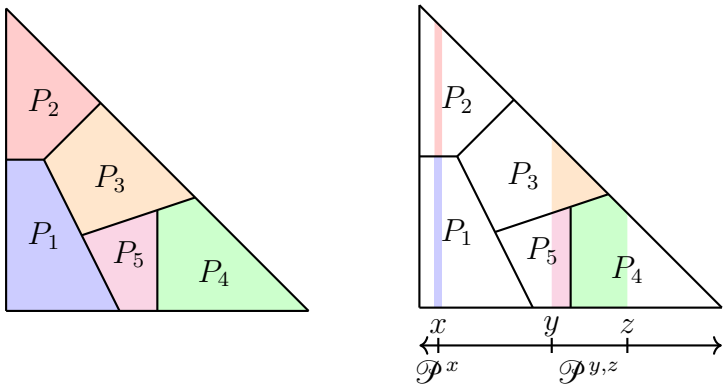

Figure 4.1: Polytope partition, cross-section and slices.

with a finite query complexity. To be more specific, in a game with n players each having k actions, our algorithm computes an ε-WSNE using $O\left(n\left(\frac{nk}{\varepsilon}\right)^{nk}\right)$ best response queries.

4.2 Preliminaries and Notation

Our main object of study will be families of polytopes that precisely cover the unit simplex, with the property that any two distinct polytopes from the family are either disjoint, meet at their boundary, or entirely coincide. Throughout, the polytopes we work with are convex.

Definition 4.1 ((m,n)-Polytope Partition)**.** *A (m,n)-polytope partition consists of a set of n convex polytopes in $\mathbb{R}^m$, $\mathscr{P} = \{P_1, ..., P_n\}$, with the following properties:*

- $\bigcup P_i = \Delta^m = \{x \in \mathbb{R}^m \mid \forall i,\ x_i \geq 0,\ \sum_i x_i \leq 1\}$.
- *For each $i \neq j$, either $relint(P_i) \cap relint(P_j) = \emptyset$ or $P_i = P_j$, where $relint(H)$ means the relative interior of H.*

Definition 4.2 (Cross-sections and Slices)**.** *Let $P \subset \mathbb{R}^m$ be a polytope and $\pi : \mathbb{R}^m \to \mathbb{R}$ the projection function into the first coordinate. For $x \in \mathbb{R}$, we define the* x-cross-section *of P as $P^x = \pi^{-1}(x) \cap P$. For any $I = [x,y] \subset \mathbb{R}$ we define the* $[x,y]$-slice *of P as $P^I = P^{x,y} = \pi^{-1}([x,y]) \cap P$. Suppose that $\mathscr{P} = \{P_i\}_i$ is an (m,n)-polytope partition. The definitions of cross-sections and slices extend to $\mathscr{P}^x = \{P_i^x\}_i$ and $\mathscr{P}^I = \mathscr{P}^{x,y} = \{P_i^{x,y}\}_i$.*

Figure 4.1 gives a visualisation of these two definitions. Notice that in the same figure, $\mathscr{P}^x$ is essentially a lower-dimensional polytope partition linearly scaled by a factor of $(1-x)$. This however, is not the case in general, as visible in Figure 4.2, where $\mathscr{P}^x$ fails the second condition of Definition 4.1. We distinguish between these two scenarios with the following formal definition:

Definition 4.3 (Non-Degenerate and Degenerate cross-sections)**.** *Let $\mathscr{P}$ be an (m,n)-polytope partition. For $x \in [0,1)$ let $f_x : \mathscr{P}^x \to \Delta^{m-1}$ be defined as $f_x(v_1,...,v_m) = \frac{1}{1-x}(v_2,...,v_m)$. If $f_x(\mathscr{P}^x)$ is an $(m-1,n)$-polytope partition, we say that $\mathscr{P}^x$ is a non-degenerate cross-section. Otherwise we say that $\mathscr{P}^x$ is a degenerate cross-section.*

The recursive structure of polytope partitions on non-degenerate cross-sections will be crucial to our constructions. Luckily, for any polytope partition, there are only a finite number of points $x \in [0,1)$ that give rise to degenerate cross-sections. Before showing this, we define an important discrete subset of $[0,1]$ given by the projections of vertices of polytopes under the same projection, π, used in Definition 4.2.

Definition 4.4 (Vertex Critical Coordinates)**.** *For a given polytope $P \subset \mathbb{R}^m$ let $V_P \subset \mathbb{R}^m$ be the vertex set of P. Define the set of* vertex critical coordinates *as $C_P = \pi(V_P) \subset \mathbb{R}$. If $\mathscr{P} = \{P_i\}_{i=1}^n$ is an (m,n)-polytope partition, then we extend our definition to define $C_{\mathscr{P}} = \bigcup_{i=1}^n C_{P_i} \subset [0,1]$ as the vertex critical coordinates of $\mathscr{P}$.*

Lemma 4.1. *Suppose that $\mathscr{P}$ is an (m,n)-polytope partition and that $x \in [0,1) \setminus C_{\mathscr{P}}$. Then $\mathscr{P}^x$ is non-degenerate.*

Proof. First we show that if $P \subset \mathbb{R}^m$ is an arbitrary polytope and $x \in \mathbb{R} \setminus C_P$ then $relint(P^x) = relint(P) \cap \pi^{-1}(x)$.

First of all, we notice that $P \neq P^x$ since we have assumed that x is not the projection of a vertex of P. Suppose that the affine dimension of P is $k \leq m$ so that P is full dimensional in the affine subspace H of dimension k. Let $z \in relint(P^x) \subset P^x$. Clearly $\pi(z) = x$, hence we simply need to show that $z \in relint(P)$. Suppose that this is not the case, then z lies on some boundary hyperplane to P in H. Call this boundary hyperplane D. D cannot lie entirely in $\pi^{-1}(x)$ due to the fact that x is not a critical coordinate. It follows that $D \cap \pi^{-1}(x)$ is thus a boundary hyperplane to P^x. This contradicts the fact that $z \in relint(P^x)$, thus establishing the fact that $z \in relint(P) \cap \pi^{-1}(x)$.

Suppose that $z \in relint(P) \cap \pi^{-1}(x)$. Since $relint(P) \subset P$, we know that $z \in P_x$. Furthermore, $z \in relint(P)$ means that for some $\varepsilon > 0$, the k-dimensional ball $B_\varepsilon(z) \cap H$ is entirely contained in P. Clearly this also holds for the $k-1$ dimensional ball $B_\varepsilon(z) \cap H \cap \pi^{-1}(x)$, establishing the fact that $z \in relint(P^x)$.

Let us return to the lemma statement which involves a polytope partition $\mathscr{P}$ instead of a single polytope P. If $x \in [0,1) \setminus C_{\mathscr{P}}$ then we have shown $relint(P_i^x) = relint(P_i) \cap \pi^{-1}(x)$ for all $P_i \in \mathscr{P}$, which from the fact that $\mathscr{P}$ satisfies the second condition of Definition 4.1 establishes the fact that $\mathscr{P}^x$ also satisfies this second condition. The fact that $\mathscr{P}^x$ satisfies the first condition of Definition 4.1 trivially follows from the fact that $\mathscr{P}$ covers Δ^m as per the first condition of Definition 4.1.

□

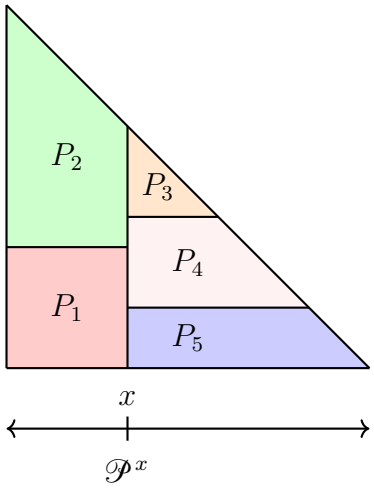

Figure 4.2: Degenerate cross-section at x.

4.2.1 Query Oracle Models

We study two natural query oracle models for polytope membership in any $\mathscr{P}$.

4.2.2 Query Oracle Models

We study two natural query oracle models for polytope membership in any polytope partition $\mathscr{P}$.

Definition 4.5 (Membership Query Oracles for Polytope Partitions)**.** *An (m, n)-polytope partition, $\mathscr{P}$ has the following membership query oracles:*

- *Lexicographic query oracle: $Q_\ell : \Delta^m \to [n]$, which for a given y returns the smallest index of polytope to which y belongs, namely $Q_\ell(y) = \min\{i \in [n] \mid y \in P_i\}$.*
- *Adversarial query oracle(s): $Q_A : \Delta^m \to [n]$, which can return any polytope to which y belongs. Namely Q_A is any function such that $Q_A(y) \in \{i \in [n] \mid y \in P_i\}$ for all $y \in \Delta^m$.*

When we wish to refer to an arbitrary oracle from the above models, we use the notation Q. Before continuing, we also clarify that for $A, B \subseteq \mathbb{R}^n$, we denote $Conv(A, B) \subseteq \mathbb{R}^m$ as the convex combination of the two sets. In addition, if $A_i \subseteq \mathbb{R}^m$ is an indexed family of sets with $i = 1, ..., r$, we denote $Conv(A_i \ | i = 1, ..., r) \subseteq \mathbb{R}^n$ as the convex combination of all A_i.

In what follows, we rigorously define our learning objective and subsequently begin by developing and analysing algorithms for the lexicographic query oracle first in Sections 4.3 and 4.4. Ultimately, we generalise our results from the lexicographic oracle to worst case adversarial oracles in Section 4.5 via upper-envelope polytope partitions.

4.2.3 ε-close Labellings

Upon making queries to Q, we can infer labels of $x \in \Delta^m$ by taking convex combinations. We abstract this notion in the following definition.

Definition 4.6 (Empirical Polytopes and Labellings)**.** *Suppose that $\mathscr{P}$ is an (m,n)-polytope partition and $S \subset \Delta^m$ is a finite set for which queries to Q have been made. Let $\widehat{P}_i = Conv(\{x \in S \mid Q(x) = i\}) \subset P_i$. We say each $\widehat{P}_i$ is an empirical polytope of P_i and that $\widehat{\mathscr{P}} = \{\widehat{P}_i\}$ is an empirical labelling of $\mathscr{P}$. Furthermore, we use the notation $\widehat{P}_\perp = \Delta^m \setminus \cup_{i=1}^n \widehat{P}_i$. to refer to points in Δ^m unlabelled under $\widehat{\mathscr{P}}$.*

An ε-net in the ℓ_2 norm for $\Delta^m \subset \mathbb{R}^m$ is a set $N_\varepsilon^m \subseteq \Delta^m$ with the property that for all $x \in \Delta^m$, there exists a $y \in N_\varepsilon^m$ such that $\|x - y\|_2 \leq \varepsilon$. Our learning goal is to use query access to an oracle, Q, to compute an empirical labelling $\widehat{\mathscr{P}}$ such that $\cup_{i=1}^n \widehat{P}_i$ is an ε-net of Δ^m.

Definition 4.7 (ε-close Labelling)**.** *Suppose that $\varepsilon \geq 0$ and that $\widehat{\mathscr{P}}$ is an empirical labelling for $\mathscr{P}$. If $\cup_{i=1}^n \widehat{P}_i$ is an ε-net of $\Delta^m \subset \mathbb{R}^m$ in the ℓ_2 norm, we say that $\widehat{\mathscr{P}}$ is an* ε-close labelling *of $\mathscr{P}$.*

Although ε-close labellings are defined for polytope partitions, we extend our terminology to also encompass slices of polytope partitions. As such, when we mention computing an ε-close labelling of $\mathscr{P}^{x,y}$, we mean an empirical labelling of $\mathscr{P}^{x,y}$ (in the same vein as Definition 4.6) with the property that the union of its empirical polytopes forms an ε-net of $(\Delta^m)^{x,y}$.

4.2.4 Learning in Thickness to Learning in Distance

Definition 4.8 (Thickness of Sets)**.** *Suppose that $Z \subseteq \mathbb{R}^m$ is a set. We define the* thickness *of Z as the largest radius of an ℓ_2 ball fully contained in Z and we denote it by $\tau(Z) = \sup\{\delta \geq 0 \mid \exists x \in Z \text{ with } B_\delta(x) \subseteq Z\}$ where $B_\delta(x) = \{y \in \mathbb{R}^m \mid \|x - y\|_2 \leq \delta\}$. In the language of convex geometry, $\tau(Z)$ is the depth of the Chebyshev centre of Z.*

For a polytope partition $\mathscr{P}$, if $\widehat{\mathscr{P}}$ is an ε-close labelling, then $\tau(\widehat{P}_\perp) \leq \varepsilon$, but the converse does not hold in general. Even though $\widehat{P}_\perp$ may be of small thickness, if it contains vertices of Δ^m, these vertices may be far from labelled points. The following results lead up to Lemma 4.4, a slightly weaker version of the converse. Lemma 4.4 shows that if we are able to learn an empirical labelling where the set of unlabelled points is of small enough thickness, then we will in fact have succeeded in learning an ε-close labelling, where any unlabelled point is close in distance to a labelled point.

We begin with Lemma 4.2, where informally, we show that for a full-dimensional convex polytope, P, and any given subset, $A \subsetneq P$, of bounded thickness, an arbitrary $x \in A$ is γ-close to $P \setminus A$, where γ is inversely proportional to $\tau(P)$ and proportional to the diameter of P. Ultimately, we wish to consider $A = \widehat{P}_\perp$ of bounded thickness, so this result is key in establishing the proximity of elements in $\widehat{P}_\perp$ to $\widehat{P}$, which is our objective in learning ε-close labellings.

Lemma 4.2. *Let $P \subset \mathbb{R}^m$ be a full-dimensional convex polytope with diameter given by $Diam(P) = \sup_{x,y \in P} \|x - y\|_2$.*

- *If $A \subsetneq P$ and $\gamma > \left(\frac{Diam(P)}{\tau(P)}\right) \tau(A)$, then $B_\gamma(x) \cap (P \setminus A) \neq \emptyset$ for all $x \in A$.*
- *If $A \subseteq P$ is such that $int(P) \setminus A \neq \emptyset$ ($int(P)$ refers to the interior of P) and $\gamma > \left(\frac{Diam(P)}{\tau(P)}\right) \tau(A)$, then $B_\gamma(x) \cap (int(P) \setminus A) \neq \emptyset$ for all $x \in A$.*

Proof. The proof of the first claim follows by considering the picture given in Figure 4.3. Pick an arbitrary $x \in A$. Due to the definition of thickness, there exists some $v \in P$ such that $B_{\tau(P)}(v) \subset P$. Consider the convex combination, $Conv(x, B_{\tau(P)}(v)) \subset P$. The furthest point in this convex combination from x is at the other extreme of $B_{\tau(P)}(v)$ from x, and we denote the distance between these two points by $z = \sup_{a \in B_{\tau(P)}(v)} \|x - a\|_2 \leq Diam(P)$. By similarity however, it now follows that if we consider $B_\gamma(x)$, and the fact that $Diam(P) \geq z$, a similar inscribed sphere of radius strictly greater than $\tau(A)$ will exist within $F = B_\gamma(x) \cap Conv(x, B_{\tau(P)}(v)) \subset B_\gamma(x) \cap P$. By definition, $\tau(F) > \tau(A)$. It follows that $F \not\subset A$, which proves the claim as $F \subset B_\gamma(x)$.

As for a proof of the second claim, it follows by considering the same picture above and noticing that $int(Conv(x, B_{\tau(P)}(v))) \subset int(P)$ as well as the fact that the former set is non-empty since P is of full dimension. □

In order to apply the previous lemma to polytope partitions, it is important we establish bounds on the thickness and diameter of Δ^m.

Lemma 4.3. *$Diam(\Delta^m) = \sqrt{2}$ and $\tau(\Delta^m) \geq \frac{1}{m+\sqrt{m}}$.*

Proof. For the first part of the statement, let us fix an $x \in \Delta^m$. If we consider the function $f_x(z) = \|x - z\|_2^2$, then this function is differentiable and clearly achieves local maximum when z is a vertex of Δ^m. It thus follows that the distance between two points in Δ^m is maximised when both are vertices. This in turn is maximal when both points are vertices not equal to the zero vector, in which case they are at distance $\sqrt{2}$ from each other.

As for the second part of the claim, we explicitly construct an inscribed sphere of the desired radius. Let $\lambda = \frac{1}{m+\sqrt{m}}$, and define $x = \lambda \vec{1} \subset \Delta^m$. Clearly $B_\lambda(x)$ is tangent to

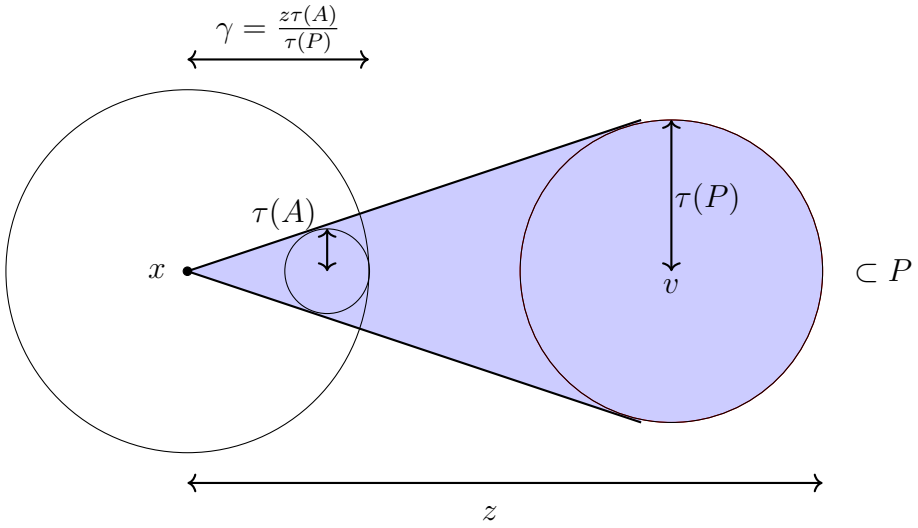

Figure 4.3: Proof of Lemma 4.2 .

Δ^m on axis-aligned faces (defined by the set of all z such that $\pi_i(z) = 0$ in the positive orthant). The remaining face is given by the set of z in the positive orthant such that $\|z\|_1 = 1$. The most extremal point of $B_\lambda(x)$ in the direction of this face is given by $\frac{1}{m}\vec{1}$, hence the sphere is inscribed in Δ^m. □

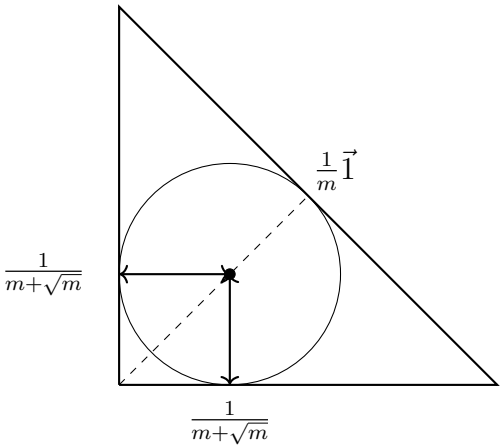

Figure 4.4: Proof of Lemma 4.3 .

Lemma 4.4. *Suppose that $\mathscr{P}$ is an (m, n)-polytope partition. Furthermore suppose that $\widehat{\mathscr{P}}$ is an empirical labelling with $\tau(\widehat{P}_\perp) < \varepsilon$. For any $\gamma > \sqrt{2}(m + \sqrt{m})\varepsilon$, it follows that $\widehat{\mathscr{P}}$ is a γ-close labelling. In particular, if $\gamma > 4m\varepsilon$, the claim also holds.*[1]

[1]An identical result which may be of separate interest holds if we consider partitions of arbitrary m-dimensional convex polytopes (not just Δ^m as per the definition of polytope partitions). As long as we can bound the thickness and diameter of the ambient convex polytope, learning in thickness translates to learning in distance.

Proof. From Lemma 4.3, we know that $\tau(\Delta^m) \geq \frac{1}{m+\sqrt{m}}$ and $Diam(\Delta^m) = \sqrt{2}$. Suppose that $x \in \widehat{P}_\perp$. From Lemma 4.2 our choice of γ implies $B_\gamma(x) \cap (\Delta_m \setminus \widehat{P}_\perp) \neq \emptyset$. This in turn means that $\widehat{\mathscr{P}}$ is a γ-close labelling. As for the final claim, this holds since $m \geq 1$. □

4.3 Constant-Dimension Generalised Binary Search for Q_ℓ

Let us first build some intuition for why generalisations of binary search lead to query efficient algorithms for computing ε-close labellings of (m, n)-polytope partitions.

Finding an ε-close labelling of a $(1, n)$-polytope partition using a lexicographic oracle is the same as approximately learning n sub-intervals of $[0, 1]$. Using binary search techniques and an optimal $O(n \log(\frac{1}{\varepsilon}))$ queries, we can compute an ε-close labelling.

Query efficiency comes from the fact that if x, y have the same label, it becomes unnecessary to further query any point in $[x, y]$. To be more specific, unless $[x, y]$ contains the boundary of a sub-interval, all labels can be inferred within $[x, y]$. Boundary points of intervals thus serve as "critical points" with respect to the query oracle Q_ℓ, where the information it provides changes.

We will use a higher-dimensional analogue of this property at the core of this section's main algorithm. At a high level, suppose that we have an (m, n)-polytope partition that we want to learn via queries and an algorithm for computing arbitrary ε-close labellings of $(m-1, n)$-polytope partitions. We can use this algorithm as a subroutine on the cross-sections of two coordinates $x \neq y$ and ask whether the convex combination of these two ε-close labellings will itself result in a $g(\varepsilon)$-close labelling of $\mathscr{P}^{x,y}$ (recall Definition 4.2) for a reasonable g.

Suppose that we could compute 0-close labellings (i.e. perfectly recover a polytope partition), it is clear that if we let $\mathscr{P}_V$ be the set of all vertices of all P_i in the polytope partition, then $\pi(\mathscr{P}_V)$ is a suitable set of critical points (not necessarily the smallest one though) in the sense that if $[x, y] \cap \pi(\mathscr{P}_V) = \emptyset$, then the convex combination of both lower-dimensional 0-close labellings for $\mathscr{P}^x$ and $\mathscr{P}^y$ will result in a 0-close labelling for $\mathscr{P}^{x,y}$. Taking the contrapositive of this, if the convex combination does not result in a 0-close labelling —a condition which can be verified— then we know there is a critical point in $[x, y]$. Thus we recover a binary-search mechanism, whereby we can isolate critical points up to a desired tolerance ε.

4.3.1 Warm-up: Learning Slices of Single Polytopes

We set up important groundwork by focusing on arbitrary polytopes $P \subset \mathbb{R}^m$. We let $\pi : \mathbb{R}^m \to \mathbb{R}$ be the projection function introduced in Definition 4.2, and we recall Definition 4.4 regarding the vertex critical coordinates of P denoted by C_P.

Lemma 4.5. *Suppose that $x, y \in \mathbb{R}$ are such that $[x, y] \bigcap C_P = \emptyset$. Then taking convex hulls of cross-sections we get $Conv(P^x, P^y) = P^{x,y}$.*

Proof. $[x, y] \cap C_P = \emptyset$ implies the vertices of the polytope $P^{x,y}$ lie in $\mathscr{P}^x$ and $\mathscr{P}^y$. Since the convex hull of the set of all vertices of a bounded polytope is the polytope itself, the claim follows. □

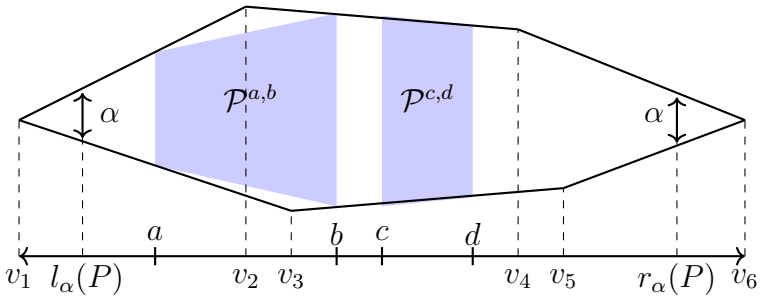

Figure 4.5: $Conv(P^a, P^b) \neq P^{a,b}$ and $Conv(P^c, P^d) = P^{c,d}$.

This property of polytopes whereby convex combinations give rise to complete information except when traversing a discrete set of critical points (visualised in Figure 4.5) is critical to CD-GBS. With query access to polytopes however, we no longer fully recover P^x perfectly, but instead an approximation given by an ε-close labelling, $\widehat{P}^x$. It becomes more subtle to show that by taking convex hulls of $\widehat{P}^x$ and $\widehat{P}^y$, we recover the desired information along $[x, y]$.

4.3.2 Necessary Machinery

We delve into the specifics of CD-GBS by defining some important machinery. We recall our notion of thickness in Definition 4.8, and see that it satisfies a sub-additivity property when the sets being considered are convex polytopes:

Lemma 4.6. *Let $P_1, .., P_k \subseteq \mathbb{R}^m$ be convex polytopes. Thickness of these sets satisfies a sub-additive bound: $\tau\left(\cup_i P_i\right) < \frac{10}{3}(\sum_i \tau(P_i))(m+1)^{3/2}$.*

Proof. Let $R = \frac{10}{3}(\sum_i \tau(P_i))(m+1)^{3/2}$. Suppose that $x \in \cup_i P_i$. We will show that $B_R(x)$ cannot be a subset of $\cup_i P_i$ via a volume argument. For this proof, we will let

$V(A)$ denote the volume of the set $A \subset \mathbb{R}^m$. We will also let $S(m, R)$ denote the volume of the hypersphere in m dimensions of radius R.

First of all, we need to show that for a given P_i, we have the following volume bound:

$$V(P_i \cap B_R(x)) \leq 2\tau(P_i)S(m-1, R).$$

This follows from Fritz John's Theorem [38] especially as referenced in [9]. The statement of this theorem says that if $K \subseteq \mathbb{R}^m$ is a convex body, then there exists a unique ellipsoid $\mathcal{E} \subseteq K$ of maximal volume, with the property that $\mathcal{E} \subseteq K \subseteq m\mathcal{E}$. Any higher-dimensional ellipsoid has at most m axes of symmetry, and for $\mathcal{E}$, it must be the case that the smallest axis is at most the thickness of the convex body: $\tau(K)$ (Otherwise there would be a sphere of radius larger than $\tau(K)$ inscribed in $\mathcal{E}$, contradicting the definition of thickness). Furthermore, since $K \subseteq m\mathcal{E}$, the projection of K onto this axis must be contained in a segment of length at most $2m\tau(K)$. This means that if we take an arbitrary polytope P_i, there exist two parallel supporting hyperplanes to P_i, call them H_1 and H_2 that are at most $2m\tau(P_i)$ apart. Call the convex body between these halfspaces H. Since the majority of the mass of a hypersphere is contained around its centre, it follows that the volume of the intersection of H with $B_R(x)$ is maximised when x is equidistant from H_1 and H_2. Furthermore, the volume of this cross-section is bounded by the distance between H_1 and H_2 multiplied by $S(m-1, R)$ which is at most $2\tau(P_i)S(m-1, R)$. Since $V(P_i \cap B_R(x)) \leq V(H \cap B_R(x))$, the claim holds.

Now if we take a union bound over all i, we get $V((\cup_i P_i) \cap B_R(x)) \leq 2m \sum_i \tau(P_i)S(m-1, R)$. If it were the case that the right hand side were strictly less than $S(m, R)$, we would have found an R such that $B_R(x)$ contains points not contained in any P_i. To this end, we use the following ratio:

$$\frac{S(m, R)}{S(m-1, R)} = R\sqrt{\pi}\frac{\Gamma(\frac{m+1}{2})}{\Gamma(\frac{m+2}{2})} \geq 0.6R(m+1)^{-1/2}.$$

The inequality uses Stirling's formula for the gamma function. We can therefore see that with our value $R > \frac{10}{3}(\sum_i \tau(P_i))(m+1)^{3/2}$, we get the desired volume bound. □

For a given polytope partition $\mathscr{P} = \{P_i\}_i$, it will be important to establish thickness bounds on P_i at specific cross-sections. The following definition allows us to associate to each polytope P_i a segment of $[0, 1]$ within which cross-sections of P_i are thick above a threshold.

Definition 4.9 (α-Critical Coordinates)**.** *Let $P \subset \mathbb{R}^m$ be a polytope. For $\alpha > 0$, we define $l_\alpha(P) = \inf\{x \in \mathbb{R} \mid \tau(P^x) \geq \alpha\}$ and $r_\alpha(P) = \sup\{x \in \mathbb{R} \mid \tau(P^x) \geq \alpha\}$ so that $\forall z \in \mathbb{R}$, $\tau(P^z) \geq \alpha$ if and only if $z \in [l_\alpha(P), r_\alpha(P)]$ (Here thickness is with respect to the natural embedding of P^x in $\mathbb{R}^{m-1}$). These are called* α-critical coordinates *for P.*

By combining this with Definition 4.4 we get the correct notion of critical coordinates mentioned at the beginning of Section 4.7.

Definition 4.10 (Critical Coordinates of a (m,n)-Polytope Partition)**.** *Suppose that $\mathscr{P} = \{P_1, ..., P_n\}$ is an (m,n)-polytope partition. For $\alpha > 0$, we let $C^{\alpha}_{\mathscr{P}}$ be the union of the sets of all vertex critical coordinates of all P_i as defined in Definition 4.4, and the set of all α-critical coordinates for all P_i as in Definition 4.9. Specifically, $C^{\alpha}_{\mathscr{P}} = (\cup_i C_{P_i}) \bigcup (\cup_i \{l_\alpha(P_i), r_\alpha(P_i)\})$.*

As mentioned in the beginning of this section, CD-GBS clusters queries around critical coordinates (up to a desired tolerance). For this reason it is important to bound the number of critical coordinates in a given (m,n)-Polytope partition.

Lemma 4.7. *If $\mathscr{P}$ is a (m,n)-polytope partition $|C^{\alpha}_{\mathscr{P}}| \leq \binom{n+m}{m} + 2n$.*

Proof. For any given (m,n)-polytope partition, $\mathscr{P}$, if a vertex occurs, it must be the case that out of the n polytopes in $\mathscr{P}$ and m boundary halfspaces of Δ^m, m of them meet. Furthermore, each collection of m polytopes and boundary halfspaces can give rise to only one vertex (which can be seen as a consequence of the fact that vertices are points in Δ^m). It follows that the set of all vertex critical coordinates is at most $\binom{n+m}{m}$ and the first part of the bound holds. As for the second half, there are at most two α-critical coordinates per P_i, which completes the expression above. □

With this machinery in hand, we are in a position to prove the main result necessary to demonstrate correctness of CD-GBS. We show that if $x, y \in [0,1]$ are such that $[x,y]$ contains no critical coordinates, then sufficiently fine empirical labellings of $\mathscr{P}^x$ and $\mathscr{P}^y$ with Q_ℓ will contain enough information to compute an ε-close labelling of $\mathscr{P}^{x,y}$ by simply taking convex combinations of these very empirical labellings at both cross-sections.

Lemma 4.8. *Given $m, n, \varepsilon > 0$ let $\alpha = \frac{\varepsilon}{20nm^{5/2}}$ and $\beta = \frac{\varepsilon^2}{85nm^{5/2}}$. Suppose that $\mathscr{P}$ is an (m,n)-polytope partition and that the following hold:*

- *$x, y \in [0,1]$ are such that $x < y \leq 1 - \frac{\varepsilon}{3}$.*
- *$[x,y] \cap C^{\alpha}_{\mathscr{P}} = \emptyset$.*
- *$\widehat{\mathscr{P}}^x$ and $\widehat{\mathscr{P}}^y$ are empirical labellings of $\mathscr{P}^x$ and $\mathscr{P}^y$ computed via Q_ℓ, such that $\cup_i \widehat{P}^x_i$ and $\cup_j \widehat{P}^y_j$ are β-nets for $(\Delta^m)^x$ and $(\Delta^m)^y$ respectively.*

Then $\bigcup_i Conv(\widehat{P}^x_i, \widehat{P}^y_i)$ is an ε-net of $(\Delta^m)^{x,y}$.

Proof. Let us define the following:

$$U = \{i \in [n] \mid [l_\alpha(P_i), r_\alpha(P_i)] \cap [x, y] = \emptyset\},$$

$$V = \{i \in [n] \mid [x, y] \subsetneq [l_\alpha(P_i), r_\alpha(P_i)]\}.$$

We call U the set of α-insignificant polytopes and V the set of α-significant polytopes. From the fact that $[x, y]$ contains no critical coordinates, we know that $U \cup V = [n]$ and from Lemma 4.1, we also know that for all $z \in [x, y]$, $\mathscr{P}^z$ is non-degenerate. We proceed by proving the following claims:

1. $V \neq \emptyset$.
2. Any point in the cross-section of an α-insignificant polytope is $\frac{2\varepsilon}{3}$ close to an α-significant polytope (within that same cross-section).
3. If $e \in P_j^x \setminus \left(\bigcup_{i=1}^n \widehat{P}_i^x\right)$, and $j \in V$, then there exists a $e' \in \widehat{P}_j^x$ such that $\|e - e'\|_2 < \frac{\varepsilon}{3}$.
4. If $w \in P_j^z$ for some $j \in V$ and $z \in [x, y]$, then there exists a $w' \in Conv(\widehat{P}_j^x, \widehat{P}_j^y) \cap \pi^{-1}(z)$ such that $\|w - w'\|_2 < \frac{\varepsilon}{3}$.

(2) and (4) suffice to prove the theorem. To see this, suppose that $w \in (\Delta^m)^{x,y}$. This means that $w \in P_i^z$ for some $i \in [n]$ and $z \in [x, y]$. If $i \in V$, then from (4) $\exists w' \in Conv(\widehat{P}_i^x, \widehat{P}_i^y) \subset \bigcup_i Conv(\widehat{P}_i^x, \widehat{P}_i^y)$ such that $\|w - w'\|_2 < \frac{\varepsilon}{3}$. On the other hand, if $i \in U$, then by (2) $\exists w' \in P_j^z$ for some $j \in V$ such that $\|w - w'\|_2 < \frac{2\varepsilon}{3}$. In turn by (1) again, $\exists w'' \in Conv(\widehat{P}_j^x, \widehat{P}_j^y) \subset \bigcup_i Conv(\widehat{P}_i^x, \widehat{P}_i^y)$ such that $\|w' - w''\|_2 < \frac{\varepsilon}{3}$. Using the triangle inequality $\|w - w''\|_2 < \varepsilon$, and hence $\bigcup_i Conv(\widehat{P}_i^x, \widehat{P}_i^y)$ is an ε-net of $(\Delta^m)^{x,y}$ in the ℓ_2 norm as desired.

Let us prove statement (1). We know that if $i \in U$, for all $z \in [x, y]$ it holds that $\tau(P_i^z) \leq \alpha$. Using the union bound from Lemma 4.6 we see $\tau(\cup_{i \in U} P_i^z) \leq \frac{10n\alpha m^{3/2}}{3} = \frac{\varepsilon}{6m}$. On the other hand, we also know that $\cup_{i \in U} P_i^z \subset (\Delta^m)^z \cong (1 - z)\Delta^{m-1}$. From Lemma 4.3, we know $\tau((\Delta^m)^z) \geq \frac{1-z}{((m-1)+\sqrt{m-1})}$. It follows that if $\frac{\varepsilon}{6m} < \frac{1-z}{((m-1)+\sqrt{m-1})}$, then $\cup_{i \in U} P_i^z \neq (\Delta^m)^z$. The condition $y \leq 1 - \frac{\varepsilon}{3}$ ensures that this happens for all $z \in [x, y]$. This in turn implies $V \neq \emptyset$.

Let us prove statement (2). From Lemma 4.3, we know $\frac{Diam((\Delta^m)^z))}{\tau((\Delta^m)^z)} \leq \sqrt{2}((m-1) + \sqrt{m-1})$. We can apply Lemma 4.2 in exactly the same fashion as Lemma 4.4 to get $\gamma_1 = \frac{2\varepsilon}{3} > \left(\frac{10n\alpha m^{3/2}}{3}\right) 4(m-1)$. We know that if $w \in \cup_{i \in U} P_i^z$, then $\exists w' \in \cup_{i \in V} P_i^z$ such that $\|w - w'\|_2 < \gamma_1 = \frac{2\varepsilon}{3}$, which is what we wanted to show.

Let us prove statement (3). Let us define $(\widehat{P}_j^x)_\perp = P_j^x \setminus \left(\bigcup_{i \in [n]} \widehat{P}_i^x\right)$. Note that these are the points in P_j^x that do not have any label whatsoever under the empirical labelling at x. Importantly, some points could have a label other than j if these points

are on the boundary of another polytope with a label that has higher priority in the lexicographic ordering. By the fact that we have a β-close labelling of $\mathscr{P}^x$, it must hold that $\tau((\widehat{P}_j^x)_\perp) \leq \beta$. Also, since $P_j^x \subset (\Delta^m)^x$, we know $Diam(P_j^x) \leq \sqrt{2}$ from Lemma 4.3. Since $j \in V$, we also know that $\tau(P_j^x) \geq \alpha$, hence $\tau((\widehat{P}_j^x)_\perp) \leq \beta < \alpha \leq \tau(P_j^x)$ which in turn implies that $int(P_j^x) \setminus (\widehat{P}_j^x)_\perp \neq \emptyset$. Let $\eta^* = \frac{1}{2}\left(\frac{\varepsilon}{3} - \left(\frac{\sqrt{2}}{\alpha}\right)\beta\right) > 0$ and let $\gamma_2 = \frac{\varepsilon}{3} - \eta^* > \left(\frac{\sqrt{2}}{\alpha}\right)\beta$ (the addition of the η^* gap is to help with the proof of statement (4)). We can use the second part of Lemma 4.2 to see $B_{\gamma_2}(e) \cap \left(int(P_j^x) \setminus (\widehat{P}_j^x)_\perp\right) \neq \emptyset$. Since all points in $int(P_j^x)$ only belong to P_j, it follows that under the lexicographic oracle one only sees the label j for those points. This implies that $B_{\gamma_2}(e) \cap \widehat{P}_j^x \neq \emptyset$, which in turn implies $\exists e' \in \widehat{P}_j^x$ such that $\|e - e'\|_2 < \gamma_2 = \frac{\varepsilon}{3} - \eta^* < \frac{\varepsilon}{3}$ as desired.

Finally, we prove statement (4). Since $[x, y]$ has no critical points, from Lemma 4.5 we know that $Conv(P_j^x, P_j^y) = P_j^{x,y}$, which in turn means that there exists a $a \in P_j^x$ and a $b \in P_j^y$ such that $w \in Conv(a, b)$. To be precise $w = Conv(a, b) \cap \pi^{-1}(z)$. Now if $a \in \widehat{P}_j^x$ and $b \in \widehat{P}_j^y$, then we are done. Let us suppose that this is not the case. We focus on a. If $a \in (\widehat{P}_j^x)_\perp$, the previously proved statement says there is some $a' \in \widehat{P}_j^x$ such that $\|a - a'\| < \frac{\varepsilon}{3}$. If $a \notin (\widehat{P}_j^x)_\perp \cup \widehat{P}_j^x$, then it must be the case that $a \in \widehat{P}_k^x \cap P_j^x$ for some other $k \in [n]$. This however only happens if $a \in P_j \cap P_k$ for some $P_k \neq P_j$, from the second property of polytope partitions from Definition 4.1 and the fact that using the lexicographic query oracle means that if $P_j = P_k$ and $j < k$ then $\widehat{P}_k = \emptyset$ always. Invoking the second property of Definition 4.1 again, we see that a lies on a bounding hyperplane of P_j. This in turn means that for every $\delta > 0$, $B_\delta(a) \cap int(P_j^x) \neq \emptyset$. Let us thus consider $\delta^* = \min\{\frac{\varepsilon}{3}, \frac{\eta^*}{2}\}$, where η^* is defined as in the previous paragraph. Let x^* be a point in $B_{\delta^*}(a) \cap int(P_j^x)$. Either $x^* \in \widehat{P}_j^x$ or $x^* \in (\widehat{P}_j^x)_\perp$. In the former case, since $\delta^* < \frac{\varepsilon}{3}$ we are done, we have succeeded in finding $a' = x^* \in \widehat{P}_j^x$ such that $\|a - a'\|_2 < \frac{\varepsilon}{3}$. In the latter case, from the previous paragraph, since $x^* \in (\widehat{P}_j^x)_\perp$ we know that $\exists a' \in \widehat{P}_j^x$ such that $\|x^* - a'\|_2 < \frac{\varepsilon}{3} - \eta^*$. Since $\delta^* < \frac{\eta^*}{2}$, we can use the triangle inequality to conclude that $\|a - a'\|_2 < \frac{\varepsilon}{3}$. In either case, we have proven what we wanted.

The same argumentation as the previous paragraph shows us that $\exists b' \in \widehat{P}_j^y$ such that $\|b - b'\|_2 < \frac{\varepsilon}{3}$. If we let $w' = Conv(a', b') \cap \pi^{-1}(z)$, then w' satisfies the requirements of statement (4) and we have finished our proof. □

For the following corollary, suppose that $\mathscr{P}$ is an (m, n) polytope partition and that $0 = t_0 < t_1, ..., < t_k = 1$ are points in $[0, 1]$. Furthermore suppose that $\beta = \frac{\varepsilon^2}{85nm^{5/2}}$ as in Lemma 4.8. For each t_i, if $t_i \notin C_{\mathscr{P}}^\alpha$, let $\widehat{\mathscr{P}}^{t_i}$ be a β-close labelling of $\mathscr{P}^{t_i}$, otherwise let $\widehat{\mathscr{P}}^{t_i} = \emptyset$. Let $\widehat{\mathscr{P}} = Conv_i(\widehat{\mathscr{P}}^{t_i})$ and for $i = 1, .., k$, let $I_j = [t_{j-1}, t_j]$. If $\widehat{\mathscr{P}}^{t_{i-1}, t_i}$ is an ε-close labelling of $\mathscr{P}^{t_{i-1}, t_i}$, we say that I_j is covered, otherwise we say I_j is uncovered.

Corollary 4. *For any collection of $\{t_i\}_{i=1}^k$, there are no more than $2C_{\mathscr{P}}^{\alpha}$ intervals I_j that are uncovered.*

Proof. Suppose that I_j is uncovered, then one of the following holds:

- Either t_{j-1} or t_j are in $C_{\mathscr{P}}^{\alpha}$,
- $t_{j-1}, t_j \notin C_{\mathscr{P}}^{\alpha}$ yet $Conv(\widehat{P}^{t_{j-1}}, \widehat{P}^{t_j})$ is not an ε-close labelling of $\mathscr{P}^{t_{j-1},t_j}$.

From the contrapositive of Lemma 4.8, the latter case implies $I_j \cap C_{\mathscr{P}}^{\alpha} \neq \emptyset$, hence in either case there is a critical coordinate in I_j. In the worst case each $x \in C_{\mathscr{P}}^{\alpha}$ lies on a t_j, causing both I_j and I_{j+1} to be uncovered. This implies that there are at most $2C_{\mathscr{P}}^{\alpha}$ intervals I_j that are uncovered. □

4.3.3 Specification of CD-GBS and Query Usage

Terms and Notation: The details of CD-GBS are presented in Algorithm 3. We recall our notation from Definition 4.3 where for $x \in [0, 1)$ we defined $f_x : (\Delta^m)^x \to \Delta^{m-1}$ given by $f_x(x, ..., v_m) = \frac{1}{1-x}(v_2, ..., v_m)$. We note that this is a bijection between both polytopes, hence it is well-defined to use f_x^{-1}. In addition, we let $\mathcal{D}^k = \{\frac{i}{2^k} \mid 1 \leq i \leq 2^i\}$ be the dyadic fractions of k-th power in the unit interval (excluding 0). For every $x \in \mathcal{D}^k$ we can associate the interval $I_x^k = [x - \frac{1}{2^k}, x]$. For each of these intervals $midpoint(I_x^k)$ denotes its midpoint. We also use the same language as Corollary 4.3.2 when we talk about whether I_x^k is covered or not (with respect to the current empirical labelling, $\widehat{\mathscr{P}}$, obtained from taking convex hulls of labels in Δ^m). We note that in order to have a well-defined base case of CD-GBS (which is equivalent to binary search), we let $\Delta^0 = \mathbb{R}^0 = \{0\}$. Finally, we say that a point $x \in [0, 1]$ is an uncovered critical point if $\widehat{\mathscr{P}}^x$ is computed via a recursive call to CD-GBS and for $(a, b) = B_{\varepsilon/2}(x) \cap [0, 1]$, it holds that $\widehat{\mathscr{P}}^{a,b}$ is not an ε-close labelling of $\mathscr{P}^{a,b}$.

Theorem 4.1. *If CD-GBS is given access to Q_ℓ for a (m, n)-polytope partition, it computes an ε-close labelling of $\mathscr{P}$ using at most $\left(\prod_{i=1}^m \left(\binom{n+i}{i} + 2n\right)\right) 2^{2m^2} \log^m\left(\frac{170nm^{5/2}}{\varepsilon}\right)$ membership queries. For constant m this constitutes $O(n^{m^2} \log^m\left(\frac{n}{\varepsilon}\right)) = poly(n, \log\left(\frac{1}{\varepsilon}\right))$ queries* [2].

[2] CD-GBS runs in polynomial time for constant m. The time-intensive operation consists of identifying uncovered intervals, but since the dimension of the ambient simplex is constant, each empirical polytope $\widehat{P}_i$ has at most a constant number of bounding hyperplanes. These hyperplanes can each be extruded by ε, and checking whether there exists a point outside all these extrusions can be done in time polynomial in n via brute force. In fact, all other algorithms in this chapter have efficient runtimes (in their relevant parameters) due to similar reasoning.

Proof. We first prove that CD-GBS indeed computes an ε-close labelling when given access to a valid Q_ℓ by inducting on m. It is straightforward to see that in the case $m = 1$, if CD-GBS is given access to a valid Q_ℓ for a $(1, n)$ polytope partition (a partition of the unit interval into conected subintervals), then it simply performs binary search on the interval $[0, 1] \cong \Delta^1$.

As for the inductive step, for $k = \lceil \log(2/\varepsilon) \rceil$, any two contiguous points of $\mathcal{D}^k$ are less than $\varepsilon/2$ away from each other. For now suppose that every recursive call to CD-GBS was along a non-degenerate cross section $\mathscr{P}^t$. From the inductive assumption, this means that CD-GBS computes an $\frac{\varepsilon}{2}$-close labelling of those cross-sections, using the triangle inequality, we know that $\widehat{\mathscr{P}}$ is an ε-close labelling of $\mathscr{P}$.

We note however that there is no guarantee for what a recursive call to CD-GBS does on a degenerate cross section $\widehat{\mathscr{P}}^t$. For this reason, it could be the case that at the end of the loop over $\mathcal{D}^i$, $\widehat{\mathscr{P}}$ is not an ε-close labelling. This can only happen if there is some $t \in C^\alpha_{\mathscr{P}} \cap \mathcal{D}^k$ which is an uncovered critical coordinate.

If t is an uncovered critical coordinate we can rectify the situation. If we find a $z \in B_{\varepsilon/2}$ that is not a critical coordinate, then $\mathscr{P}^z$ is non-degenerate and computing CD-GBS along the cross-section gives us an $\frac{\varepsilon}{2}$-close labelling of $\mathscr{P}^z$. Using the triangle inequality, we see that this in turn removes t from the set of uncovered critical coordinates, and we say that t is "fixed". Thus the final while loop of the algorithm eliminates the set of uncovered critical coordinates so that $\widehat{\mathscr{P}}$ is indeed an ε-close labelling.

It thus remains to show that the final while loop terminates. However, there are at most $|C^\alpha_{\mathscr{P}}|$ uncovered critical coordinates, and over the course of fixing all uncovered critical coordinates, there are at most $|C^\alpha_{\mathscr{P}}|$ bad guesses for $z \in B_{\varepsilon/2}(x)$ where $\mathscr{P}^z$ is degenerate. Therefore the final while loop makes at most $2|C^\alpha_{\mathscr{P}}|$ invocations to CD-GBS along cross-sections. This concludes the proof of correctness for CD-GBS.

Let us bound the total query usage of CD-GBS. For all values of k in the first for loop, we know from Corollary 4.3.2 that since Q_ℓ is a valid lexicographic oracle for $\mathscr{P}$, that the number of uncovered I^k_x will not exceed $2\left(\binom{n+m}{m} + 2n\right)$, and since CD-GBS is called once per uncovered interval, it follows that for each k there at most $2\left(\binom{n+m}{m} + 2n\right)$ recursive calls to CD-GBS. Furthermore, since Q_ℓ is a valid lexicographic oracle for $\mathscr{P}$, it will also never be the case that $\exists i, j \in [n],\ z \in \Delta^m$ such that $dim(\widehat{P}_i) = m$ and $z \in int(\widehat{P}_i)$.

In the worst case, k loops from 1 to $\lceil \log(2/\varepsilon) \rceil$ and makes an extra $2|C^\alpha_{\mathscr{P}}|$ recursive calls to CD-GBS to fix all uncovered critical coordinates. In total if we let $T(m, n, \varepsilon)$ denote the query cost of running CD-GBS on a valid lexicographic oracle, we get the following recursion:

$$T(m, n, \varepsilon) \leq 2|C^\alpha_{\mathscr{P}}| \log\left(\frac{2}{\varepsilon}\right) T\left(m - 1, n, \frac{\varepsilon^2}{85nm^{5/2}}\right) + 2|C^\alpha_{\mathscr{P}}|.$$

In order to make this more amenable, we define $f(m) = \left(\binom{n+m}{m} + 2n\right)$ and use Lemma 4.7 to bound this expression as follows:

$$T(m, n, \varepsilon) \leq 3f(m) \log\left(\frac{2}{\varepsilon}\right) T\left(m-1, n, \frac{\varepsilon^2}{85nm^{5/2}}\right).$$

Furthermore, from the fact that the base case is binary search, we know $T(1, n, \varepsilon) \leq n \log\left(\frac{2}{\varepsilon}\right)$.

To unpack the recursion. Let us define the values $\varepsilon_0 = \varepsilon$ and $\varepsilon_{k+1} = \frac{\varepsilon_k^2}{85n(m-k)^{5/2}}$ for $k = 1, ..., m-1$. With this in hand, we can unroll the recursion to obtain:

$$T(m, n, \varepsilon) \leq \left(3^{m-1} \prod_{i=1}^{m-1} f(i)\right) \left(\prod_{k=1}^{m-1} \log\left(\frac{2}{\varepsilon_k}\right)\right).$$

Since each $\varepsilon_{k+1} < \varepsilon_k$, we can upper bound the right-hand product by bounding each term with ε_{m-1}. If we first solve for this value, we obtain:

$$\begin{aligned}
\varepsilon_{m-1} &= \frac{\varepsilon^{2^{m-1}}}{\prod_{j=1}^{m-1}(85nj^{5/2})^{2^j}}, \\
&\geq \frac{\varepsilon^{2^{m-1}}}{\prod_{j=1}^{m-1}(85nm^{5/2})^{2^j}}, \\
&\geq \left(\frac{\varepsilon}{85nm^{5/2}}\right)^{2^m}.
\end{aligned}$$

In the first inequality we bounded the denominator product in the base by $j \leq m$, as for the second inequality, we evaluated the geometric series in 2 for the exponent to bound the exponent by 2^m. With this in hand we obtain the desired bounds:

$$\begin{aligned}
T(m, n, \varepsilon) &\leq 3^m 2^{m^2} \prod_{i=1}^{m} f(i) \log^m\left(\frac{170nm^{5/2}}{\varepsilon}\right), \\
&\leq \left(\prod_{i=1}^{m}\left(\binom{n+i}{i} + 2n\right)\right) 2^{2m^2} \log^m\left(\frac{170nm^{5/2}}{\varepsilon}\right).
\end{aligned}$$

Finally, For large enough n, every term in $\prod_{i=1}^{m}\left(\binom{n+i}{i} + 2n\right)$ is bounded by $(n+m)^m + 2n$. It follows that this product is $O(n^{m^2})$, and thus for constant m, this constitutes $O(n^{m^2} \log^m\left(\frac{n}{\varepsilon}\right)) = poly(n, \log\left(\frac{1}{\varepsilon}\right))$ queries. □

The previous results show that for constant dimension, m, CD-GBS is query efficient in n and $\frac{1}{\varepsilon}$. In the following section we use this algorithm as a building block to construct a method for computing efficient ε-close labellings when the number of regions, n, is held constant instead.

Algorithm 3 CD-GBS(m, n, ε, Q)

Input: $m \geq 0,\ n, \varepsilon > 0$, query access to oracle $Q : \Delta^m \to [n]$.
Output: $\widehat{\mathscr{P}}$: an ε-close labelling of $\mathscr{P}$.

```
if m = 0 then
    Query Q(0)
else
    P̂^0 ← f_0^{-1}( CD-GBS( m − 1, n, ε²/(85nm^{5/2}), Q ∘ f_0^{-1} ) ), P̂^1 ← Q(e⃗_1).
    for k = 1 to ⌈log(2/ε)⌉ do
        if Number of uncovered I_x^k exceeds 2( C(n+m, m) + 2n ) then
            Halt
        for x ∈ 𝒟^k do
            if I_x^k is uncovered then
                t ← midpoint(I_x)
                P̂^t ← f_t^{-1}( CD-GBS( m − 1, n, ε²/(85(1−t)nm^{5/2}), Q ∘ f_t^{-1} ) )
            Recompute P̂ by taking convex hulls of labels
            if ∃i, j ∈ [n] such that int(P̂_i) ∩ P̂_j ≠ ∅ or P̂ is an ε-close labelling then
                Halt
    while ∃x ∈ [0, 1] an uncovered critical point do
        t ← z for arbitrary z ∈ B_{ε/2}(x)
        P̂^t ← f_t^{-1}( CD-GBS( m − 1, n, ε²/(85(1−t)nm^{5/2}), Q ∘ f_t^{-1} ) )
        Recompute P̂ by taking convex hulls of labels
return P̂
```

4.4 Constant-Region Generalised Binary Search for Q_ℓ

In this section we introduce Constant-Region Generalised Binary Search, (CR-GBS), which as the name suggests, is a query-efficient algorithm for computing ε-close labellings of (m, n)-polytope partitions when n is constant and m and ε are allowed to vary.

The intuition behind the algorithm lies in the fact that if m is much greater than n (it suffices for $m > \binom{n}{2}$), then any vertex of a given P_i cannot lie in the interior of the ambient simplex Δ^m. This is because a vertex in Δ^m must consist of the intersections of at least m half-spaces, all of which cannot arise from adjacencies between different P_i.

Not only do all vertices lie on the boundary of Δ^m, but one can easily show that they are all contained in faces of the boundary of Δ^m that have dimension $O(n^2)$ which is presumed to be constant. The number of such faces in the boundary of Δ^m is thus polynomial in m, and moreover if we could compute 0-close labellings of these faces we could take convex combinations and recover a 0-close labelling of the entire polytope partition.

We will demonstrate that for an appropriate value of ε', if we compute ε'-close labellings of such faces in the boundary, we can recover an ε-close labelling of the entire polytope partition over all of Δ^m by taking convex combinations. CR-GBS computes the necessary ε'-close labellings of lower dimensional faces by using CD-GBS as a subroutine, which as we shall see results in our desired query efficiency for n constant.

We note however, that not all faces in the boundary of Δ^m are axis-aligned, which poses a problem if we are to use CD-GBS as a subroutine. As we show in the following section, this is not an issue since we can translate such simplices into axis-aligned simplices via a simple transformation.

4.4.1 Non-axis-aligned Simplices

So far we have focused on the case where $\Delta^m = \{x \in \mathbb{R}^m \mid \|x\|_1 \leq 1,\, x_i \geq 0\}$. In a straightforward fashion we transform our results to the equivalent simplex $\Lambda^m = \{x \in \mathbb{R}^{m+1} \mid \|x\|_1 = 1,\, x_i \geq 0\}$. To do so, we define the invertible linear map $\phi_m : \Delta^m \to \Lambda^{m+1}$ given by $\phi_m(x_1, ..., x_m) = \left((1 - \sum_{i=1}^m x_i), x_1, ..., x_m\right)$. It is straightforward to see that ϕ_m is $\sqrt{m+1}$-Lipschitz continuous. Via standard Lipschitz continuity arguments we get the following:

Lemma 4.9. *Suppose that $\mathscr{P}$ is an (m, n)-polytope partition of Λ^{m+1}. If $\widehat{\mathscr{P}}$ is an $\frac{\varepsilon}{\sqrt{m+1}}$-close labelling of $\phi_m^{-1}(\mathscr{P})$, then $\phi_m(\widehat{\mathscr{P}})$ is an ε-close labelling of $\mathscr{P}$.*

Proof. Suppose that $x \in \Lambda^{m+1}$ has no label under $\phi_m(\widehat{\mathscr{P}})$. Since $\widehat{\mathscr{P}}$ is an $\frac{\varepsilon}{\sqrt{m+1}}$-close labelling, there must be some $y \in \Delta^m$ with the property that $\|\phi_m^{-1}(x) - y\|_2 < \frac{\varepsilon}{\sqrt{m+1}}$. If we consider $\phi_m^{-1}(x)$, since $\widehat{\mathscr{P}}$ is $\frac{\varepsilon}{\sqrt{m+1}}$-close, there must be some y from an empirical polytope $\widehat{P}_i \subset \Delta^m$ with the property that $\|\phi_m(x) - y\|_2 < \frac{\varepsilon}{\sqrt{m+1}}$. It follows that $\phi_m(y) \in \phi_m(\widehat{\mathscr{P}})$ and by Lipschitz continuity of ϕ_m, $\|x - \phi_m(y)\|_2 < \varepsilon$ as desired. □

4.4.2 Necessary Machinery for CR-GBS

Suppose that $\mathscr{P}$ is an (m, n)-polytope partition with the property that $m > \binom{n}{2}$. Furthermore, let $k = \binom{n}{2}$ and let $\partial_k(\Delta^m)$ denote all k-dimensional faces of Δ^m. For each face F, let $\mathscr{P}_F$ be the restriction of $\mathscr{P}$ to F. If F is axis-aligned (equivalently, if F contains the origin), then it is an isometric embedding of Δ^k in Δ^m, so we let ϕ_F be a canonical isomorphism from F to Δ^k. If F is not axis-aligned, we let ϕ_F be any canonical isomorphism from F to Δ^k as per Section 4.4.1.

As mentioned previously, computing empirical labellings of every face in $\partial_k(\Delta^m)$ via CD-GBS will be enough to compute an empirical labelling for $\mathscr{P}$. The only issue with this strategy however, is that CD-GBS is only guaranteed to return an ε-close labelling if it is

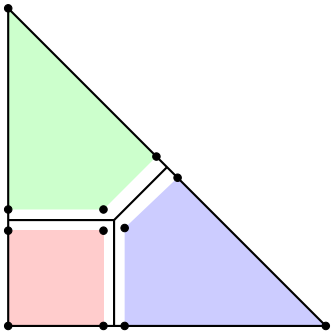

Figure 4.6: γ-Interiors of Polytopes in a Partition.

given access to a valid lexicographic membership oracle for a polytope partition, and for an arbitrary polytope partition, it is not always the case that $\phi_F(\mathscr{P}_F)$ is a (k,n)-polytope partition for all $F \in \partial_k(\Delta^m)$. As an example, consider a polytope partition with an arbitrary $(m-1)$-dimensional polytope P_i contained in $F = \mathscr{P}^0$ (the 0-cross-section of $\mathscr{P}$). Any full-dimensional $P_j \in \mathscr{P}$ must have the property that $0 \notin \pi(relint(P_j))$, hence it still holds that $relint(P_i) \cap relint(P_j) = \emptyset$. However, when restricted to $\mathscr{P}_F$, relative interiors are with respect to $\mathscr{P}^0$, and it can be the case that $relint((P_i)_F) \cap relint((P_j)_F) \neq \emptyset$. For this reason, we slightly refine our notion of polytope partition.

Definition 4.11. *Suppose that $\mathscr{P}$ is an (m,n)-polytope partition such that for all $0 \leq k \leq m$ and $F \in \partial_k(\Delta^m)$, $\phi_F(\mathscr{P}_F)$ is a (k,n)-polytope partition. Then we say that $\mathscr{P}$ is a* proper polytope partition.

For the remainder of this section, we focus on proper polytope partitions. In addition, in order to prove correctness of CR-GBS we define a robust approximation of any $P_i \in \mathscr{P}$.

Definition 4.12. *Suppose that $P \subset \Delta^m$ is a polytope. We define $int_\gamma(P)$ as*

$$int_\gamma(P) = \{x \in P \mid B_\gamma(x) \cap \Delta^m \subset P\}.$$

We call this the γ-interior *of P.*

Intuitively, the γ-interior of P consists of points that are "robustly" within P by a margin of γ relative to the interior of Δ^m, as visualised in Figure 4.6. In Lemma 4.10 we show that $int_\gamma(P)$ is a sub-polytope of P with certain supporting hyperplanes translated towards the interior of P by a margin of γ. We also show that if P_i is an element of an (m,n)-polytope partition where $m > k$, then the vertices of $int_\gamma(P_i)$ also lie in some $F \in \partial_k(\Delta^m)$.

Lemma 4.10. *Suppose that $\mathscr{P}$ is an (m,n)-polytope partition with $m > k = \binom{n}{2}$. For each $P_i \in \mathscr{P}$, and any $\gamma > 0$, $int_\gamma(P_i)$ is a sub-polytope of P_i. Furthermore, each vertex of $int_\gamma(P_i)$ lies in some $F \in \partial_k(\Delta^m)$.*

Proof. Since $P_i \subset \mathbb{R}^m$ is a polytope, it can be expressed as the intersection of finitely many half-spaces: $P_i = \bigcap_{j=1}^{q} H_j$, such that $H_j = \{x \in \mathbb{R}^m \mid a_j \cdot x \geq b_j, \text{ where } a_j \in \mathbb{R}^m,\ \|a_j\|_2 = 1,\ b_j \in \mathbb{R}\}$. As before, each half-space, H_j, can either arise as an adjacency of P_i with the boundary of Δ^m, or as an adjacency of P_i with some other $P_r \in \mathscr{P}$. Let us call the former set of half-spaces A and the latter B. We abuse notation slightly and also let A refer to the sets of indices $j \in [q]$ such that $H_j \in A$ (similarly for B).

For each $H_j \in B$, let $H'_j = \{x \in \mathbb{R}^m \mid a_j \cdot x \geq b_j + \gamma, \text{ where } a_j \in \mathbb{R}^m,\ \|a_j\|_2 = 1,\ b_j \in \mathbb{R}\}$. Clearly $H'_j \subset H_j$, and in fact the boundary hyperplane of H'_j is parallel to that of H_j (and translated by a margin of γ towards the interior of H_j). We now define $C = \left(\bigcap_{j\in A} H_j\right) \cap \left(\bigcap_{j\in B} H'_j\right)$ and we show that $int_\gamma(P_i) = C$, which proves the first part of the lemma.

Suppose that $x \in C$. By virtue of the construction of all H'_j, it must be the case that $B_\gamma(x)$ does not intersect the boundary of any $H_j \in B$. Since all $H_j \in A$ are unchanged in C, we obtain $B_\gamma(x) \cap \Delta^m \subset P_i$, therefore $x \in int_\gamma(P_i)$.

Now suppose that $x \in int_\gamma(P_i)$. Since $int_\gamma(P_i) \subset P_i$, it is clear that $x \in H_j$ for all $H_j \in A$. As for $H_j \in B$, we know that $x \in H_j$ from the fact that $int_\gamma(P_i) \subset P_i$. If $x \notin H'_j$, then $B_\gamma(x) \not\subset H_j$, which in turn implies $B_\gamma(x) \not\subset P$, contradicting our assumption that $x \in int_\gamma(P_i)$. This proves the claim that $C = int_\gamma(P_i)$.

As for the final claim of the lemma, we note that since each $H_j \in A$ arises as an adjacency of P_i with the boundary of Δ^m, it must be the case that $|A| \leq m$. Furthermore, since each $H_j \in B$ arises as an adjacency of two polytopes in $\mathscr{P}$, it follows that $|B| \leq \binom{n}{2} = k < m$. Since at least m half-spaces need to meet in $\mathbb{R}^m$ to make a vertex, it must be the case that any vertex of $C = int_\gamma(P_i)$ lies on some $F \in \partial_k(\Delta^m)$. □

Suppose that $\mathscr{P}$ is a proper (m, n)-polytope partition with $m > k = \binom{n}{2}$. Furthermore, suppose that $P_i \in \mathscr{P}$ is of full affine dimension and consider a vertex, v, of $int_\gamma(P_i)$ which is "robustly" in the interior of P_i by definition. From the previous lemma we know that v lies in some $F \in \partial_k(\Delta^m)$. We now show that due to the margin γ with which v lies within P_i, we can recover a label of v by computing a suitable empirical-labelling of F.

Lemma 4.11. *Suppose that $\mathscr{P}$ is a proper (m, n)-polytope partition with $m > k = \binom{n}{2}$. Furthermore, suppose that $P_i \in \mathscr{P}$ is of full affine dimension and that v is a vertex of $int_\gamma(P_i)$ that lies on some face $F \in \partial_k(\Delta^m)$. It follows that any $\frac{2\gamma}{5}$-close labelling of F that correctly labels the vertices of F gives v the label i. Furthermore, suppose that for all $F \in \partial_k(\Delta^m)$ we compute a $\frac{2\gamma}{5}$-close labelling. By taking convex combinations of these empirical labellings, we get $\tau(\widehat{P}_\perp) \leq \frac{10}{3} n^2 \gamma (m+1)^{3/2}$.*

Proof. If v is a vertex as in the statement of the lemma, it must either be a vertex of the original simplex, or $B_\gamma(v) \cap P_i \cap F$ must contain a r-dimensional ℓ_2 ball of radius γ which

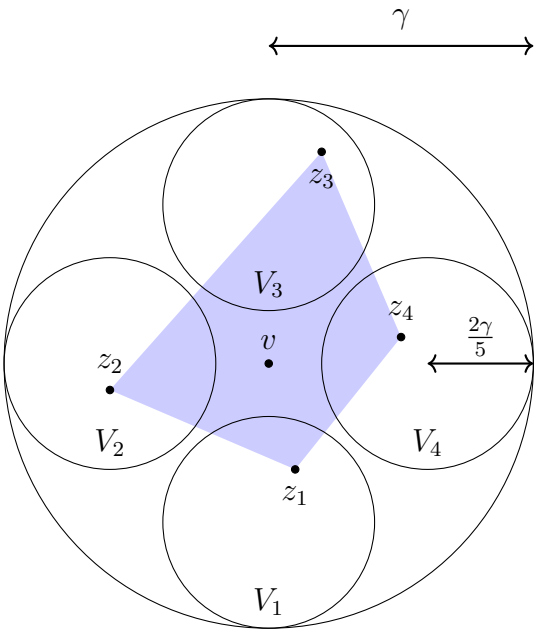

Figure 4.7: Proof of Lemma 4.11.

we call A_2 (where r corresponds to the dimension of the sub-face of F that v lies on, implying $1 \leq r \leq k$). If v is a vertex of the original simplex, then it is correctly labelled by assumption, so we focus on the the latter case.

Let A_1 be any r-dimensional ℓ_1 ball of radius $\frac{3\gamma}{5}$ such that $A_1 \subsetneq A_2$, and denote the corners of A_1 by $x_1, ..., x_s$. For $i = 1, ..., s$, let $V_i = B_{2\gamma/5}(x_i) \cap A_2 \subset F$. We note that $V_i \cap V_j = \emptyset$ for all $i \neq j$.

By the conditions of empirical labellings and the fact that P_i is of full affine dimension, there must exist $z_1, ..., z_s$ such that $z_r \in V_r$ and z_r gets its correct label, i, under Q_ℓ. Furthermore, it is straightforward to see that $v \in Conv(z_1, ..., z_s)$, hence v gets its correct label, i, as visualised in Figure 4.7.

Along with Lemma 4.10, this shows that if for all $F \in \partial(\Delta^m)^k$ we compute $\frac{2\gamma}{5}$-close labellings that correctly label the vertices of F, then we will have correctly labelled all vertices of the polytope $int_\gamma(P_i)$. Consequently, by taking convex combinations of these labellings, the entirety of $int_\gamma(P_i)$ will be labelled correctly for an arbitrary full-dimensional P_i.

For a given full-dimensional $P_i \subset \mathscr{P}$, it is the case that $P_i \backslash int_\gamma(P_i)$ can be expressed as a disjoint union of at most $k \leq n^2$ polytopes of thickness bounded by γ (using the notation from the proof of Lemma 4.10, these polytopes are all of the form $(H_j \setminus H'_j) \cap P_i$, of which there are at most $|B| = k$). For a given P_j that is not full-dimensional, it trivially holds that $\tau(P_j) = 0$ Thus we can use Lemma 4.6 to see that $\tau(\widehat{P}_\perp) = \tau(\cup_i (P_i \setminus int_\gamma(P_i))) \leq \frac{10}{3} n^2 \gamma (m+1)^{3/2}$. □

Corollary 5. *Suppose that $\mathscr{P}$ is a proper (m,n)-polytope partition. Let $\gamma = \frac{3\varepsilon}{40n^2(m+1)^{5/2}}$, and suppose that for all $F \in \partial_k(\Delta^m)$, a $\frac{2\gamma}{5}$-close labelling that correctly labels the vertices of F is computed with Q_ℓ. Taking a convex combination of these empirical labellings results in an ε-close labelling of $\mathscr{P}$.*

Proof. This follows from the fact that $\tau(\widehat{P}_\perp) \leq \frac{10}{3}n^2\gamma(m+1)^{3/2}$ from the previous theorem. We can therefore use Lemma 4.4 and obtain the desired result. □

The previous result gives us precisely what we need to prove the correctness of CR-GBS. In fact, it shows that CR-GBS can use any algorithm as a sub-routine (not just CD-GBS) as long as it computes empirical labellings of polytope partitions along all faces $F \in \partial(\Delta^m)^k$ while correctly labelling the vertices of Δ^m.

4.4.3 Specification of CR-GBS and Query Usage

Terms and Notation: For $F \in \partial_k(\Delta^m)$, we let ϕ_F denote a canonical isomorphism from F to Δ^k as per Section 4.4.2. Furthermore, for each such F, we let $\widehat{\mathscr{P}}_F$ empirical labelling returned by CD-GBS on a given face, F.

Algorithm 4 CR-GBS(m,n,ε,Q)

Input: $m,n,\varepsilon > 0$, query access to oracle $Q : \Delta^m \to [n]$
Output: ε-close labelling of $\mathscr{P}$.

$k \leftarrow \binom{n}{2}$
for $F \in \partial_k(\Delta^m)$ **do**
$\widehat{\mathscr{P}}^F \leftarrow \phi_F^{-1}\left(\text{CD-GBS}\left(k, n, \frac{3\varepsilon}{100n^2\sqrt{k+1}(m+1)^{5/2}}, Q \circ \phi_F^{-1}\right)\right)$.
$\widehat{\mathscr{P}} \leftarrow Conv_F(\widehat{\mathscr{P}}_F)$
return $\widehat{\mathscr{P}}$

Theorem 4.2. *Let $\mathscr{P}$ be a proper (m,n)-polytope partition where n is constant and $m > k = \binom{n}{2}$. CR-GBS computes an ε-close labelling of $\mathscr{P}$ and uses $O\left(m^k \log^k\left(\frac{m}{\varepsilon}\right)\right) = poly(m, \log\left(\frac{1}{\varepsilon}\right))$ queries.*

Proof. The correctness follows from Corollary 5. In the worst case faces are of the form Λ^k, which incur an extra cost of $\sqrt{k+1}$ in the approximation factor of empirical labellings. We use this as a worst case bound.

For simplicity in notation, we define $m_0 = k$, $\varepsilon_0 = \frac{3\varepsilon}{100n^2\sqrt{k+1}(m+1)^{5/2}}$. From Theorem 4.1, the CD-GBS subroutine uses at most $\left(\prod_{i=1}^{m_0}\left(\binom{n+i}{i} + 2n\right)\right) 2^{2m_0^2} \log^{m_0}\left(\frac{170nm_0^{5/2}}{\varepsilon_0}\right)$ queries. Since $k = \binom{n}{2}$ is constant, this expression can be written as $O\left(\log^k\left(\frac{m^{5/2}}{\varepsilon}\right)\right) = O\left(\log^k\left(\frac{m}{\varepsilon}\right)\right)$. Finally, there are $\binom{m}{k}$ possible faces upon which CD-GBS can be called as a subroutine, hence the total query usage is indeed $O\left(m^k \log^k\left(\frac{m}{\varepsilon}\right)\right) = poly(m, \log\left(\frac{1}{\varepsilon}\right))$. □

4.5 Upper Envelope Polytope Partitions

Up until now we have focused completely on the lexicographic query oracle Q_ℓ, creating algorithms CD-GBS and CR-GBS that compute ε-close labellings of (m, n)-polytope partitions when given access to Q_ℓ. If these algorithms are given access to an adversarial oracle Q_A however, they may fail. It suffices to see this for CD-GBS since CR-GBS uses it as a subroutine.

To see why CD-GBS may fail under Q_A we recall that the algorithm recursively computes ε-close labellings of cross-sections $\mathscr{P}^t$ for different values of $t \in [0, 1]$. If ever CD-GBS is called on a degenerate cross-section $\mathscr{P}^t$, it has conditions to either tell that it is being called on a degenerate cross-section (when it notices that there exist $i, j \in [n]$ and $z \in \Delta^m$ such that $z \in int(\widehat{P}_i) \cap \widehat{P}_j$), or in the worst case, prevent it from exceeding its query balance. In both cases however, the algorithm returns a valid empirical labelling, i.e., $\widehat{P} = \{\widehat{P}_i\}_{i=1}^n$ such that $\widehat{P}_i \subseteq P_i$.

When an adversarial oracle is used however, we may see $i, j \in [n]$ and $z \in \Delta^m$ such that $z \in int(\widehat{P}_i) \cap \widehat{P}_j$. Indeed this can occur if $P_i = P_j$ and both are full-dimensional. The natural solution seems to merge P_i and P_j (since the second condition of the definition of polytope partitions tells us that $P_i = P_j$ in this case). The main problem however, is that there is no way of telling when the condition above is an artifice of the adversarial oracle, or simply due to the fact that $\mathscr{P}^t$ is degenerate. If we blindly merge labels, we may in fact be performing an incorrect merge on a degenerate cross-section! This of course may return inconsistent polytope partitions.

Since the key problem is the existence of degenerate cross-sections, we consider a slightly stronger variant of polytope partitions with the key property that cross-sections are never degenerate. Furthermore, this special type of polytope partition is expressive enough for our game theoretic applications, and best of all, it allows us to prove results in the adversarial query oracle model.

Definition 4.13 (Upper Envelope Polytope Partition)**.** *Suppose that $A \in \mathbb{R}^{n \times m}$ is an $n \times m$ real-valued matrix and that $b \in \mathbb{R}^n$. Let $P_i = \{y \in \Delta^m \mid (Ay + b)_i \geq (Ay + b)_j$ for all $j \neq i\}$. We denote the collection $\mathscr{P}(A, b) = \{P_1, \ldots, P_n\}$, as the upper envelope polytope partition (UEPP) arising from (A, b).*

It is straightforward to see that for any (A, b), $\mathscr{P}(A, b)$ is itself an (m, n)-polytope partition. Crucially however, it satisfies more properties than the previous definition of polytope partitions.

Lemma 4.12. *Suppose that A is an $n \times m$ real valued matrix and that $b \in \mathbb{R}^n$. Then $\mathscr{P}(A, b) = \{P_1, \ldots, P_n\}$ has the following properties:*

- *For any $x \in [0, 1)$ let f_x be the canonical affine transformation that maps $(\Delta^m)^x$ to Δ^{m-1}. There exists an $n \times (m-1)$ real matrix A^x and $b^x \in \mathbb{R}^n$ such that $\mathscr{P}(A^x, b^x) = f_x(\mathscr{P}(A, b)^x)$.*

- *$\mathscr{P}(A, b)$ is a proper polytope partition (Definition 4.11).*

- *If $A_{i,\bullet} = A_{j,\bullet}$ and $b_i = b_j$ then $P_i = P_j$. Conversely if P_i is of full affine dimension and $relint(P_i) \cap P_j \neq \emptyset$, then $A_{i,\bullet} = A_{j,\bullet}$ and $b_i = b_j$; consequently, $P_i = P_j$.*

- *Suppose that $a_1, \ldots, a_k \in \mathbb{R}$ are such that $\sum_{i=1}^{k} a_i < 1$ with $k < m$. Let $H = \{(z_1, \ldots, z_m) \in \Delta^m \mid z_i = a_i, i = 1, \ldots, k\}$ where H has affine codimension k. If $x_1, \ldots, x_{m-k} \in \Delta^m$ are affinely independent points of $P_i \cap H$ and in addition, $y \in Conv(x_1, \ldots, x_{m-k})$ belongs to P_j, then P_i and P_j coincide in H.*

Proof. The first bullet point follows from two facts: affine transformations restricted to affine subspaces are themselves affine transformations, and compositions of affine transformations are themselves affine transformations.

To be rigorous, define the affine transformation $g : \mathbb{R}^m \to \mathbb{R}^m$ to be $g(x) = Ax + b$. Let $g' = g \restriction_{(\Delta^m)^x}$ be the restriction of g to the affine subspace $(\Delta^m)^x \subset \mathbb{R}^m$ of codimension 1. As we mentioned before, g' is itself an affine transformation.

Now let us recall that f_x is the canonical affine transformation that maps $(\Delta^m)^x$ to Δ^{m-1}. It is straightforward to see that f_x^{-1} exists (Δ^{m-1} and $(\Delta^m)^x$ are clearly isomorphic) and is itself an affine transformation. Consequently $g' \circ f_x^{-1} : \Delta^{m-1} \to \mathbb{R}^n$ is itself an affine transformation, which can be identified with a matrix A^x and vector b^x such that $(g' \circ f_x^{-1})\, z = A^x z + b^x$. It is straightforward to see that (A^x, b^x) are such that $\mathscr{P}(A^x, b^x) = f_x(\mathscr{P}(A, b)^x)$ as desired.

As for the second bullet point, let $F \in \partial_{m-1}(\Delta^m)$ be an arbitrary face of Δ^m of codimension 1. In addition, we use the notation ϕ_F as before to denote the canonical isomorphism from F to Δ^{m-1}. ϕ_F is itself an affine transformation, hence we can use identical argumentation from before to show that by restricting the original affine functions arising from (A, b) to $F \in \partial_{m-1}(\Delta^m)$, we can find equivalent affine functions that render $\phi_F(\mathscr{P}_F)$ an upper-envelope polytope partition. For arbitrary $0 \leq k \leq m-1$, we can use the previous statement inductively to show that for any $F \in \partial_k(\Delta^m)$, $\mathscr{P}_F$ is a (k, n)-polytope partition. This concludes the proof that $\mathscr{P}$ is a proper polytope partition.

As for the third bullet point, the fact that $A_{i,\bullet} = A_{j,\bullet}$ and $b_i = b_j$ implies $P_i = P_j$ is trivial. Let us focus on the case when P_i is of full affine dimension and $relint(P_i) \cap P_j \neq \emptyset$. For the sake of contradiction, let us suppose that $A_{i,\bullet} = A_{j,\bullet}$ and $b_i \neq b_j$. If this holds, then $(Ay + b)_i \neq (Ay + b)_j$ for all y, which contradicts our assumption that $relint(P_i) \cap P_j \neq \emptyset$. Let us therefore suppose that $A_{i,\bullet} \neq A_{j,\bullet}$. Let H be the set of y

such that $(Ay+b)_i = (Ay+b)_j$. Since $A_{i,\bullet} \neq A_{j,\bullet}$, H has codimension of at least 1. By assumption, there exists a $z \in relint(P_i) \cap P_j$. It must be the case that $z \in H$ as well. However, using the fact that $z \in relint(P_i)$ and that P_i is of full affine dimension, for some $\varepsilon > 0$, the $B_\varepsilon(z) \subsetneq P_i$. However, since $z \in H$, which is of codimension 1, then half of $B_\varepsilon(z)$ must not belong to P_i, which is a contradiction.

The final bullet point follows from putting the first and third bullet points together and inducting on k. The base case follows from the fact that for $w \in [0,1)$, we know that $\mathscr{P}^w$ is itself a scaled upper envelope polytope partition (from the first bullet point). Now suppose that $x_1, ..., x_{m-1} \in P_i$ are affinely independent in $\mathscr{P}(A,b)^w$. Furthermore suppose that $Conv(x_1, ..., x_{m-1})$ contains a point $y \in P_j$. Since the x_i are affinely independent, it follows that P_i^w is full-dimensional in $\mathscr{P}(A,b)^w$, hence we can apply the third bullet point to show that P_i^w and P_j^w coincide in $\mathscr{P}(A,b)^w$ (which is in fact what we desired).

Let us suppose that the claim holds for a given $k-1 < m-1$ and that we are given $a_1, ..., a_k$. From the first bullet point, $\mathscr{P}(A,b)^{a_1}$ is a scaled lower-dimensional upper envelope polytope partition. Let us define $H = \{(z_1, ..., z_m) \in \Delta^m \mid z_i = a_i, i = 1, .., k\}$ and $H_2 = \{(z_1, ..., z_m) \in \Delta^m \mid z_i = a_i, i = 2, .., k\}$. It follows that $\mathscr{P}(A,b) \cap H = \mathscr{P}(A,b)^{a_1} \cap H_2$, and in the later we can use the inductive assumption (since $(m-1) - (k-1) = m-k$) to show that if $x_1, ..., x_{m-k}$ are affinely independent points in $P_i^{a_1} \cap H_2 = P_i \cap H$, and $y \in Conv(x_1, ..., x_{m-k})$ belongs to P_j, then $P_i^{a_1}$ and $P_j^{a_1}$ coincide in H_2, which is the same as saying P_i and P_j coincide in H as desired. □

4.5.1 Adversarial CD-GBS

Suppose that $\mathscr{P}$ is an UEPP. Since it is also a proper (m,n)-polytope partition, it inherits all the properties from before. Along with Lemma 4.12 we have the necessary tools to show that Algorithm 5 is a query efficient way of computing ε-close labellings of $\mathscr{P}$ with an adversarial query oracle. In the specification of CD-GBS, we use identical terms and notation from Algorithm 3.

Theorem 4.3. *If CD-GBS is given access to an adversarial query oracle Q_A of an (m,n)-polytope partition based on a UEPP, it computes an ε-close labelling of $\mathscr{P}$ using at most $\left(\prod_{i=1}^m \left(\binom{n+i}{i} + 2n\right)\right) 2^{2m^2} \log^m\left(\frac{170nm^{5/2}}{\varepsilon}\right)$ membership queries. For constant m this constitutes $O(n^{m^2}\log^m\left(\frac{n}{\varepsilon}\right)) = poly(n, \log\left(\frac{1}{\varepsilon}\right))$ queries.*

Proof. As in the proof of correctness of CD-GBS, we begin by noting that when $m = 1$ the algorithm runs identically to binary search. We thus focus on the case where $m > 1$.

The key observation of the proof of correctness is the following: At any given k in the first for loop there are at most $2|C_{\mathscr{P}}^{\alpha}|$ values of x such that I_x^k is uncovered. Let us consider the empirical polytope $\widehat{\mathscr{P}}$ that has been constructed at the time of the execution

Algorithm 5 Adversarial CD-GBS(m, n, ε, Q_A)

Input: $m \geq 0,\ n, \varepsilon > 0$, query access oracle $Q_A : \Delta^m \to [n]$.

Require: Recursive calls to CD-GBS$\left(m-1, n, \frac{\varepsilon^2}{85(1-x)nm^{5/2}}, Q_A \circ f_x^{-1}\right)$.

Output: ε-close labelling of $\mathscr{P}$.

```
if m = 0 then
    Query Q_A(0)
else
    P̂^0 ← f_0^{-1}(CD-GBS(m − 1, n, ε²/(85nm^{5/2}), Q_A ∘ f_0^{-1}))
    P̂^1 ← Q(e⃗_1).
    for k = 1 to ⌈log(2/ε)⌉ do
        for x ∈ D^k do
            if I_x^k is uncovered then
                t ← midpoint(I_x)
                P̂^t ← f_t^{-1}(CD-GBS(m − 1, n, ε²/(85(1−t)nm^{5/2}), Q_A ∘ f_t^{-1}))
            Recompute P̂ by taking convex hulls of labels
            while ∃i, j ∈ [n], z ∈ Δ^m such that dim(P̂_i) = m and z ∈ int(P̂_i) do
                Merge label i with label j
                Recompute P̂ by taking convex hulls of labels
            if P̂ is an ε-close labelling then
                Break
return P̂
```

of the k-th loop. Due to the fact that we have merged any labels from the execution of the loop at value $k-1$, it follows that for all i, j, $\widehat{P}_i \cap \widehat{P}_j$ is not of full affine dimension. In turn this means that there exists a hyperplane $H_{i,j}$ that separates the interiors of $\widehat{P}_i$ and $\widehat{P}_j$. Furthermore, denote $H_{i,j}^+$ as the halfspace defined by $H_{i,j}$ in which $\widehat{P}_i$ is contained. This means that in turn we can define $\overline{P}_i = \cap_j H_{i,j}^+$ so that $\widehat{P}_i \subset \overline{P}_i$. Furthermore, it is straightforward to see that we can define $\overline{\mathscr{P}} = \{\overline{P}_i\}$ as a valid (m, n)-polytope partition that is consistent with $\widehat{\mathscr{P}}$. Since $\overline{\mathscr{P}}$ is consistent with our current observations from Q_A, we can actually simulate CD-GBS on $\overline{P}_i$ for the first $k-1$ iterations of the algorithm (ordering polytopes accordingly to simulate a lexicographic query oracle). The empirical polytope returned when doing so will in fact be $\widehat{\mathscr{P}}$, and thus we can apply corollary 4.3.2 to tell us that the number of uncovered I_x^k is in fact bounded by $2|C_{\mathscr{P}}^{\alpha}|$.

Returning to our proof of correctness of the algorithm. It is not hard to see that upon termination it is correct if we assume that calling CD-GBS as a subroutine works correctly as per the inductive assumption and crucially the fact that from Lemma 4.12 each cross-section of $\mathscr{P}$ is non-degenerate. Therefore we focus on the query cost of the algorithm. At each $k = 1$ to $\lceil \log(2/\varepsilon) \rceil$, from our previous result there can only be at most $2|C_{\mathscr{P}}^{\alpha}|$ uncovered I_x^k, which are precisely the I_x^k that result in queries. By simple multiplication we thus get that the number of cross-section queries is at most $2\lceil \log(2/\varepsilon) \rceil |C_{\mathscr{P}}^{\alpha}|$, hence

we get the following recursion for bounding the query cost of adversarial CD-GBS:

$$T(m,n,\varepsilon) \leq 2\left(\binom{n+m}{m}+2n\right)\log\left(\frac{2}{\varepsilon}\right)T\left(m-1,n,\frac{\varepsilon^2}{85nm^{5/2}}\right).$$

with base case $T(1,n,\varepsilon) \leq n\log\left(\frac{2}{\varepsilon}\right)$. If we unpack the recursion in the same way as Theorem 4.1, we get the desired result. □

4.5.2 Adversarial CR-GBS

In this section we formalize an adversarial variant of CR-GBS. We note that most of the notation is identical to lexicographic CR-GBS (Algorithm 4).

Algorithm 6 CR-GBS$(m,n,\varepsilon,\mathscr{P})$

Input: $m,n,\varepsilon>0$, query access to Q_A for (m,n)-polytope partition $\mathscr{P}$.
Output: ε-close labelling of $\mathscr{P}$.

```
k ← (n choose 2)
for F ∈ ∂(Δ^m)^k do
    P̂^F ← φ_F^{-1}( CD-GBS( k, n, 3ε / (100 n^2 √(k+1) (m+1)^{5/2}), Q ∘ φ_F^{-1} ) )
P̂ ← Conv_F(P̂_F)
while ∃ i, j ∈ [n], z ∈ Δ^m such that dim(P̂_i) = m and z ∈ int(P̂_i) do
    Merge label i with label j
    Recompute convex hulls of labels
return Q̂
```

Theorem 4.4. *Let $\mathscr{P}$ be an (m,n)-polytope partition where n is constant. Furthermore, let $k=\binom{n}{2}$. CR-GBS computes an ε-close labelling of $\mathscr{P}$ and uses $O\left(m^k\log^k\left(\frac{m}{\varepsilon}\right)\right) = poly(m,\log\left(\frac{1}{\varepsilon}\right))$ queries.*

Proof. As in the proof of Theorem 4.3, we know that there exists a polytope partition $\overline{\mathscr{P}}$ that is consistent with $\widehat{\mathscr{P}}$. Once again, we notice that this invocation of adversarial CR-GBS is identical to running lexicographic CR-GBS on $\overline{\mathscr{P}}$, which would in turn return $\widehat{\mathscr{P}}$ an ε-close labelling of $\overline{\mathscr{P}}$. However, it is straightforward to see from the definition of ε-close labellings that this also makes $\widehat{\mathscr{P}}$ an ε-close labelling of $\mathscr{P}$ as desired. Finally, the query usage of adversarial CR-GBS is identical to lexicographic CR-GBS, hence the rest of the theorem follows. □

4.5.3 Pairwise Comparison Oracles

In this section we take a slight detour to give a short exposition on a natural, more powerful, extension to membership oracles in upper-envelope polytope partitions. The oracles we speak of are *pairwise comparison oracles*, which we will define shortly. At

a high level, for an (m, n)-UEPP, $\mathscr{P}$, a pairwise comparison oracle takes as input two indices, $i, j \in [n]$ and a point $x \in \Delta^m$, and answers whether the affine function defining P_i dominates that of P_j or vice versa.

We introduce algorithms for computing ε-close labelings of UEPP in this setting with a more benign query cost than our previous algorithms that only make use of membership query oracles. Ultimately, we will show in Section 4.7, that pairwise comparison oracles in this setting are a natural analogue of pairwise comparison oracles, $\mathcal{Q}_{PW}$, in games (Chapter 2). This in turn will result in algorithms for computing ε-WSNE via $\mathcal{Q}_{PW}$.

The Oracle Model

Upon close inspection, it is clear that an (m, n)-UEPP is also uniquely defined as a family of real-valued affine functions over Δ^m, which we call $f_1, ..., f_n$. For a given (m, n)-UEPP, $\mathscr{P}$, defined by an $n \times m$ matrix, A, and vector, $b \in \mathbb{R}^n$, these functions are in fact $f_i(y) = (Ay+b)_i$, and we use this notation throughout this section. As a first observation, it is clear that $P_i = \{x \in \Delta^m \mid f_i(x) \geq f_j(x)\ \forall j \in [n]\}$, hence for a given $x \in \Delta^m$, any variation of a membership oracle for $\mathscr{P}$ is simply returning any number of indices $i \in [n]$ of f_i that dominate all other f_j at x.

Definition 4.14 (Pairwise Comparison Oracle)**.** *Suppose that $\mathscr{P}$ is an (m, n)-UEPP parametrised by the affine functions: $f_1, ..., f_n$. We let $Q_{PW} : [n]^2 \times \Delta^m \to \{0, 1\}$ be the* pairwise comparison oracle *for $\mathscr{P}$ and define it as follows:*

$$Q_{PW}(i, j, x) = \begin{cases} 1, & \text{if } f_i(x) \geq f_j(x), \\ 0 & \text{otherwise.} \end{cases}$$

The first thing to notice about the definition is that we haven't made the distinction for when $f_i(x) = f_j(x)$, and it may seem that we are imposing a lexicographic preference. However, it is straightforward to note that if we wish to establish whether $f_i(x) = f_j(x)$, it suffices to make the queries $Q_{PW}(i, j, x)$ and $Q_{PW}(j, i, x)$ and see whether both return 1. In this same vein, we could also consider differing strengths of pairwise comparison oracles that break ties in different ways as we have done with best response oracles throughout this chapter. Nonetheless, the high-level aim of this section is to simply illustrate how the added information from Q_{PW} can be used to compute ε-close labelings, so we focus on this model.

Computing ε-close labellings with Q_{PW}

To simplify our exposition, for any $i, j \in [n]^2$ such that $i \neq j$, we let $H^{i,j} = \{x \in \Delta^m \mid Q_{PW}(i, j, x) = 1\}$. It is straightforward to see that $P_i = \cap_j H^{i,j}$. This hints at the

fact that if we approximately learn all $H^{i,j}$, then we will have have enough information to construct an ε-close labelling of $\mathscr{P}$. To this end, we let $H^{i,j}_\alpha = int_\alpha(H^{i,j}) = \{x \in H^{i,j} \mid \forall y \in H^{j,i},\ \|x-y\|_2 \geq \alpha\}$. In addition we let $F(i,j,\alpha) = \Delta^m \setminus (H^{i,j}_\alpha \cup H^{j,i}_\alpha)$ and $F_\alpha = \bigcup_{i<j} F(i,j,\alpha)$.

Lemma 4.13. *Let $\mathscr{P}$ be a (m,n)-UEPP and $\varepsilon > 0$. If we let $\alpha = \frac{3\varepsilon}{20(m(m+1)^{1/2})(n(n-1))}$ then $\tau(F_\alpha) < \frac{\varepsilon}{4m}$.*

Proof. From the definition of $H^{i,j}_\alpha$ it is clear that $\tau(F(i,j,\alpha)) \leq 2\alpha$ for all values of i,j and α. Let $R = \frac{\varepsilon}{4m}$ and suppose that $x \in F_\alpha$. We will show that $B_R(x)$ cannot be a subset of F_α via a volume argument identical to Lemma 4.6. In what follows we let $k = \binom{m}{2}$ for convenience.

Once again, we let $V(A)$ denote the volume of a set $A \subset \mathbb{R}^m$, and we let $S(m,R)$ denote the volume of the m-dimensional hypersphere of radius R. The first thing to notice is that since $F(\alpha,i,j)$ is precisely the convex region between two hyperplanes at distance 2α from each other, it follows that $V(F(\alpha,i,j) \cap B_R(x)) \leq 2\alpha S(m-1,R)$. This is due to the fact that a majority of the mass of a hypersphere is concentrated around its centre, and hence the volume of the intersection of $F(\alpha,i,j)$ and $B_R(x)$ is maximised when x is equidistant to the supporting planes of $F(\alpha,i,j)$. Furthermore, the volume of this cross-section is bounded by the distance between these supporting hyperplanes multiplied by $S(m-1,R)$, which is at most $2\alpha S(m-1,R)$ as claimed.

Given the bound above, we can take a union bound and see that $V(F_\alpha \cap B_R(x)) \leq 2k\alpha S(m-1,R)$ Hence it suffices to show that $2k\alpha S(m-1,R) < S(m,R)$ to show that $B_R(x)$ cannot be a subset of F_α. As in Lemma 4.6, we recall the following identity:

$$\frac{S(m,R)}{S(m-1,R)} \geq 0.6R(m+1)^{-1/2}.$$

From which the fact that $B_R(x)$ is not a subset of F_α follows from our choice of parameters. □

In Algorithm 7 we provide the specification for Approx-Halfspace, which for a given $\alpha > 0$ and $i,j \in [n]$ such that $i < j$ computes $\widehat{H}^{i,j}$ and $\widehat{H}^{j,i}$ as approximations to $H^{i,j}$ and $H^{j,i}$ respectively via binary search along the edges of Δ^m.

Lemma 4.14. *Suppose that Approx-Halfspace is given access to a pairwise comparison oracle Q_{PW} for a UEPP, $\mathscr{P}$. For a given $\alpha > 0$ and $i,j \in [n]$ such that $i < j$, and $f_i \neq f_j$, Approx-Halfspace(α,i,j) computes $\widehat{H}^{i,j}$ and $\widehat{H}^{j,i}$ such that $\widehat{H}^{i,j}_\alpha \subseteq H^{i,j}$ and $\widehat{H}^{j,i}_\alpha \subseteq H^{j,i}$. On the other hand, if $f_i = f_j$, then $\widehat{H}^{i,j} = \Delta^m$ and $\widehat{H}^{j,i} = \emptyset$.*

Proof. First we notice that $Q_{PW}(i,j,e_r)$ gives the same answer for all $r \in [m]$ if and only if $f_i(x) \geq f_j(x)$ for all $x \in \Delta^m$. In either case, this means that $\widehat{H}^{i,j} = \Delta^m$ and $\widehat{H}^{j,i} = \emptyset$. When $f_i(x) > f_j(x)$ for all $x \in \Delta^m$, Then by definition $H_\alpha^{i,j} = \Delta^m$ and $H_\alpha^{j,i} = \emptyset$ and the claim holds. The second case where $f_i = f_j$ is also vacuously covered in the lemma. We therefore focus on the scenario where not all $Q_{PW}(i,j,e_r)$ give the same answer.

As per the notation of Algorithm 7, we let V_i be the vertices of Δ^m such that $Q_{PW}(i,j,e_r) = 1$ and V_j be all other vertices. Suppose that $e_r \in V_i$, $e_s \in V_j$, and that y_{rs} is the edge of Δ^m between these two vertices. If $H_\alpha^{i,j} \neq \emptyset$, we let z_i be the unique $x \in H_\alpha^{i,j} \cap y_{rs}$ that is at a maximal distance from e_r, otherwise we let $z_i = e_r$. Likewise, if $H_\alpha^{j,i} \neq \emptyset$, we let z_j be the unique $x \in H_\alpha^{j,i} \cap y_{rs}$ that is at a maximal distance from e_s, otherwise we let $z_j = e_s$.

Our first observation is that $\|z_j - z_i\|_2 \geq \alpha$. $\|z_j - z_i\|_2$ is in fact minimised when y_{rs} is perpendicular to the separating hyperplane between $H^{i,j}$ and $H^{j,i}$, in which case $\|z_j - z_i\|_2 = \alpha$, hence the claim holds. Given this observation, it follows that $z_i \in Conv(\{e_r, x_{r,s}^i\})$ and $z_j \in Conv(\{e_s, x_{r,s}^j\})$, where $x_{r,s}^i$ and $x_{r,s}^i$ are the limiting points found via binary search along y_{rs} in Approx-Halfspace. The reason for this is that our binary search parameter is in fact α, which as we saw is the minimal distance between z_i and z_j. Finally, it is a straightforward observation that $H_\alpha^{i,j}$ is in fact contained in the convex combination of all such z_i (for every edge between V_i and V_j) and all vertices in V_i. It thus follows that $\widehat{H}_\alpha^{i,j} \subseteq \widehat{H}_\alpha^{i,j}$ as desired.

□

With Approx-Halfspace in hand, we can use it as a subroutine in our end algorithm for computing ε-close labelings of UEPP via Q_{PW}. We call our algorithm for this task Comparison-Polytope(ε), with specification as per Algorithm 8.

Theorem 4.5. *Comparison-Polytope(ε) correctly returns an ε-close labelling of Δ^m. Furthermore, it uses $O(n^2m^2\log(\frac{nm}{\varepsilon}))$ calls to Q_{PW}*

Proof. From Lemma 4.14 we know that each $\widehat{H}^{i,j} \subseteq H_\alpha^{i,j} \subseteq H^{i,j}$. Furthermore, we also know that $P_i = \cap_j H^{i,j}$, hence it follows that $\widehat{P}_i = \cap_j \widehat{H}^{i,j} \subseteq P_i$. Furthermore, it also follows that $\widehat{P}_\perp = \Delta^m \setminus (\cup_i \widehat{P}_i) \subseteq F_\alpha$, hence from Lemma 4.13 it follows that $\tau(\widehat{P}_\perp) < \frac{\varepsilon}{4m}$, which from Lemma 4.3, implies that $\widehat{\mathscr{P}}$ is indeed an ε-close labelling.

As for query usage, we once again let $k = \binom{n}{2}$, and note that we call Approx-Halfspace precisely k times. Each time, we perform binary search along at most $\left(\frac{m}{2}\right)^2$ edges of Δ^m up to a resolution of $\alpha = \frac{3\varepsilon}{10(m(m+1)^{1/2})(n(n-1))}$. It follows that the query usage of Comparison-Polytope is at most $k\left(\frac{m}{2}\right)^2 \log\left(\frac{1}{\alpha}\right) = O(n^2m^2\log\left(\frac{nm}{\varepsilon}\right))$ as desired. □

Algorithm 7 Approx-Halfspace(α, i, j)

Require: $\alpha > 0$, $i, j \in [n]$ such that $i < j$
  $V_i, V_j = \emptyset$
  for $r \in [m]$ **do**
    Query $Q_{PW}(i, j, e_r)$
    if $Q_{PW}(i, j, e_r) = 1$ **then**
      $V_i \leftarrow V_i \cup \{e_r\}$
    else
      $V_j \leftarrow V_j \cup \{e_r\}$
  for $r, s \in [m]^2$ such that $r < s$ **do**
    if $Q_{PW}(i, j, e_r) \neq Q_{PW}(i, j, e_s)$ **then**
      $L \leftarrow e_r$
      $R \leftarrow e_s$
      while $\|L - R\|_2 \geq \alpha$ **do**
        $C \leftarrow (L + R)/2$
        if $Q_{PW}(i, j, C) = Q_{PW}(i, j, R)$ **then**
          $R \leftarrow C$
        else
          $C \leftarrow C$
      if $Q_{PW}(i, j, e_r) = 1$ **then**
        $x^i_{r,s} \leftarrow L$
        $x^j_{r,s} \leftarrow R$
      else
        $x^i_{r,s} \leftarrow R$
        $x^j_{r,s} \leftarrow L$
  $\widehat{H}^{i,j} \leftarrow Conv(\{x^i_{r,s}\}, V_i)$
  $\widehat{H}^{j,i} \leftarrow Conv(\{x^j_{r,s}\}, V_j)$
  return $\widehat{H}^{i,j}, \widehat{H}^{j,i}$

Algorithm 8 Comparison-Polytope(ε)

Require: $\varepsilon > 0$
  $\alpha \leftarrow \frac{3\varepsilon}{10(m(m+1)^{1/2})(n(n-1))}$
  for $r, s \in [m]^2$ such that $r < s$ **do**
    $\widehat{H}^{r,s}, \widehat{H}^{s,r} \leftarrow$ Approx-Halfspace(α, r, s)
  for $i = 1, ..., n$ **do**
    $\widehat{P}_i = \cap_j \widehat{H}^{i,j}$
  return $\widehat{\mathscr{P}}$

Unlike CD-GBS and CR-GBS, the most striking feature of Comparison-Polytope is that it has a query complexity that is polynomial in all relevant parameters: m, n, and $\log\left(\frac{1}{\varepsilon}\right)$. As we will see in Section 4.8, this will in turn correspond to algorithms for computing ε-WSNE in $m \times n$ bimatrix games with a query cost that is polynomial in m, n, and $\log\left(\frac{1}{\varepsilon}\right)$, where m and n are the number of strategies of the row and column player respectively.

4.6 Bimatrix Games and Lipschitz Continuity of Utility Functions

Now that we have established query-efficient algorithms for learning ε-close labellings of polytope partitions, we turn our attention to game theory to prove the connection between learning these labellings and computing approximate well-supported Nash equilibria in bimatrix games.

Suppose that $G = (A, B)$ is an $m \times n$ bi-matrix game where $A, B \in [0,1]^{m \times n}$ are the row player and column player payoff matrices respectively with payoffs normalised to $[0, 1]$. We wish to identify an ε-well-supported Nash equilibrium (ε-WSNE) using only limited information on G. The set of row player pure strategies is $[m] = \{1, \ldots, m\}$ and similarly that of the column player pure strategies is $[n] = \{1, \ldots, n\}$. Furthermore, the set of all row player mixed strategies can be associated with the axis-aligned $(m-1)$-simplex: $\Delta^{m-1} = \{\vec{x} \in \mathbb{R}^{m-1} | \sum_{i=1}^{m-1} x_i \leq 1 \text{ and } x_i \geq 0\}$. Similarly, column player mixed strategies are identified with Δ^{n-1}.

Definition 4.15 (Bimatrix Utility Functions)**.** *Suppose that $u \in \Delta^{m-1}$, and $v \in \Delta^{n-1}$ are row and column player mixed strategies. Let $u' = (1 - \sum u_i, u_1, ..., u_{n-1})$ and $v' = (1 - \sum v_i, v_1, ..., v_{n-1})$. Then for strategy profile (u, v), row player utility is $U_r(u, v) = u'^T A v'$ and column player utility is $U_c(u, v) = u'^T B v'$.*

It will also be useful to have shorthand for the following functions: $U_r^i(y) = U_r(e_i, y)$ as the row player utility for playing pure strategy i, and $E_R(y) = \max_{i \in [m]} U_r^i(y)$ as the maximal utility the row player can achieve against mixed strategies. In an identical fashion we can define U_c^j and E_C as the column player utility in playing strategy j and the maximal column player utility. With this notation in hand, we can define the best response oracles algorithms will have access to when computing approximate Nash equilibria.

Definition 4.16 (Best Response Query Oracles)**.** *Any bimatrix game has the following best response query oracles:*

- *Strong query oracles: for the column player,* $BR_s^C(u) = \{j \in [n] \mid U_c^j(u) = E_C(u)\}$ *and for the row player,* $BR_s^R(v) = \{i \in [m] \mid U_r^i(v) = E_R(v)\}$.
- *Lexicographic query oracles: for the column player,* $BR_\ell^C(u) = \min_{j\in[n]} j \in BR_s^C(u)$ *and for the row player,* $BR_\ell^R(v) = \min_{i\in[m]} i \in B_s^R(v)$.
- *Adversarial query oracles: for the column player, any function* BR_A^C *such that* $BR_A^C(u) \in BR_s^C$ *and for the row player, any function such that* $BR_A^R(v) \in BR_s^R(v)$.

From the perspective of the row (column) player, when faced against a column (row) player strategy v (u), these best response oracles provide information regarding optimal actions of the row (column) player. The full best response oracle returns all optimal actions, the lexicographic oracle breaks ties between optimal actions in a consistent way, and finally, an adversarial oracle simply returns any given optimal action in case there are ties.

For a given mixed strategy, $u \in \Delta^{m-1}$, we say the support of u is the set of pure strategies that are played in u with non-zero probability. It will be useful to formulate this as a function in order to define Nash equilibria combinatorially in the following section.

Definition 4.17 (Support Functions)**.** *Let* $S^R : \Delta^{m-1} \to \mathcal{P}([m])$ *be the function which returns the support of a row player mixed strategy. Similarly let* $S^C : \Delta^n \to (\mathcal{P}[n])$ *return the support of column player mixed strategies.*

We recall the definition of a Nash equilibrium from Chapter 2: a pair (u, v) is a *Nash Equilibrium (NE)* if for all $u' \in \Delta^{m-1}$ and $v' \in \Delta^{n-1}$: $U_r(u, v) \geq U_r(u', v)$ and $U_c(u, v) \geq U_c(u, v')$. Furthermore, (u, v) is a NE if and only if $S_R(u) \subseteq BR_R^s(v)$ and $S_C(v) \subseteq BR_C^s(u)$. I.e. u is supported by best responses to v and vice versa. When using best response queries only, one does not have access to utility values (as emphasised in the first definition), however this second equivalent definition of Nash equilibria can be verified by using best response oracles and supports alone. In addition, we also note that for utility queries the complexity of an exact Nash equilibrium is finite: we can exhaustively query the game. On the other hand, the following theorem shows this is not the case for best response queries. This further motivates studying the computation of approximate equilibria with best response query oracles.

Theorem 4.6. *The query complexity of computing an* ε*-WSNE in bimatrix games with strong best response queries is* $\Omega(\log(\frac{1}{\varepsilon}))$*. Consequently, the query complexity of computing exact NE with strong best response queries is unbounded.*

Proof. Let us consider a family of bimatrix games $G_{x,y} = (A_x, B_y)$, parametrised by $x, y \in (0,1)$ and with payoff matrices:

$$A_x = \begin{pmatrix} x & x \\ 0 & 1 \end{pmatrix}, \text{ and } \; B_y = \begin{pmatrix} 0 & y \\ 1 & y \end{pmatrix}.$$

It is clear that for any $x, y \in (0,1)$, $G_{x,y}$ has no pure equilibria, but its unique NE is the mixture: row strategy $u = (1-y, y)$ and column strategy $v = (1-x, x)$. Furthermore, the set of ε-WSNE of this game correspond precisely to $B_\varepsilon(x) \times B_\varepsilon(y) \subseteq [0,1]^2$.

Before continuing, suppose that $x \in (0,1)$ is fixed and that $a_1, ..., a_k \in [0,1]$. Given x, we can define A_x as above, and this gives rise to the row player best response oracle BR^s_R which takes as input the probability a column player places on the rightmost column. No matter what the choice of x, it is always the case that $BR^s_R(0) = \{1\}$ (the top row) and $BR^s_R(1) = \{2\}$ (the bottom row). With this in hand, we define two quantities $\alpha_1, \alpha_2 \in [0,1]$. If $BR^s_R(a_i) = \{2\}$ for all a_i, we let $\alpha_1 = 0$, otherwise we let $\alpha_1 = \max\{a_i \mid BR^s_R(a_i) = \{1\}\}$. We also define α_2 in a similar fashion. If $BR^s_R(a_i) = \{1\}$ for all a_i, we let $\alpha_2 = 1$, otherwise we let $\alpha_2 = \min\{a_i \mid BR^s_R(a_i) = \{2\}\}$.

We claim that no matter the choice of $a_1, ..., a_k$, there is always a choice of x such that $\alpha_2 - \alpha_1 \geq 2^{-k}$, and we prove this via induction. For the base case, we consider a single point $a_1 \in [0,1]$. If $a_1 \leq 1/2$, then any $x > a_1$ leads to $BR^s_R(a_1) = \{1\}$ which suffices. On the other hand, if $a_1 > 1/2$, any $x < a_1$ leads to $BR^s_R(a_1) = \{2\}$ which suffices.

Now let us suppose that the claim holds for the points $a_1, ..., a_k \in [0,1]$ with values α_1 and α_2 such that $\alpha_2 - \alpha_1 \geq 2^{-k}$. Let us consider a new point a_{k+1}. If $a_{k+1} \in [0, \alpha_1] \cup [\alpha_2, 1]$, then $BR^s_R(a_{k+1})$ is forcibly determined, and the resulting α'_1 and α'_2 are unchanged and the claim still holds. On the other hand, let us suppose that $a_{k+1} \in (\alpha_1, \alpha_2)$ and let $\zeta = \frac{1}{2}(\alpha_1 + \alpha_2)$. If $a_{k+1} \leq \zeta$, any $x \in [\zeta, \alpha_2]$ leads to $BR^s_R(a_{k+1}) = \{1\}$ which suffices. On the other hand, if $a_{k+1} > \zeta$, any $x \in [\alpha_1, \zeta)$ leads to $BR^s_R(a_{k+1}) = \{2\}$ which suffices. This proves the claim.

Now let us suppose that an algorithm, A computes ε-WSNE for the family of games $G_{x,y}$ and that A makes less than $k = \log(\frac{1}{\varepsilon}) - 2$ queries to BR^s_R and BR^c_R. Forcibly it must be the case that less than k queries are made to BR^s_R, and from our claim above, we know that no matter the queries made, there is an adversarial choice of $x \in [0,1]$ such that $\alpha_2 - \alpha_1 > 2^{-k} = 4\varepsilon$. Furthermore, any $x \in [\alpha_1, \alpha_2]$ is consistent with the answers to BR^s_R. This however shows that A cannot possibly be a correct algorithm, for if it is to compute an ε-WSNE, it must return a value $z \in [\alpha_1, \alpha_2]$ that is at a distance ε from the parameter $x \in [0,1]$ chosen by an adversary. Since $\alpha_2 - \alpha_1 = 4\varepsilon$ this is impossible.

□

4.6.1 Algebraic Properties of Utility Functions

The definition of a ε-WSNE needs approximate best responses, yet we only have access to the best response oracle in our model. In order to resolve this, we delve into the algebraic properties of utility functions of both the column and row player.

Lemma 4.15. *If the domains of U_r^i and U_c^j are endowed with the ℓ_2 norm, then the functions are λ_R and λ_C Lipschitz continuous respectively, for some $0 \leq \lambda_R \leq \sqrt{n-1}$ and $0 \leq \lambda_C \leq \sqrt{m-1}$. If the domains are endowed with the ℓ_1 norm, then both functions are 1-Lipschitz continuous.*

Proof. We focus on U_r^i, the case for the column player is identical. Let $c = [A^T]_i = (a_1, ..., a_n)$ be the i-th row vector of the row player's payoff matrix, and suppose that $v = (v_1, ..., v_{n-1}) \in \Delta^{n-1}$ is a column player mixed strategy, where $v_0 = 1 - \sum_{i=1}^{n-1} v_i$ is implicit. Let $z = (z_i)_{i=1}^{n-1}$ with $z_i = (a_i - a_0)$ for $i = 1, ..., n-1$. Then it is clear that $U_r^i(v) = a_0 + \sum_{i=1}^{n-1} z_i \cdot v_i$. This function is linear, and trivially $\|z\|_2$-Lipschitz continuous. Since the game is normalised, $\|z\|_2 \leq \|\vec{1}\|_2 = \sqrt{n-1}$.

As for the second part of the claim, the domain of U_r^i can be equivalently represented as $\Lambda^n = \{x \in \mathbb{R}^n \mid \|x\|_1 = 1, x_i \geq 0\}$ by using the invertible linear map $\phi_{n-1} : \Delta^{n-1} \to \Lambda^n$ given by $\phi_{n-1}(x_1, ..., x_{n-1}) = \big(x_1, ..., x_{n-1}, \big(1 - \sum_{i=1}^{n-1} x_i\big)\big)$. This space can be endowed with total variation distance as a metric, which for two distributions, $x, y \in \Lambda^n$ is defined as $TV(x, y) = \max_{s \subseteq n} |\mathbb{P}_x(S) - \mathbb{P}_y(S)| = \frac{1}{2}\|x - y\|_1$. Since utilities are bounded to be in the interval $[0, 1]$, it follows that U_r^i is 1-Lipschitz as a function with domain Λ^n in the total variation metric.

Now suppose that $x, y \in \Delta^{n-1}$. We wish to show that $TV(\phi_{n-1}(x), \phi_{n-1}(y)) \leq \|x - y\|_1$. To see this, let $x_n = 1 - \sum_{i=1}^{n-1} x_i$ and $y_n = \sum_{i=1}^{n-1} y_i$. Then $\|\phi_{n-1}(x) - \phi_{n-1}(y)\|_1 = \|x - y\|_1 + |x_n - y_n|$. From this we see that $|x_n - y_n| = |\sum_{i=1}^{n-1}(y_i - x_i)| \leq \|x - y\|_1$, which in turn implies $\|\phi_{n-1}(x) - \phi_{n-1}(y)\|_1 \leq 2\|x - y\|_1$. Dividing the expresion by 2 and applying the fact that $TV(x, y) = \frac{1}{2}\|x - y\|_1$ proves our desired inequality. 1-Lipschitz continuity in the ℓ_1 norm for domain Δ^{n-1} follows immediately. □

Corollary 6. *Since E_C and E_R are defined as a pointwise maximum, it follows that they are also Lipschitz continuous with constant $\lambda_C \leq \sqrt{m-1}$ and $\lambda_R \leq \sqrt{n-1}$ in the ℓ_2 norm.*

With bounded Lipschitz continuity, we have guarantees on how much utilities can deviate between "close" mixed strategy profiles. This has interesting implications even for the best response query oracle, for this means that if u and u' are close in the ℓ_2 norm with $c_i \in BR_s^C(u)$, $c_j \in BR_s^C(u')$ and $c_i \neq c_j$, then we can say that c_i and c_j are both approximate best responses in the vicinity of u and u'.

This idea is formalised in the following Lemma which establishes how to obtain information regarding approximate best responses using only the best response oracle and "nearby" queries. With some thought one can see that this in general does not reveal *all* approximate best response information. For example, if a strategy were strictly dominated, a best response oracle would never see it, and hence never be able to tell if it was an approximate best response.

Lemma 4.16. *Fix $\varepsilon > 0$ and let $\delta_C = \frac{\varepsilon}{2\sqrt{m-1}}$. Suppose that $u \in \Delta^{m-1}$ is a row player mixed strategy with $c_j \in BR_s^C(u)$. For any u' such that $\|u - u'\|_2 \leq \delta_C$, if $c_i \in BR_s^C(u')$, then $|U_c^i(u) - U_c^j(u)| \leq \varepsilon$. In other words, c_i is an ε-best response to u. Similarly, let $\delta_R = \frac{\varepsilon}{2\sqrt{n-1}}$. Suppose that $v \in \Delta^{n-1}$ is a column player mixed strategy with $r_j \in BR_s^R(u)$. For any v' such that $\|v - v'\|_2 \leq \delta_R$, if $r_i \in BR_s^R(v')$, then $|U_r^i(v) - U_r^j(v)| \leq \varepsilon$. In other words, r_i is an ε-best response to v.*

Proof. Suppose that u' is such that $\|u - u'\| \leq \delta_C$ and $c_i \in BR_s^C(u')$. By definition, $E_c(u') = U_c^i(u') \geq U_c^j(u')$ and by Lemma 4.15, $|U_j^c(u) - U_j^c(u')| \leq \lambda_C \|u - u'\|_2$, and $\|U_i^c(u) - U_i^c(u')| \leq \lambda_C \|u - u'\|_2$. With these expressions we obtain the following inequalities:

$$|E_c(u) - U_c^i(u)| \leq 2\lambda_C \|u - u'\|_2 \leq 2\lambda_C \frac{\varepsilon}{2\lambda_C} = \varepsilon.$$

The proof of the second half of the lemma is identical. □

4.7 Nash's Theorem with Discrete Approximations

We are now in a position to prove the intimate connection between computing ε-close labellings of upper envelope polytope partitions and computing ε-WSNE for bimatrix games using best response queries.

Definition 4.18 (Best Response Sets)**.** *Let $G = (A, B)$ be a bimatrix game. We define column best response sets as the collection of $C_i = \{x \in \Delta^{m-1} \mid BR_C^s(x) = c_i\}$. Similarly we define row player best response sets as the collection of $R_j = \{y \in \Delta^{n-1} \mid BR_R^s(y) = r_j\}$. We denote the collections by $\mathscr{C} = \{C_i\}_{i=1}^n$ and $\mathscr{R} = \{R_j\}_{j=1}^m$.*

Since utilities are affine functions, it is immediately clear that $\mathscr{C}$ and $\mathscr{R}$ are upper envelope polytope partitions. Now the best response oracles play the same role as membership oracles, Q, from before. Since adversarial oracles are the weakest of the three membership oracles (in the sense that they are a valid lexicographic oracle and they can be simulated with access to a strong oracle), we focus on using adversarial best response oracles. Furthermore, with our language of empirical labellings we can now define the following important concept, but first we clarify some notation: $d(x, S)$ denotes the infimum distance of a point, x to a set S.

Definition 4.19 (Voronoi Best Response Sets)**.** *Suppose that* $\widehat{\mathscr{C}} = \{\widehat{C}_i\}$ *and* $\widehat{\mathscr{R}} = \{\widehat{R}_j\}$ *are empirical labellings of* $\mathscr{C}$ *and* $\mathscr{R}$ *as in Definition 4.6. The* Voronoi Best Response Sets *of the row and column player are* $VR_j = \{y \in \Delta^{n-1} \mid \operatorname{argmin}_j d(y, \widehat{R}_j) = r_j\}$ *and* $VC_i = \{x \in \Delta^{m-1} \mid \operatorname{argmin}_i d(x, \widehat{C}_i) = c_i\}$*, defined for any* $j \in [m]$ *and* $i \in [n]$*. Furthermore, we let* $V^R(v) = \{i \mid VR_i \ni v\}$ *and* $V^C(u) = \{j \mid VC_j \ni u\}$ *be the row and column player* Voronoi Best Responses*.*

Lemma 4.17. *Voronoi best response sets partition* Δ^{m-1} *and* Δ^{n-1} *into closed connected regions with non-empty interior and piecewise linear boundaries.*

Proof. Without loss of generality, let us focus on a given column player Voronoi best response set; i.e. some $VC_i \subset \Delta^{m-1}$ that is non-empty and arises from the empirical labelling $\widehat{\mathscr{C}} = \{\widehat{C}_i\}$ of column-player best responses. First we note that $\widehat{C}_i \subset VC_i$ by definition, and the former is a convex, closed, and connected polytope of Δ^{m-1}. Therefore it remains to show that if we pick an arbitrary $x \in VC_i \setminus \widehat{C}_i$ it is connected to C_i. To do so, suppose that $p \subset \Delta^{m-1}$ is the unique shortest path from x to C_i. It is clear that all points along p must also lie in VC_i, therefore the claim holds.

As for closedness, note that if $\{x_n\}$ is a sequence in VC_i that converges to some $x \in \Delta^{m-1}$ then x must also be in VC_i. This follows from the continuity of Euclidian distance for $\Delta^{m-1} \subset \mathbb{R}^m$. Now suppose that x is a limit point of VC_i, then we can construct a sequence as above, and thus x must also be in VC_i, rendering the set closed.

Finally, the piecewise linear boundary arises from the fact that $\widehat{C}_i$ is itself a closed convex polytope which has a piecewise linear boundary. Decision boundaries between different $\widehat{C}_i$ and $\widehat{C}_j$ are composed of nearest neighbour decision boundaries between piecewise linear boundaries, which in turn results in piecewise linear decision boundaries between VC_i and VC_j. □

Although the previous lemma proves that Voronoi best response sets partition Δ^{m-1} and Δ^{n-1} into closed connected regions with non-empty interior and piecewise linear boundaries, they need not be convex. This ends up not being an issue for our subsequent results. The reason we deal with these objects however is due to the following Lemma. We recall that $\lambda_R \leq \sqrt{m-1}$ and $\lambda_C \leq \sqrt{n-1}$ are the relevant Lipschitz continuity constants for row player and column player expected utility functions. The following is a straightforward consequence of Lemma 4.16.

Lemma 4.18. *Suppose that* $\widehat{\mathscr{C}}$ *is a* $\frac{\varepsilon}{2\sqrt{m-1}}$*-close labelling and* $\widehat{\mathscr{R}}$ *is a* $\frac{\varepsilon}{2\sqrt{n-1}}$*-close labelling. Then Voronoi best responses are* ε *best-responses in* G

We recall that the combinatorial formulation of Nash's theorem implies that with full information of best response sets in all of Δ^{m-1} and Δ^{n-1}, one is able to compute and verify an exact Nash equilibrium. Best responses only partially recover this information in convex patches of Δ^{m-1} and Δ^{n-1}. Voronoi best response sets however allow us to take this partial information and extend it to approximate best response information *across the entire domains* Δ^{m-1} *and* Δ^{n-1} (Voronoi best response sets cover Δ^{m-1} and Δ^{n-1} after all). This hints at the fact that Voronoi best response sets hold enough information to compute ε-WSNE. In fact we can prove this in the same way as Nash's theorem: via Kakutani's fixed point theorem. In order to do so, we define a Voronoi best response correspondence (which as we have shown before is an approximate best response correspondence), and show that it satisfies the properties of Kakutani's fixed point theorem. The guaranteed fixed point of this correspondence will in turn be an ε-WSNE. In what follows we clarify that we use the notation $\mathcal{P}(X)$ to denote the power set of an arbitrary set X.

Definition 4.20 (Voronoi Approximate Best Response Correspondence)**.** *For a given mixed strategy profile of both the row and column player,* $(u, v) \in \Delta^{m-1} \times \Delta^{n-1}$*, we define* $B^*(u, v)$ *to be the set of all possible mixtures over Voronoi best response profiles both players may have to the other player's strategy.* $B^* : \Delta^{m-1} \times \Delta^{n-1} \to \mathcal{P}(\Delta^{m-1} \times \Delta^{n-1})$ *is defined as follows:*

$$B^*(u, v) = \big(conv(V^R(v)), conv(V^C(u))\big) \subseteq \Delta^{m-1} \times \Delta^{n-1}.$$

Theorem 4.7 (Kakutani's Fixed Point Theorem [40])**.** *Let* A *be a non-empty subset of a finite-dimensional Euclidian space and* $f : A \to \mathcal{P}(A)$ *be a set-valued function satisfying the following conditions:*

- A *is a compact and convex set.*
- $f(x)$ *is non-empty for all* $x \in A$*.*
- $f(x)$ *is a convex-valued correspondence: for all* $x \in A$*,* $f(x)$ *is a convex set.*
- $f(x)$ *has a closed graph: that is, if* $\{x_n, y_n\} \to \{x, y\}$ *with* $y_n \in f(x_n)$ *for all* n*, then* $y \in f(x)$*.*

Then f *has a fixed point, that is there exists some* $x \in A$ *such that* $x \in f(x)$*.*

Theorem 4.8. B^* *satisfies all the conditions of Kakutani's fixed point Theorem, and hence there exists a strategy profile* (u^*, v^*) *such that* $(u^*, v^*) \in B^*(u^*, v^*)$*. In particular, if the Voronoi best responses for* B^* *arise from* $\widehat{\mathscr{C}}$*, a* $\frac{\varepsilon}{2\sqrt{m-1}}$*-close labelling and* $\widehat{\mathscr{R}}$*, a* $\frac{\varepsilon}{2\sqrt{n-1}}$*-close labelling, then this in turn implies that* (u^*, v^*) *is an* ε*-WSNE of* G*.*

Proof. We need to prove the following conditions for Kakutani's fixed point Theorem:

- B^* has a compact and convex domain.
- $B^*(u, v)$ is non-empty and convex for all $(u, v) \in \Delta^{m-1} \times \Delta^{n-1}$.
- (Graph Closedness) Suppose that $\{\sigma_n\}$ and $\{\sigma'_n\}$ are sequences in $\Delta^{m-1} \times \Delta^{n-1}$ that converge to σ and σ' respectively. Furthermore, suppose that $\sigma'_n \in B^*(\sigma_n)$ for all n. Then $\sigma' \in B^*(\sigma)$.

For the first item, the domain of B^* is $\Delta^{m-1} \times \Delta^{n-1}$ which clearly satisfies the desired condition.

As for the second and third item, from the definition of B^* the image of any (u, v) consists of convex combinations of Voronoi best responses, which are defined for all (u, v) (thus satisfying non-emptyness), and since they are convex combinations, they are convex subsets of $\Delta^{m-1} \times \Delta^{n-1}$.

Finally for the fourth item, let us consider such a sequence where $\sigma_n = (u_n, v_n)$, $\sigma = (u, v)$, $\sigma'_n = (u'_n, v'_n)$, and $\sigma' = (u', v')$. To show the claim, it suffices to consider the sequences $\{u_n\}$ and $\{v'_n\}$ with respective limits u and v', and show that $v' \in conv(V^C(u))$

To show this however, it suffices to use the fact that Voronoi best response sets are closed. Suppose that u has a certain set $S \subset [n]$ of Voronoi best responses. Then there exists a constant $\mu > 0$ such that $B_\mu(u) \cap VC_i \neq \emptyset$ if and only if $i \in S$; namely the μ neighbourhood around u only intersects Voronoi best response sets from u's Voronoi best responses.

To explicitly construct such a μ, let us consider $D_i = d(u, \widehat{C}_i)$ to be the distance between u and the empirical best response set $\widehat{C}_i$. This means that $S = \text{argmin}_i D_i$, so let us define $D = \min_i D_i$ and $\mu = \frac{\min_{j \notin S} D_j - D}{3}$ which is positive due to the fact that there are finitely many partial best response sets. Now suppose that $x \in B_\mu(u)$, then for any $j \notin S$ we have $d(u, \widehat{C}_j) \leq d(u, x) + d(x, \widehat{C}_j)$ by the triangle inequality, which rearranging gives us: $d(x, \widehat{C}_j) \geq d(u, \widehat{C}_j) - d(u, x) \geq (D + 3\mu) - \mu = D + 2\mu$. On the other hand, for any $i \in S$, $d(x, \widehat{C}_i) \leq d(x, u) + d(u, \widehat{C}_i) \leq D + \mu$. It thus follows that x can only have Voronoi best responses from S.

Now from the fact that $u_n \to u$, then for some $N > 0$, if $n > N$ then $u_n \in B_\mu(u)$. This in turn means that $v_n \in conv(S)$ by assumption, which means that $v \in conv(S)$ as well, which is what we wanted to show. To extend this to σ and σ', it suffices to repeat the previous argument in each component of the correspondence.

Now that the conditions of Kakutani's fixed point Theorem are satisfied, we know of the existence of an (u^*, v^*) such that $(u^*, v^*) \in B^*(u^*, v^*)$. As in the statement of the Theorem, suppose that Voronoi best responses for B^* arise from an $\frac{\varepsilon}{2\sqrt{m-1}}$-close labelling

of Δ^{m-1} and an $\frac{\varepsilon}{2\sqrt{n-1}}$ of Δ^{n-1}, then we know that all Voronoi best responses are ε best responses for both players. The conditions of the fixed point amount to saying that both players are playing convex combinations of Voronoi best responses, therefore (u^*, v^*) is an ε-WSNE. □

With Theorem 4.8 in hand and our algorithms for constructing ε-close labellings, we can put everything together and prove our desired results regarding the query complexity of computing an ε-WSNE in general bimatrix games.

Theorem 4.9. *Suppose that G is an $m \times n$ bimatrix game and let n be constant. We can compute an ε-WSNE using $O(m^{n^2} \log^{n^2}\left(\frac{m}{\varepsilon}\right)) = poly(m, \log\left(\frac{1}{\varepsilon}\right))$ adversarial best response queries.*

Proof. Suppose that $\mathscr{C}$ and $\mathscr{R}$ are the polytope partitions arising from best-response sets in G. This means that $\mathscr{C}$ is a $(m-1, n)$-polytope partition and $\mathscr{R}$ is a $(n-1, m)$-polytope partition. Let $\varepsilon_C = \frac{\varepsilon}{2\sqrt{m-1}}$ and $\varepsilon_R = \frac{\varepsilon}{2\sqrt{n-1}}$. From Theorem 4.8, we know that computing an ε_C-close labelling of $\mathscr{C}$ and a ε_R-close labelling of $\mathscr{R}$ suffice to compute an ε-WSNE of G. We use adversarial CR-GBS on $\mathscr{C}$ and adversarial CD-GBS on $\mathscr{R}$.

n is the number of polytopes in the partition $\mathscr{C}$, which is assumed to be constant. Consequently, Theorem 4.4 states that computing an ε_C-close labelling of $\mathscr{C}$ using CR-GBS uses $O((m-1)^k \log^k\left(\frac{m-1}{\varepsilon_C}\right))$ adversarial queries, where $k = \binom{n}{2}$. Since $k \leq n^2$, we can upper bound the number of queries by $O(m^{n^2} \log^{n^2}\left(\frac{m}{\varepsilon}\right))$.

$n-1$ is the dimension of the ambient simplex in the partition $\mathscr{R}$, which is assumed to be finite. Consequently, Theorem 4.3 states that computing an ε_R-close labelling of $\mathscr{R}$ using CD-GBS uses $O(m^{(n-1)^2} \log^{n-1}\left(\frac{1}{\varepsilon}\right))$ queries. We trivially upper bound this quantity by $O(m^{n^2} \log^{n^2}\left(\frac{m}{\varepsilon}\right))$. Putting everything together, the total query usage is thus $O(m^{n^2} \log^{n^2}\left(\frac{m}{\varepsilon}\right)) = poly(m, \log\left(\frac{1}{\varepsilon}\right))$ as desired. □

ε-WSNE via Pairwise Comparison Oracles

We recall that in Section 4.5.3 we showed that having access to pairwise comparison oracles in a UEPP opens the door to algorithms that are more query-efficient. It is not difficult to see that for a given bimatrix game, G, having access to $\mathcal{Q}_{PW}$ amounts to having access to Q_{PW} for both $\mathscr{R}$ and $\mathscr{C}$. This means that we can apply Theorem 4.5 to get a query complexity for computing ε-WSNE with $\mathcal{Q}_{PW}$ from Chapter 2 that is polynomial in all relevant parameters of G: n, m, and $\log\left(\frac{1}{\varepsilon}\right)$.

Theorem 4.10. *Suppose that G is an $m \times n$ bimatrix game. We can compute an ε-WSNE using $O(n^2 m^2 \log\left(\frac{nm}{\varepsilon}\right))$ pairwise comparison queries.*

We also note that the lower bound in Theorem 4.6 also applies for comparison queries, since the family of games at hand are binary action and hence comparison queries are the same as best response queries. This in turn implies that the computing an ε-WSNE requires precisely $\Theta(\log\left(\frac{1}{\varepsilon}\right))$ pairwise comparison queries.

4.8 A Brief Foray into Multiplayer Games

In this section we partially extend our results from sections 4.6 and 4.7. In particular, we show that in multiplayer games utility functions are Lipschitz continuous as well, which allows us to uncover approximate best-response information when observing different best responses at "nearby" mixed strategy profiles. In addition we generalise our definitions of best response sets and of ε-close labellings to obtain a result similar to Theorem 4.8 where we showed that obtaining a precise enough empirical labelling provides enough information to compute a well-supported approximate Nash equilibrium.

The main difference in the multiplayer setting however is that best response sets are no longer polytopes nor convex, which means that our algorithms for computing empirical labellings via generalisations of binary search no longer apply. This does not preclude us however from simply querying an ε-net, which as we will show will suffice for computing an ε-WSNE.

4.8.1 Notation for Multiplayer Games

We briefly review notation from Chapter 2. For simplicity we will focus on games with n players where each player has a strategy set $\mathcal{A}_i$ consisting of $|\mathcal{A}_i| = k$ pure strategies. It is straightforward to extend our results to more general games where different players have action sets of different cardinalities.

In general, we let $\mathcal{A} = \prod_{i=1}^{n} \mathcal{A}_i$ be the space of all pure strategy profiles of all players. For the i-th player, we also denote $\mathcal{A}_{-i} = \prod_{j \neq i} \mathcal{A}_j$ as the space of all pure strategy profiles of players other than i. Since every player has k actions, it is straightforward to see that all $\mathcal{A}_{-i}$ are isomorphic, hence without loss of generality we can assume that for all i, $\mathcal{A}_{-i}$ is in a canonical representation. For a given pure strategy profile $a \in \mathcal{A}$, we may wish to distinguish the pure strategy taken by the i-th player, and this is done by writing $a = (a_i, a_{-i})$ with $a_i \in A_i$ and $a_{-i} \in A_{-i}$.

We denote the i-th player's mixed strategy space by $\Delta(\mathcal{A}_i)$, and we note it is equivalent to Δ^{k-1}. This means that the space of mixed strategy profiles of all players is $\Delta(\mathcal{A}) = \prod_{i=1}^{n}(\Delta^{k-1}) = (\Delta^{k-1})^n$. In addition, for the i-th player, we also denote $\Delta(\mathcal{A}_{-i}) = \prod_{i=1}^{n}(\Delta^{k-1}) = (\Delta^{k-1})^{n-1}$ as the space of all mixed strategy profiles of players other than i. Once again, since all players have k actions, we can assume without loss of generality

that all $\Delta(\mathcal{A}_{-i})$ have the same canonical representation. For a mixed strategy profile $x \in \Delta(\mathcal{A})$, we may wish to distinguish the mixed strategy of the i-th player by writing $x = (x_i, x_{-i})$ with $x_i \in \Delta(\mathcal{A}_i)$ and $x_{-i} \in \Delta(\mathcal{A}_{-i})$.

Definition 4.21 (Multiplayer Utility Functions)**.** *For any player i and action $r \in \mathcal{A}$, we denote $U_i^r : \mathcal{A}_{-i} \to [0,1]$ as the i-th player's utility for playing r. If $a = (a_i, a_{-i}) \in \mathcal{A}$ is a pure strategy profile, the utility player i receives is $U_i^{a_i}(a_{-i})$ which we denote by $U_i(a)$.*

Utility functions are defined for pure strategy profiles, but we extend the domain to include mixed strategy profiles. In particular, if $x = (x_i, x_{-i})$ is a mixed strategy profile, we let $U_i^r(x_{-i}) = \mathbb{E}_{a_{-i} \sim x_{-i}}(U_i^r(a_{-i}))$ and by extension $U_i(x) = \mathbb{E}_{a \sim x}(U_i(a))$.

As in bimatrix games, for a given player i and $x_{-i} \in \Delta(A)_{-i}$, we let $E_i(x_{-i}) = \max_{r \in A_i} U_i^r(x_{-i})$ be the maximal expected utility player i can obtain against the mixed strategy profile x_{-i} of all other players. We also recall the definition of best response oracles

Definition 4.22 (Best Response Query Oracles)**.** *Let G be an n-player game:*

- *There are n strong best response oracles, $BR_1, ..., BR_n$ as per Section 2.1.*
- *Suppose each $\mathcal{A}_i$ has a strict ordering denoted by $\prec_i$. Then there are n lexicographic best response oracles, $BR_i^\ell : \Delta(\mathcal{A}_{-i}) \to \mathcal{A}_i$ for $i = 1, ..., n$. Each of these is defined by $BR_i^\ell(x) = \min_{\prec_i}(BR_i(x))$.*
- *A family of adversarial best response oracles is a collection of n functions denoted $BR_i^A : \Delta(\mathcal{A}_{-i}) \to \mathcal{A}_i$ for $i = 1, ..., n$. Each of these satisfies $BR_i^A(x) \in BR_i(x)$.*

For completeness we have defined all best response oracles, but we focus on adversarial best response oracles. As mentioned before, they are the weakest from the fact that a lexicographic oracle is a valid adversarial oracle and the fact that adversarial oracles can be simulated with strong best response oracles. This implies that our results for adversarial oracles carry over to other oracle models.

4.8.2 Lipschitz Continuity of Utility Functions

As in Section 4.6, we will show that for each player i and each $r \in \mathcal{A}_i$, U_i^r is a Lipschitz continuous function. In order to do so, we must regard the domain of U_i^r, which is $\Delta(\mathcal{A}_{-i}) = (\Delta^{k-1})^{n-1}$, as a subset of Euclidean space endowed with the ℓ_1 norm.

Lemma 4.19. *For any player i and action $r \in \mathcal{A}_i$, U_i^r is 1-Lipschitz when the domain is endowed with the ℓ_1 norm.*

Proof. Consider an arbitrary player j with $j \neq i$. If we endow $\Delta(\mathcal{A}_j)$ with the ℓ_1 norm, from Lemma 4.15, we know that U_i^r as a function of $x_j \in \Delta(\mathcal{A}_j)$ (which is a component of $\Delta(\mathcal{A}_{-i})$) is 1-Lipschitz. This is because if all mixed strategies other than those of player i and j are fixed, we obtain a $k \times k$ bimatrix game between player i and j. Now let us consider $x, y \in \Delta(\mathcal{A}_{-i})$.

$$\begin{aligned} |U_i^r(x) - U_i^r(y)| &= \left| \sum_{j=1}^{n-1} U_i^r(y_1, .., y_j, x_{j+1}, ..., x_n) - U_i^r(y_1, .., y_{j+1}, x_{j+2}, ..., x_n) \right|, \\ &\leq \sum_{j=1}^{n-1} \left| U_i^r(y_1, .., y_j, x_{j+1}, ..., x_n) - U_i^r(y_1, .., y_{j+1}, x_{j+2}, ..., x_n) \right|, \\ &\leq \sum_{j=1}^{n} \|x_i - y_i\|_1, \\ &= \|x - y\|_1. \end{aligned} \tag{4.1}$$

□

Since Lipschitz continuity is maintained over maxima, with the same Lipschitz constant, we get the following result that says that best responses to nearby mixed strategy profiles are in fact approximate best responses.

Lemma 4.20. *$E_i : \Delta(\mathcal{A}_{-i}) \to [0, 1]$ is 1-Lipschitz when its domain is endowed with the ℓ_1 norm. In particular, if $x, y \in \Delta(\mathcal{A}_{-i})$ and $\|x - y\|_1 \leq \frac{\varepsilon}{2}$, then any $r \in BR_i^s(y)$ is an ε best response to x for player i.*

4.8.3 Nash's Theorem with Discrete Approximations in Multiplayer Games

Just as in bimatrix games, we have a notion of best response sets.

Definition 4.23 (Multiplayer Best Response Sets)**.** *Let G be a game with n players, each with k strategies. For a player i and pure strategy $r \in \mathcal{A}_i$, we define $P_i^j = \{x \in \Delta(\mathcal{A}_{-i}) \mid BR_i^s(x) = r\}$. We say that P_i^j is a best response set corresponding to strategy r for player i, and note that $\{P_i^r\}_{r \in \mathcal{A}_i}$ cover $\Delta(\mathcal{A}_{-i})$.*

In bimatrix games our goal was to learn ε-close labellings of the polytope partitions induced by best response sets. In the multiplayer setting that goal can be generalised.

Definition 4.24 (Multiplayer ε-close labellings)**.** *Let G be a game with n players, each with k actions, and let i be a specific player in the game. Suppose that for each action $r \in \mathcal{A}_i$, $\widehat{P}_i^r \subseteq P_i^r$ is a closed set. We say the collection $\{\widehat{P}_i^r\}_{r \in \mathcal{A}_i}$ is an empirical labelling*

of $\Delta(\mathcal{A}_{-i})$*. If in addition* $\bigcup_{r\in\mathcal{A}_i}\widehat{P}_i^r$ *is an* ε*-net of* $\Delta(\mathcal{A}_{-i})$ *in the* ℓ_1 *norm, we say that the collection* $\{\widehat{P}_i^r\}_{r\in\mathcal{A}_i}$ *is an* ε*-close labelling of* $\Delta(\mathcal{A}_{-i})$*.*

As we will see shortly, if we manage to compute an $\frac{\varepsilon}{2}$-close labelling for all $\Delta(\mathcal{A}_{-i})$, we have enough information to compute an approximate equilibrium.

In bimatrix games, each P_i^r is a polytope, but in multiplayer games, expected utilities are no longer linear in $\Delta(\mathcal{A}_{-i})$. Consequently, best response sets are semi-algebraic sets instead. This means that in general best response set are not connected and thus not convex. Without convexity and polytope structure we can no longer use binary search methods to learn ε-close labellings. As mentioned before, this does not preclude us from computing an ε-close labelling via a brute force method of querying an entire ε-net of $\Delta(\mathcal{A}_{-i})$. Before we show this suffices however, we show the key result of this section: computing $\frac{\varepsilon}{2}$-close labellings for all $\Delta(\mathcal{A}_{-i})$ suffices to compute ε-WSNE. To do so we revisit Voronoi best response sets. We recall $d(x, S)$ denotes the infimum distance of a point, x to a set S.

Definition 4.25 (Multiplayer Voronoi Best Response Functions and Best Response Sets)**.** *Let* G *be a game with* n *players, each with* k *actions, and let* i *be a specific player in the game. Suppose that* $\{\widehat{P}_i^r\}_{r\in\mathcal{A}_i}$ *is an empirical labelling of* $\Delta(\mathcal{A}_{-i})$*. Player* i*'s Voronoi Best Response function is denoted by* $V^i : \Delta(\mathcal{A}_{-i}) \to \mathcal{A}_i$*. The function is defined as* $V^i(x) = \operatorname{argmin}_{r\in\mathcal{A}_i} d(x, \widehat{P}_i^r)$*. We also define player* i*'s Voronoi Best Response Sets as* $V_i^r = \{x \in \Delta(\mathcal{A}_{-i}) \mid V^i(x) = r\}$*.*

If we invoke Lemma 4.20, we obtain the same result as in bimatrix games whereby Voronoi best responses are actually approximate best responses.

Lemma 4.21. *Suppose that* $\{\widehat{P}_i^r\}_{r\in\mathcal{A}_i}$ *is an* $\frac{\varepsilon}{2}$*-close labelling of* $\Delta(\mathcal{A}_{-i})$*, then Voronoi Best Response for player* i *are* ε *Best Responses in* G*.*

In addition, our definition of empirical labellings stipulated that $\widehat{P}_i^r$ are all closed sets. This is a property which is inherited by Voronoi Best Response Sets.

Lemma 4.22. *Voronoi best response sets are closed.*

Proof. The proof is identical to Lemma 4.17 since ℓ_1 distance is still a continuous function of the relevant domain $\Delta(\mathcal{A}_{-i})$. □

We now define the generalisation to the Voronoi Best Response Correspondence from before.

Definition 4.26 (Multiplayer Voronoi Best Response Correspondence)**.** *Suppose that $x \in \Delta(A)$ is a mixed strategy profile. We define the Voronoi Best Response Correspondence $B^* : \Delta(A) \to \mathcal{P}(\Delta(A))$ as follows:*

$$B^*(x) = \prod_{i=1}^{n} conv(V^i(x_{-i})) \subseteq \Delta(\mathcal{A}).$$

Theorem 4.11. *B^* satisfies all the conditions of Kakutani's fixed point Theorem, and hence there exists a mixed strategy profile $x^* \in \Delta(A)$ such that $x^* \in B^*(x^*)$. In particular, if the Voronoi best response for B^* arise from $\frac{\varepsilon}{2}$-close labellings for all $\Delta(\mathcal{A}_{-i})$, then this in turn implies that x^* is an ε-WSNE.*

Proof. As in the proof of Theorem 4.8, the first three conditions of Kakutani's fixed point theorem are trivial. We focus on proving that B^* has a closed graph. It suffices to show the following for any player i: if $\{x_n\}$ is a sequence in $\Delta(\mathcal{A}_{-i})$ that converges to x, and $\{y_n\}$ is a sequence in $\Delta(\mathcal{A}_i)$ converging to y with the property that $y_n \in conv(V^i(x_n))$ for all n, then $y \in conv(V^i(x))$.

To show this, we prove that there exists a constant $\delta > 0$ with the property that $B^1_\delta(x) \cap V^r_i$ if and only if $r \in V_i(x)$. Here $B^1_\delta(x)$ denotes the ℓ_1 ball of radius δ around x. We prove this claim by contradiction.

Suppose instead that for every $\delta > 0$, $B^1_\delta(x)$ contains some point from a V^r_i, where $r \notin V^i(x)$. Let $\delta_1 > 0$ be arbitrary, and let z_1 be the guaranteed point in $B^1_{\delta_1}(x)$ that is contained in a collection of V^r_i where none belong to $V^i(x)$. Let $\delta_2 = \|x - z_1\|_1$. We continue in this fashion where for a given $\delta_k > 0$, we let $z_k \in B^1_{\delta_k}(x)$ be a point contained in a collection of V^r_i where none belong to $V^i(x)$. Accordingly we define $\delta_{k+1} = \|x - z_k\|_1$. Since we can always continue this process, we recover a sequence $\{z_n\}$ of elements in $\Delta(A)_{-i}$ that converges to x. There are only a finite number of Voronoi Best Response sets, hence this sequence must contain an infinite subsequence of points belonging to the same voronoi best response set, say $V^{r'}_i$, where r' does not belong to $V^i(x)$. This however implies that x is a limit point of $V^{r'}_i$, and since this set is closed, this implies that $x \in V^{r'}_i$, which contradicts the fact that $r' \notin V^i(x)$, thus proving our desired claim.

Returning to the sequences $\{x_n\}$ and $\{y_n\}$, the existence of a fixed $\delta > 0$ with the property that $B^1_\delta(x) \cap V^r_i$ if and only if $r \in V_i(x)$ means that for some $N > 0$, if $n > N$, $x_n \in B^1_\delta(x)$, which in turn means that $y_n \in Conv(V^i(x))$. This in turn means that $y \in Conv(V^i(x))$, as desired. Applying this result to each i yields the graph-closedness of B^*, thus establishing that B^* satisfies all the properties of Kakutani's fixed point theorem.

As for the second part of the theorem, suppose that x^* is the guaranteed fixed point of B^* as guaranteed by Kakutani's fixed point theorem. From Lemma 4.21, we know

that Voronoi Best Responses are ε-Best responses. By the definition of B^*, the support of each $x_i^* \in \Delta(\mathcal{A}_i)$ consists of ε best responses to x^*, thus establishing the fact that x^* is an ε-WSNE. □

With the previous theorem in hand, we have established that computing a $\frac{\varepsilon}{2}$-close labellings of all $\Delta(\mathcal{A}_{-i})$ suffices to compute an ε-WSNE. Although the lack of convexity in best response sets prevents us from using binary search techniques, we can always query an ε-net of $\Delta(\mathcal{A}_{-i})$ in the ℓ_1 norm. In this vein, we construct an explicit ε-net for $\Delta(\mathcal{A}_{-i})$.

Lemma 4.23. *$M_\varepsilon^n = \left(\frac{2\varepsilon}{n}\mathbb{Z}\right)^n \bigcap \Delta^n$ is an ε-net in the ℓ_1 norm for Δ^n. Furthermore, $|M_\varepsilon^n| = O\left((\frac{n}{2\varepsilon} + n)^n\right)$.*

Proof. The first claim follows from noting that lattice points lie on vertices of axis-aligned hypercubes of side-length $\frac{2\varepsilon}{n}$. These cubes have a diagonal of length 2ε in the ℓ_1 norm, hence their centres are at most ε-away from a given queried vertex.

As for the cardinality, using a stars and bars argument, if $S = \frac{1}{\kappa}\mathbb{Z}^n \cap \Delta^n$ for some $\kappa \in \mathbb{N}$, then $|S| = \binom{\kappa+n}{n} = O\left((\kappa + n)^n\right)$. □

Since $\Delta(\mathcal{A}_{-i})$ is a product of simplices, we can take products of the above ε-net constructions to in turn obtain an ε-net for $\Delta(\mathcal{A}_{-i})$.

Lemma 4.24. *Suppose that $\varepsilon > 0$ and let $\varepsilon' = \frac{\varepsilon}{n-1}$. Furthermore, let us define the set $H_\varepsilon^{n,k} = (M_{\varepsilon'}^{k-1})^{n-1} \subset \Delta(\mathcal{A}_{-i}) \cong (\Delta^{k-1})^{n-1}$. Then $H_\varepsilon^{n,k}$ is an ε net for $\Delta(\mathcal{A}_{-i})$ in the ℓ_1 norm. Furthermore $|H_\varepsilon^{n,k}| = O\left((\frac{nk}{2\varepsilon})^{nk}\right)$.*

Querying all points in an ε-net trivially gives rise to an ε-close labelling. Consequently, with Lemma 4.24 and Theorem 4.11 in hand, we have proven our main result regarding the query complexity of computing an ε-WSNE using adversarial Best Response Queries.

Theorem 4.12. *Suppose that G is a game with n players with k pure strategies each. One can compute an ε-WSNE of G using $O\left(n(\frac{nk}{\varepsilon})^{nk}\right)$ adversarial Best Response Queries.*

Pairwise Comparison Oracle in Multiplayer Games

We briefly notice that the same lack of convexity in multiplayer best response sets also prevents us from using an approach similar to Comparison-Polytope with access to $\mathcal{Q}_{PW}$. That being said, we recall that we can trivially simulate the best response oracle of any $i \in [n]$, BR_i, via $O(k^2)$ calls to $\mathcal{Q}_{PW}$. Therefore, in the multiplayer setting we can also compute an ε-WSNE of G using $O\left(k^2 n(\frac{nk}{\varepsilon})^{nk}\right)$ calls to $\mathcal{Q}_{PW}$.

4.9 A Brief Foray into Approximate Query Oracles

In this section we informally discuss at a superficial level how some of our results carry over to the approximate query oracles εBR and $\mathcal{Q}^{\lambda}_{QR}$ from Chapter 2.

ε-Best Response Oracles

Suppose that G is an $m \times n$ bimatrix game and that we are given access to adversarial best response oracles εBR^A_R and BR^A_C as per Definition 2.9. A similar result to Lemma 4.16 still holds, since the underlying utility functions of G are Lipschitz irrespective of the oracle used in an algorithm. To be specific, if $u, u' \in \Delta^{m-1}$ are row player mixed strategies such that $\|u - u'\|_2 < \frac{\alpha}{2\sqrt{m-1}}$, then if $\varepsilon BR^A_C(u')$ is an $(\varepsilon + 2\alpha)$ best-response against u. An identical result holds for column player mixed strategies.

With this in hand, we can follow an identical narrative to Section 4.7 to show that in order to compute an $(\varepsilon + \alpha)$-WSNE it suffices to:

- Compute a partition of Δ^{m-1} into closed sets $\widehat{P}_1, ..., \widehat{P}_n$ with the property that i is an ε best-response in each P_i and $\cup_i \widehat{P}_i$ is a $\frac{\alpha}{4\sqrt{m-1}}$-net of Δ^{m-1} in the ℓ_2 norm.
- Compute a partition of Δ^{n-1} into closed sets $\widehat{P}_1, ..., \widehat{P}_m$ with the property that i is an ε best-response in each $\widehat{P}_i$ and $\cup_i \widehat{P}_i$ is a $\frac{\alpha}{4\sqrt{n-1}}$-net of Δ^{n-1} in the ℓ_2 norm.

Suppose that εP_i is the set of $x \in \Delta^{m-1}$ such that $i \in [n]$ is an ε-best response to row player mixed strategy $x \in \Delta^{m-1}$. It follows that the set of $\{\varepsilon P_i\}_{i=1}^n$ indeed cover Δ^{m-1}, but this collection is no longer as a polytope partition, since the individual εP_i may have non-trivial intersections. This means that CD-GBS and CR-GBS as is cannot be used for εBR, but perhaps a similar approach can be adapted from our results.

On the other hand, it is straightforward to notice that we can use an ε-net approach of Theorem 4.12 from the previous section to trivially compute the empirical partitions, $\{\widehat{P}_i\}_{i=1}^n$, of Δ^{m-1} and Δ^{n-1} mentioned above. This implies that we can compute an $(\varepsilon + \alpha)$-WSNE at a query cost of $O\left(n(\frac{nk}{\alpha})^{nk}\right)$ calls to any approximate best response oracle

Quantal Response Oracles

As we noted in Chapter 2, a quantal response oracle $\mathcal{Q}^{\lambda}_{QR}$ cannot in general efficiently simulate either a best response oracle or a pairwise comparison oracle. The reason for this is that the underlying distribution of $\mathcal{Q}^{\lambda}_{QR}$ samples from may place minute differences of probability mass on different actions, and so discerning this with high probability becomes impossible. That being said, we also noted that with knowledge of λ it does

become possible to simulate an adversarial ε-best response oracle with high probability, hence our exposition regarding εBR holds for $\mathcal{Q}_{QR}^{\lambda}$.

In the bimatrix games setting, we notice that with knowledge of λ, we can in fact approximately simulate $\mathcal{Q}_{PW}$ with high probability. What we mean by this is that for a given player $i \in [n]$, pair of actions $j, k \in \mathcal{A}_i$ and an opponent mixed strategy profile $x_{-i} \in \mathcal{A}_{-i}$, we can answer whether $U_i(j, x_{-i}) \geq U_i(k, x_{-i}) - \varepsilon$ with high probability. This alone however does not salvage the use of Comparison-Polytope in this setting. This is due to the fact that an agent, $i \in [n]$, may be approximately indifferent between two actions, $j, k \in \mathcal{A}_i$ irrespective of an opponent strategy x_{-i}. This in turn precludes us from ascertaining the $H_{\alpha}^{i,j}$ and $H_{\alpha}^{j,i}$ with high probability.

4.10 Conclusion and Future Directions

In this chapter we introduced the concept of learning ε-close labellings of (m, n)-polytope partitions with membership queries, and derived query efficient algorithms for when either the dimension of the ambient simplex in the polytope partition, m, is held constant, or when the number of polytopes in the partition, n, is held constant. In addition, we also study the problem of learning ε-close labellings via pairwise comparison queries and show that with this more powerful oracle, we can compute an ε-close labelling with a query cost polynomial in m, n and $\log(\frac{1}{\varepsilon})$.

Most importantly, we introduced a novel reduction from computing ε-WSNE with best response queries to this very geometric problem.This allows us to show that in the best response and pairwise comparison query models, computing ε-WSNE of a bimatrix game has a finite query complexity. More specifically, for $m \times n$ bimatrix games, the pairwise comparison query complexity is polynomial in m, n and $\log(\frac{1}{\varepsilon})$, and in games with one parameter, say n, constant, the best response query complexity is polynomial in m and $\log\left(\frac{1}{\varepsilon}\right)$. Furthermore, we partially extended our results from bimatrix games to n-player games. Although the underlying geometry in n-player games prevents us from using our results from learning polytope partitions, we were still able to show that querying a fine-enough ε-net of the mixed strategy space of all players suffices to compute an ε-WSNE.

As mentioned in the introduction, this geometric framework could be of use in other areas where Lipschitz continuous structures appear over domains with convex partitions. Upon further inspection, it is not difficult to see that polytope partitions do not need to be contained in Δ^m, and in fact our algorithms extend to arbitrary ambient polytopes. Furthermore, it would be of great interest to create algorithms with a better query cost, prove lower bounds with regards to computing ε-close labellings, or simply explore weaker

query paradigms, as mentioned in Section 4.9. Finally, we have mentioned that in the multiplayer setting, best response sets are no longer polytopes, but rather semi-algebraic sets. It would be of interest to create learning algorithms for ε-close labellings of these more complicated geometric objects, since doing so suffices to compute ε-WSNE.

Chapter 5

Bitcoin Overview

We now proceed to the second major portion of this thesis, where we focus on applications of game-theoretic equilibria to Bitcoin. Though this is a departure from query-based equilibrium computation algorithms, the rest of the thesis highlights important game-theoretic issues in incentive-based cryptocurrencies such as Bitcoin.

5.1 Nakamoto Consensus

Just over a decade ago, Satoshi Nakamoto published a landmark white paper outlining the backbone protocol to Bitcoin [56], arguably the world's first mainstream digital currency. Through the Nakamoto protocol and blockchain technology, users can reliably agree upon a common transaction history maintained by miners on the public Bitcoin ledger in a decentralised and anonymous fashion. This is possible in part because agents within the protocol are assumed to be strategic rather than adversarial—a fair assumption when creating a ledger for currency. The Nakamoto protocol is thus at its core a decentralised consensus protocol among strategic agents that is built from cryptographic primitives and judicious incentive engineering.

There are two principal types of agent in Bitcoin: users and miners. Users hold and transfer Bitcoin between addresses via transactions, whereas miners maintain a consistent history of transactions for the Bitcoin ecosystem. In order to update Bitcoin's ledger, miners bundle pending valid transactions into "blocks" which must have a hash value lower than a target that is dynamically adjusted by the Bitcoin protocol (see Figure 5.1). Furthermore, a valid block also points to a predecessor block and contains the hash value of said predecessor block in order to make the ledger "tamper-proof" as changes in the predecessors of a given block can be observed by the changes they incur in hash values. Subsequently, valid blocks are propagated to all agents within Bitcoin's system, and all agents are able to efficiently verify the validity of blocks and transactions therein. To extract a transaction history consensus from this protocol, honest agents agree to use

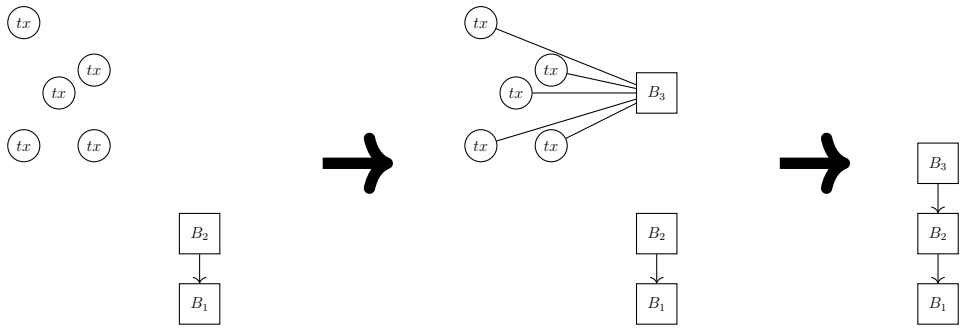

Figure 5.1: We give a very brief overview of some of the main points of the Nakamoto consensus protocol behind Bitcoin. At first there are two blocks in the ledger, each with their own consistent transactions. Circles represent pending valid transactions to be put into the ledger. A miner/pool then collects pending transactions into a block which points to the previous end of the ledger (B_2 in this case). In order for this block to be valid, its hash has to be below a certain threshold (this is the computationally intensive step, as often a miner has to attempt multiple block configurations to attain the threshold). Once this is complete, all other miners can verify that the block with its transactions is valid, and the ledger is thus updated. The existence of this block in the longest consensus path implicitly rewards the miner/pool owning the block with Bitcoin.

the longest path in their local copy of the blockchain as the global transaction history of Bitcoin. Ultimately, miners are rewarded for their computational efforts in maintaining this blockchain, as they earn a block reward for each valid block they contribute to the longest path in the blockchain as well as all transaction fees therein.

Given the financial rewards a miner obtains from updating the blockchain, we model these agents as strategic entities seeking to maximise their amount of Bitcoin. As such, the prescribed honest behaviour of miners is to follow the Bitcoin protocol: mining upon the longest chain in their local history, and immediately propagating valid blocks once found. It is straightforward to see that if all miners within Bitcoin are honest, their block rewards will be proportional to the amount of computational power they have. More specifically, to model miner dynamics, we suppose that there are n strategic miners, m_1,...,m_n, with hash power $\alpha_1, ..., \alpha_n$ respectively. Each α_i represents the proportional amount of computational power m_i has in the Bitcoin ecosystem, such that $\sum_{i=1}^{n} \alpha_i = 1$. To make our previous statement more precise, if there are L blocks in the longest chain of the ledger, then in expectation each m_i will own $\alpha_i \cdot L$ valid blocks in the longest chain if all agents are honest.

Under the specification of Bitcoin, block difficulty is dynamically adjusted so that the entire Bitcoin ecosystem finds a block on average every 10 minutes. Furthermore, given the fierce competition in mining, if a specific miner m_i has a small hash power, α_i, then although in expectation they earn an amount of coin proportional to α_i per block that is mined (as they have an α_i probability of finding any given block in the longest chain), the

variance of this payoff may be huge. For this reason, miners collect their computational resources into pools where rewards are evenly re-distributed on a more regular basis to reduce variance in miner payoffs.

5.2 Literature Review

Selfish mining was originally introduced in [22]. In this work, the authors describe Selfish Mining (SM), a specific mining strategy that deviates from the prescribed honest mining strategy of the Bitcoin network with the key property that it is more profitable than honest mining for miners with over 1/3 of the hash power of the entire Bitcoin network. Subsequently, [58] and [65] identify a generalised class of selfish mining strategies to which SM belongs and show that in general there are more aggressive and profitable strategies than SM within this family of strategies. In a similar vein, [44] uses game theory to formalise the decision a single strategic miner may take to employ different strategies from the generalised family of selfish mining strategies. In particular, they define complete information games that are analogous to real-life mining and show that for these games, if no miner has a large enough hash power, honest mining is a Nash equilibrium.

Perhaps most similar to our work in Chapter 6 is [50], where the authors simulate multiple strategic miners employing strategies other than honest mining. Their results are simulation-based, whereas we provide closed-form results for a specific truncation of selfish mining. In fact, our model can be seen as a variant of the model used in [6], which we developed concurrently to allow for an arbitrary number of strategic agents. Furthermore we take a larger focus on the game-theoretic considerations miners may take in deciding whether to employ truncations of selfish mining in varying degrees.

Subversive mining strategies can also be combined with network level attacks to exacerbate undue profits. This is discussed in [58] where the authors combine selfish mining strategies with eclipse attacks; an eclipse attack is when an entity holds all connections with a subset of the mining swarm and can thus control all communication between them and the rest of the miners. The authors show that no combination of a selfish mining strategy and eclipse attack is optimal at all times. The choice of what selfish mining strategy to adopt as well as how to eclipse a victim is highly dependent on the network parameters in which one is operating. These parameters include computational power, percentage of the network that can be eclipsed, and the percentage of remaining miners that can be influenced.

There are additional attacks miners can wage outside the family of selfish mining. At the pool level, managers can wage withholding attacks as per [16] and [20], where a

malicious pool infiltrates a victim pool, submitting shares and withholding full solutions. Indeed this notion of "partitioning" one's pool is similar to our partition games from Section 6.4.2. [20] shows that this can be profitable for a single malicious pool, but when multiple pools engage in block withholding attacks, this results in a situation akin to the prisoner's dilemma, where the equilibrium of all malicious pools is to infiltrate and thus reduce the overall profit of every pool in the network. Withholding attacks are further refined in [47], where a malicious pool still withholds full solutions from a victim pool, but may share said full solutions when it hears of a full solution being found by a miner outside of the malicious and victim pool. The intent of this strategy is to incentivise the victim pool manager to cause a fork, and this behaviour does away with the prisoner's dilemma of [20], as there are equilibria where larger pools are strictly better off than honest mining. Furthermore, there is some evidence showing that this family of pool-level attacks can be difficult to detect for victim pools [16].

Along with work covering subversive mining attacks and which strategies miners should adopt based on network parameters, there have also been efforts to defeat these attacks. In [21] the authors outline a new blockchain protocol, Bitcoin Next Generation, which decouples leader election and transaction serialization for better scalability. In their new protocol, they propose that when an honest miner is presented with two chains of the same length, they choose which one to mine on uniformly at random. With this change, the lower bound on computational power needed to mine selfishly increases, thus making it harder to act subversively. While this was conjectured to be true and showed to be so with simulation, there are contradictory results. In [65] the authors show that while this change does limit the strength of large selfish miners, it enhances the strength of medium sized selfish miners and that selfish miners with computational power less than 25% can still gain from acting subversively. Recently, [48] have also demonstrated that if miners are allowed to pay portions of their block reward forward to future miners who successfully mine upon their block, then strategic mining of blocks can be disincentivised for larger miners though they may still reap a larger than fair share of the rewards.

The related research we have mentioned until now is mainly involved with strategic mining with respect to block rewards, but our work in Chapter 7 is dedicated to strategic mining within pools with specified payment protocols, see [64] for an extensive survey of pool protocols. [66] give necessary conditions for honest inter-pool mining for a specific family of mining pool protocols. In their paper, honest inter-pool mining is when miners report shares and blocks to the pool manager immediately upon finding them. Not only do they show that certain popular strategies are not incentive compatible or budget balanced (proportional payment, fixed payment per share), but they describe an incentive compatible variant pool protocol that is also budget balanced. In addition, they also

partially answer the question of whether Pay-per-last-N-shares (PPLNS) is an incentive compatible mining pool scheme by exploring when honest mining is robust against the specific strategic deviation where miners keep a single share private with the hopes of finding a subsequent block to ensure said private share is paid by publishing it immediately before revealing a block. Subsequently [76] study a different class of strategic deviations in PPLNS where a miner hoards a certain number $x \in \mathbb{N}$ of shares. Subsequent shares are published immediately, and whenever a block is found, those x shares are published immediately before publishing the block. Their analysis makes the assumption that each strategic miner reaches their threshold x, and show when being honest outperforms being strategic in this setting. Finally, [62] exhibit specific reporting strategies that can be beneficial to strategic miners at high enough hash rates.

Chapter 6

Competing (Semi-)Selfish Miners

6.1 Introduction

One of the key innovations in the Nakamoto protocol behind Bitcoin [56] is the assumption that agents involved in the upkeep of the digital ledger, so called miners, are strategic rather than adversarial, which invites a game-theoretic analysis of the underlying protocol. Under this relaxed assumption, Bitcoin enjoys more robust guarantees on its security: the adage being "it is in a miner's best interest to be honest when there is an honest majority of miners".

This adage however was famously proven to be incorrect in [22], where the authors first described "Selfish Mining", a non-honest miner strategy that gives more returns to miners than honest mining, even if a majority of other agents are honest. Subsequently, there has been much work exploring the extensions and limitations of selfish mining, but most of this work is limited to the case in which there is a single selfish miner and the rest of the network acts honestly. In this chapter we study scenarios where more than one miner deviates from the honest mining protocol. We show that there are substantial game-theoretic differences when multiple miners can be strategic with implications to Bitcoin's security. First of all, there are hash rates where a miner is incentivised to be honest if mining is treated as a one-shot game, yet where the miner is incentivised to be strategic if he is the leader in sequential (Stackelberg) game. Second of all, we show that with multiple strategic miners, specific deviations from honest mining by multiple strategic agents can outperform honest mining, even if individually miners would not be incentivised to be dishonest. These two previous points effectively render the Bitcoin protocol to be less secure than previously thought.

6.1.1 Our Contributions

We study miner incentives when multiple miners employ variants of selfish mining strategies. Original selfish mining (SM) consists of secretly withholding mined blocks and judiciously publishing private blocks in an attempt to increase stale block rates of other miners. Though such an attack is not immediately profitable, as the block rate of all miners decreases, it can be profitable in a longer time horizon as block difficulty rates decrease. In SM, miners may keep an arbitrarily long private chain, which makes it difficult to analytically solve for relative revenues when more than one miner employs SM. For this reason, we study a truncation of this strategy, semi-selfish mining (SSM), where miners keep a private chain of length at most 2.

SSM falls within the family of generalised selfish mining strategies of [65] and [58], and our work begins by studying analytic properties of SSM's performance against honest mining. In Section 6.4.1, we show that although SSM achieves less relative revenue than SM against honest mining, it is always a more profitable strategy for a strategic miner than honest mining if the miner has a hash rate larger than 38% of the total system hash rate, and if the strategic miner is able to propagate blocks to other miners quickly, this threshold lowers to around 26.8%. In fact, we show the relative revenue of SSM is an asymptotically tight lower bound of the relative revenue of SM as a strategic miner's hash power tends to 0.

As mentioned before, the benefit of SSM is that it can be represented with a reduced state space, and hence we can explicitly solve for relative revenues in the case where multiple strategic miners employ SSM. In Section 6.4.2 we focus on systems with two strategic miners and describe the Markov chain that governs block publishing dynamics. This allows us to explicitly solve for relative revenues of all miners in the steady state.

With the steady state solutions in hand, we are able to study the incentives that govern the decision whether a miner uses SSM against another strategic miner. To do so, we define a binary action two-player game amongst both strategic miners which we call the SSM game. In the SSM game both miners are denoted by m_1 and m_2 and they have corresponding utility functions U_1 and U_2. In addition, each miner has the action set $\{H, S\}$ representing honest mining and SSM mining. Interestingly, we find multiple scenarios at different hash rates:

- For all pure strategy profiles, $s \in \{(H,H),(S,H),(H,S),(S,S)\}$, there exist hash rates of both strategic miners such that s is a unique pure Nash equilibrium.
- When both strategic miners have roughly around 0.2 to 0.27 of the system's hash power, both (H,H) and (S,S) are simultaneously pure Nash equilibria of the SSM game.

- There exist hash rates where a specific miner is not unilaterally incentivised to employ SSM, yet (S, S) is the only pure Nash equilibrium of the game. This effectively lowers the minimum hash rate required for SSM to be profitable by virtue of the existence of another strategic miner.

- There exist hash rates where $U_1(S, S) < U_1(H, H) < U_1(S, H)$ (once again, an identical result holds with the roles of miners reversed). This is interesting because although SSM is individually rational for the first strategic miner, the second (larger) strategic miner has the ability to "penalise" the first miner were they to retaliate by using SSM.

We also consider a richer action space for miners: we allow them to partition their hash power into an honest portion and an SSM portion. The game specified by these utilities is called the partition game, and when treated as a one-shot game, it yields the same pure Nash equilibria as the SSM game. The more interesting result stems from treating this game as a Stackelberg game and understanding optimal commitments a miner may make to elicit a desired behaviour in the other miner. It turns out that in the partition game, there exist hash rates with non-trivial Stackelberg equilibria that can result in large gains for leader miners. In fact, there are even hash rates where a miner is honest in the one-shot SSM game, yet strategic in the sequential partition game's Stackelberg equilibrium. This has important consequences for the security of Bitcoin, as miners with smaller hash rates than what was known before may be incentivised to be strategic in a sequential setting.

In Section 6.10 we consider the scenario where $M > 2$ miners are strategic. For $1 \leq M \leq 8$, we compute bounds on the minimal $\alpha \in [0, 1]$ such that if the M strategic miners each with hash power α have to decide between employing honest mining and SSM, the strategy profile where all such miners employ SSM Pareto-dominates honest mining. For each M, we call α the uniform profitability threshold for SSM, and we show that not only is it a decreasing function in M, but that already for $M = 8$, α is as low as 0.11. This is striking, because at such hash rates, miners are far from being individually incentivised to employ SSM, implying that the existence of other strategic miners can effectively hurt the stability of Bitcoin.

We also note that in Section 6.8 we explicitly extend our game-theoretic formalism from Section 6.4.2 and Section 6.7 to the multi-player setting, and we specify how to compute utilities in these games. Furthermore, in Section 6.9 we extensively map incentives of 3 strategic miners akin to Section 6.4.2 and Section 6.7. We find that the game-theoretic observations of the two-player setting generalise appropriately.

6.2 Model Assumptions and Notation

The decentralised design of Bitcoin consists of clients: users of Bitcoin, who own accounts designated by addresses. A client can send Bitcoin from an address he owns to an arbitrary address by broadcasting a transaction to the Bitcoin P2P network. This transaction will eventually be appended to a global ledger called the Blockchain. The upkeep of the Blockchain is performed by miners, who collect transactions in blocks and append these blocks to the chain. For this task, miners are rewarded with Bitcoin, either in the form of a block reward or transaction fees.

We model the Blockchain system as a set of M strategic miners, $m_1, ..., m_M$, and an implicit honest miner m_{M+1}. Each strategic miner m_i, controls an $\alpha_i \in (0, 0.5]$ portion of the system hash power (we don't consider strategic miners strong enough to perform a 51 percent attack), and the honest miner m_{M+1} controls a $\beta = 1 - \sum_{i=1}^{M} \alpha_i > 0$ portion of the system hash power. The implicit honest miner is without loss of generality for if any number of miners (beyond the strategic miners $m_1, ..., m_M$) employ honest mining, this is equivalent to one miner of their combined hash power employing honest mining. For convenience we denote the set of valid strategic miner hash rates by $\mathcal{H}^M = \{\alpha \in (0, 0.5]^M \mid \sum_{i=1}^{M} \alpha_i \leq 1\}$.

Given strategic miner hash rates $\alpha \in \mathcal{H}^M$, any found block has an α_i probability of being found by the i-th strategic miner m_i, and a β probability of being found by m_{M+1}. We also assume that the system overall finds blocks at a rate of λ according to a Poisson process. In terms of the actual implementation of the Bitcoin protocol, λ is roughly one block every 10 minutes, which is ensured by dynamically adjusting the difficulty of the block hash target.

The append-only nature of the block renders the Blockchain into a tree with a root at the genesis block. Since the longest path of the tree is the agreed-upon transaction history, a miner's revenue consists of his block rewards and transaction fees arising from blocks that eventually become a part of the longest path in the Blockchain. In this chapter, we focus on block rewards and normalise such rewards to unit value, hence the revenue of a miner is the number of his blocks that are accepted in the longest path of the blockchain.

Indeed it could be the case that a longest path in the blockchain is eventually surpassed by a competing path: this is a key aspect to selfish mining strategies. This of course makes it difficult to ascertain revenues when miners are arbitrary agents. In our work however, we pit specific mining strategies against each other and hence obtain well-defined block creation rates for all agents involved. Furthermore, we assume that agents are rational and that the utility they wish to maximise is their *relative revenue*: which, for a miner

m_i, is the expected number of blocks m_i publishes in the blockchain normalised by the expected number of blocks produced by all miners $m_1, ..., m_{M+1}$. The justification behind this utility function comes from the fact that Bitcoin dynamically adjusts its difficulty, hence relative revenue in the long-term corresponds to overall revenue.

6.3 Miner Strategies

Mining strategies are often defined with an implicit assumption that a miner following the strategy will be pitted against miners employing a specific strategy (i.e. honest mining). Since our work focuses on miner incentives when multiple miners deviate from honest mining, we find ourselves in need of rigorously defining miner strategies with respect to all possible changes in the blockchain, not just those changes that can occur against a specific kind of miner.

In this vein, we formally describe three specific mining strategies: honest mining, selfish mining, and semi-selfish mining. We describe the strategies for an arbitrary miner denoted by m.

To execute these strategies, m must keep track of their private chains, the public chain, a block upon which to mine and an internal state $\ell \in \{0, 0'\} \cup \mathbb{N}$. As for additional notation, $priv$ denotes the private chain of m, pub denotes the public chain and F (frontier) denotes the set of blocks at the ends of the longest paths of the public chain. Arbitrary blocks are usually denoted by B. We also let $len(priv)$ and $len(pub)$ denote the length of the longest path in the miner's private chain and the length of the longest path of the public chain respectively. For a given set of a blocks S, we let $oldest(S)$ denote the oldest block in S of which m was aware. Finally, we let $end(priv)$ be the block at the end of the miner's private chain and $p(m)$ be the block upon which m is mining.

The integer of the internal state, ℓ represents a miner's "lead": how much longer the miner's private chain is than the public chain. For all three mining strategies states 0 and 0' will not only mean that the miner has no lead with respect to the public chain, but that the miner's private chain is in fact the public chain (a fact which follows from the rules governing the strategies). Finally, the difference between 0 and 0' is that the latter state occurs when there is a tie on the public chain, i.e. $|F| > 1$. The choices available to miners are where to mine, $p(m)$, and whether to reveal parts of their private chain.

6.3.1 Honest Mining

Honest miners are those who follow the prescribed Bitcoin mining protocol faithfully. We describe the strategy in terms of what actions m takes when in states $\ell \in \{0, 0'\}$:

- Case 1: m finds a block, B.
  - m publishes B.
  - $\ell \leftarrow 0$.
  - $p(m) \leftarrow B$.
- Case 2: *pub* changes to *pub*$'$ with frontier F'.
  - If $|F'| = 1$, then $\ell \leftarrow 0$.
  - If $|F'| > 1$, then $\ell \leftarrow 0'$.
  - $p(m) \leftarrow oldest(F')$.

It is straightforward to check that if all miners mine honestly, their expected relative revenue is precisely their hash rate:

Lemma 6.1. *For any $\alpha \in \mathcal{H}^M$, if all strategic miners are honest, the expected (block) reward of any strategic miner m_i is α_i (and β for the extra honest miner m_M).*

6.3.2 Selfish Mining

Eyal and Sirer introduced Selfish Mining (SM) in [20] as a specific strategy that outperforms honest mining when a rational agent has sufficient computational resources. SM can be described by the actions m takes in the following states:

$\ell = 0$ **and** $p(m) = oldest(F)$

- Case 1: m finds a block: B.
  - m keeps B private.
  - $\ell \leftarrow 1$.
  - $p(m) \leftarrow B$.
- Case 2: *pub* changes to *pub*$'$ with frontier F'
  - If $|F'| = 1$, then $\ell \leftarrow 0$.
  - If $|F'| > 1$, then $\ell \leftarrow 0'$.
  - $p(m) \leftarrow oldest(F')$

$\ell = 0'$ **and** $p(m) = oldest(F)$

- Case 1: m finds a block: B.
  - m publishes B.
  - $\ell \leftarrow 0$.
  - $p(m) \leftarrow B$
- Case 2: pub changes to pub' with frontier F'.
  - If $|F'| = 1$, then $\ell \leftarrow 0$.
  - If $|F'| > 1$, then $\ell \leftarrow 0'$.
  - $p(m) \leftarrow oldest(F')$

$\ell \geq 1$ **and** $p(m) = end(priv)$

- Case 1: m finds a block: B.
  - m keeps B private.
  - $\ell \leftarrow \ell + 1$.
  - $p(m) \leftarrow B$.
- Case 2: pub changes to pub' with frontier F' and $k = len(pub') - len(pub) \leq max(\ell - 2, 0)$.
  - m publishes k-prefix of $priv$.
  - $\ell \leftarrow \ell - k$.
  - $p(m) \leftarrow p(m)$.
- Case 3: pub changes to pub' with frontier F' and $k = len(pub') - len(pub) > max(\ell - 2, 0)$.
  - m publishes $priv$, resulting in pub'' with frontier F''
  - If $|F''| = 1$, then $\ell \leftarrow 0$.
  - If $|F''| > 1$, then $\ell \leftarrow 0'$.
  - $p(m) \leftarrow oldest(F'')$.

At a glance, this characterisation of SM may look different to how it is usually described. Upon closer inspection however, one can see that this is equivalent to what was presented in [20]. In particular, the fact that in state $\ell = 0'$, $p(m) = oldest(F)$, means that when a tie involves a block mined by m (as would be the case if they had published a previously private block), they will indeed continue mining upon it, as they will have necessarily seen it first amongst blocks in F.

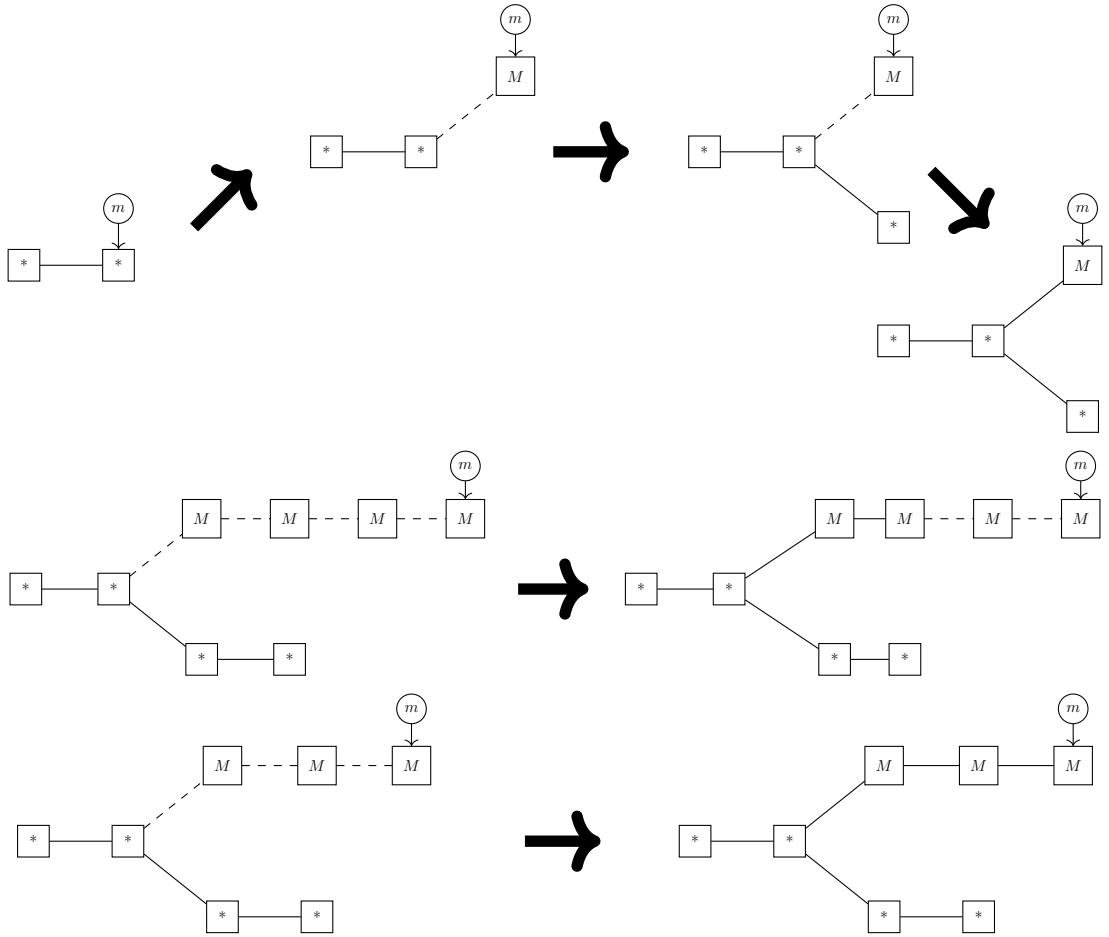

Figure 6.1: SM Dynamics. A square with M is a block mined by m. The circle with m represents the value of $p(m)$. A solid line means that portion of the chain is public, and a dashed line means that portion of the chain is private. If a miner employing SM has a lead of $\ell = 1$ that is diminished, he publishes his private chain and hopes to win the tie (Top). If the miner has a larger lead that is partially encroached, he publishes a prefix of his private chain to push other miners into a race (Middle). If a miner of lead $\ell > 1$ sees his lead encroached to $\ell = 1$, he publishes all blocks to overtake (Bottom).

6.3.3 Semi-Selfish Mining

SM can be generalised to a class of strategies where a miner maintains a private chain and has the following actions at hand: publishing a portion of his private chain, mining

upon his private chain, and foregoing his private chain to mine upon the public chain. Indeed, this general class of selfish mining strategies is studied in [58] and [65].

We focus on the simplest selfish mining strategies from this family by exhibiting a specific strategy, Semi-Selfish Mining (SSM), where the selfish miner never maintains a private chain of length greater than 2 and prioritises breaking ties immediately. Notice that the first characteristic of SSM is necessary if the selfish miner is to gain any benefit from selfish mining, for if the miner only maintains at most one private block, he can only hurt his chances of having this block (and hence any block) published when facing honest miners. Furthermore, much like the original SM strategy, we can show that SSM leads to increased revenue ratios for the selfish miner if they have sufficient hash power.

The reason we study such a simple strategy from the rich space of selfish mining strategies is that it still obtains higher relative revenues than honest mining in certain parameter regimes, yet it has a much simpler state space than most selfish mining strategies. This reduced state space will eventually allow us to explicitly solve for expected relative revenues when two selfish miners play against each other. SSM can be described by the actions m takes in the following states:

$\ell = 0$ **and** $p(m) = oldest(F)$

- Case 1: m finds a block: B.
  - m keeps B private.
  - $\ell \leftarrow 1$.
  - $p(m) \leftarrow B$.
- Case 2: pub changes to pub' with frontier F'
  - If $|F'| = 1$, then $\ell \leftarrow 0$.
  - If $|F'| > 1$, then $\ell \leftarrow 0'$.
  - $p(m) \leftarrow oldest(F')$

$\ell = 0'$ **and** $p(m) = oldest(F)$

- Case 1: m finds a block: B.
  - m publishes B.
  - $\ell \leftarrow 0$.
  - $p(m) \leftarrow B$
- Case 2: pub changes to pub' with frontier F'.

- If $|F'| = 1$, then $\ell \leftarrow 0$.
- If $|F'| > 1$, then $\ell \leftarrow 0'$.
- $p(m) \leftarrow oldest(F')$

$\ell = 1$ **and** $p(m) = end(priv)$

- Case 1: m finds a block: B.
  - m keeps B private.
  - $\ell \leftarrow 2$.
  - $p(m) \leftarrow B$.
- Case 2: pub changes to pub' with frontier F' and $k = len(pub') - len(pub) = 0$.
  - m does nothing.
- Case 3: pub changes to pub' with frontier F' and $k = len(pub') - len(pub) > 0$.
  - m publishes $priv$, resulting in pub'' with frontier F''
  - If $|F''| = 1$, then $\ell \leftarrow 0$.
  - If $|F''| > 1$, then $\ell \leftarrow 0'$.
  - $p(m) \leftarrow oldest(F'')$.

$\ell = 2$ **and** $p(m) = end(priv)$

- Case 1: m finds a block: B.
  - m publishes $oldest(priv \setminus pub)$.
  - $\ell \leftarrow 2$.
  - $p(m) \leftarrow B$.
- Case 2: pub changes to pub' with frontier F' and $k = len(pub') - len(pub) = 0$.
  - m does nothing.
- Case 3: pub changes to pub' with frontier F' and $k = len(pub') - len(pub) > 0$.
  - m publishes $priv$, resulting in pub'' with frontier F''
  - If $|F''| = 1$, then $\ell \leftarrow 0$.
  - If $|F''| > 1$, then $\ell \leftarrow 0'$.
  - $p(m) \leftarrow oldest(F'')$.

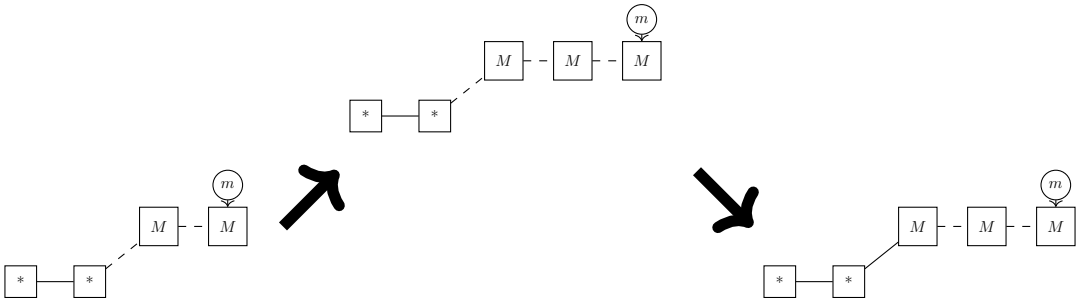

Figure 6.2: SSM as a truncation of SM. Once again, a square with M is a block mined by m. The circle with m represents the value of $p(m)$. A solid line means that portion of the chain is public, and a dashed line means that portion of the chain is private. Here m has a lead of $\ell = 2$ and upon mining a block, publishes his oldest private block.

6.4 Relative Revenue Computation

6.4.1 Warmup: One Strategic Miner

We begin by studying how one strategic miner of hash power $\alpha \in \mathcal{H}^1 = (0, 0.5]$ performs against honest miners of hash power $\beta = 1 - \alpha$ in terms of relative revenue. As in [20], we let γ be the proportion of honest miners who mine upon an SSM chain in the case of a tie, a parameter which we call the propagation of the strategic miner (as this is related to a miner's ability to propagate their blocks across the underlying Bitcoin P2P network). In what follows, we let r_{SM} and r_{SSM} be the expected block creation rate of a single miner using SM and SSM respectively against honest miners. Consequently, we let r_{others} be the block creation of other honest miners in the system (this is dependant upon whether SM or SSM is used, but we use the same term for the sake of simplicity). Finally, we let R_{SM} and R_{SSM} denote the relative revenues of a single miner using SM and SSM respectively against honest miners.

Theorem 6.1 (Selfish Mining Relative Revenue [20] [70])**.** *A single strategic miner of hash power α and propagation γ attains the following revenue ratio using SM against honest miners:*

$$R_{SM} = \frac{r_{SM}}{r_{SM} + r_{others}} = \frac{\alpha(1-\alpha)^2(4\alpha + \gamma(1-2\alpha)) - \alpha^3}{1 - \alpha(1 + (2-\alpha)\alpha)}.$$

Asymptotically around $\alpha = 0$ the expression is the following:

$$R_{SM} = \alpha\gamma + \alpha^2(4 - 3\gamma) + \alpha^3(4\gamma - 5) + \alpha^4(7 - 5\gamma) + \alpha^5(6\gamma - 7) + O(\alpha^6).$$

We can use a similar Markov chain analysis to derive the revenue ratio of SSM against honest miners. We recall that the strategic miner, m_1, has hash power $\alpha \in \mathcal{H}^1 = (0, 0.5]$ and the honest miner m_2 has hash power $\beta = 1 - \alpha$. Let us define the state space

$S = \{S_0, S_1, S_2\}$ corresponding to the number of private blocks belonging to the miner employing SSM. We can now describe the transitions and their corresponding revenues (expected block creation rate per state):

Transitions from state S_0

- $S_0 \to S_0$ occurs if m_2 find a block. The probability of this transition is β and m_2 wins a block.
- $S_0 \to S_1$ occurs if m_1 finds a block. The probability of this transition is α and no players win a block.

Transitions from state S_1

- $S_1 \to S_0$ occurs if m_2 finds and publishes a block, which occurs with probability β. A fork is created when m_1 subsequently publishes his hidden block and from here three events can occur: A first scenario occurs when m_1 finds another block to resolve the tie in his favour, resulting in two blocks for m_1. This occurs with probability α. A second scenario occurs when an honest miner finds a block that resolves the tie in favour of m_1, resulting in one block for m_1 and one block for m_2. This occurs with probability $\gamma\beta$. A final scenario occurs when an honest miner finds a block that resolves the tie in favour of m_2 which results in two blocks for m_2. This final event occurs with probability $(1-\gamma)\beta$. In all aforementioned scenarios the resulting state is S_0, thus the probability of the transition to state S_0 is β.
- $S_1 \to S_2$ occurs if m_1 finds a block and keeps it private as per SSM. This event occurs with probability α and no blocks are awarded to any agent.

Transitions from state S_2

- $S_2 \to S_0$ occurs when m_2 finds a block. The probability of this transition is β and m_1 wins two blocks.
- $S_2 \to S_2$ occurs if m_1 finds a block. The probability of this transition is α and m_1 wins a block.

The transitions are visualised in Figure 6.3. Furthermore, we can fully express the transition matrix of the Markov chain as follows:

$$M = \begin{bmatrix} 1-\alpha & 1-\alpha & 1-\alpha \\ \alpha & 0 & 0 \\ 0 & \alpha & \alpha \end{bmatrix}$$

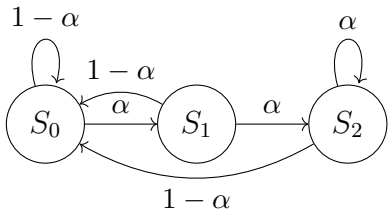

Figure 6.3: States and Transitions for Single SSM vs. Honest Miners.

For a given probability distribution $x \in \mathbb{R}^3$ over the state space S, Mx gives the resulting probability distribution over S after one transition under the Markov chain above. Since the chain is easily seen to be ergodic, there exists a unique steady state distribution, π, such that $M\pi = \pi$. Using Gaussian elimination we obtain $\pi = (1-\alpha, \alpha(1-\alpha), \alpha^2)^T$ as the unique steady state. Furthermore, from the transitions mentioned above we obtain the following expected block creation rates (denoted by r_{SSM} and r_{others}) per state:

Table 6.1: Expected revenue per state

State	$\mathbb{E}(r_{SSM} \mid S_i)$	$\mathbb{E}(r_{others} \mid S_i)$
S_0	0	$1-\alpha$
S_1	$(1-\alpha)(\gamma(1-\alpha)+2\alpha)$	$(1-\alpha)(\gamma(1-\alpha)+2(1-\alpha)(1-\gamma))$
S_2	$\alpha + 2(1-\alpha)$	0

We let r_{SSM} and r_{others} denote the expected revenue per round at steady state π for m_1 and m_2. We also let R_{SSM} and R_{others} denote the revenue ratios of m_1 and m_2 at steady state. Given our expected revenues per state, we obtain $r_{SSM} = (2-\gamma)\alpha^4 + (3\gamma - 5)\alpha^3 + (4-3\gamma)\alpha^2 + \gamma\alpha$ and $r_{others} = (1-\alpha)^2\left((\gamma-2)\alpha^2 + (2-\gamma)\alpha + 1\right)$.

Theorem 6.2. *A strategic miner of hash power α and propagation γ, attains the following revenue ratio when using SSM against honest miners:*

$$R_{SSM} = \frac{r_{SSM}}{r_{SSM} + r_{others}} = \frac{\alpha(\alpha(\alpha(2\alpha-5)+4) - (\alpha-1)^3\gamma)}{(\alpha-1)\alpha^2 + 1}.$$

Asymptotically around $\alpha = 0$, the expression is the following:

$$R_{SSM} = \alpha\gamma + \alpha^2(4-3\gamma) + \alpha^3(4\gamma-5) - \alpha^4(6-5\gamma) + \alpha^5(7\gamma-9) + O(\alpha^6).$$

Comparing Performance of SM and SSM

Asymptotically SM and SSM have the same performance as $\alpha \to 0$. In fact $R_{SM} - R_{SSM} = O(\alpha^4)$. For all parameter settings SM outperforms SSM, as evidenced in the graphs in Figure 6.4. At $\gamma = 0$ SM becomes profitable at $\alpha = 1/3$ and SSM becomes profitable at

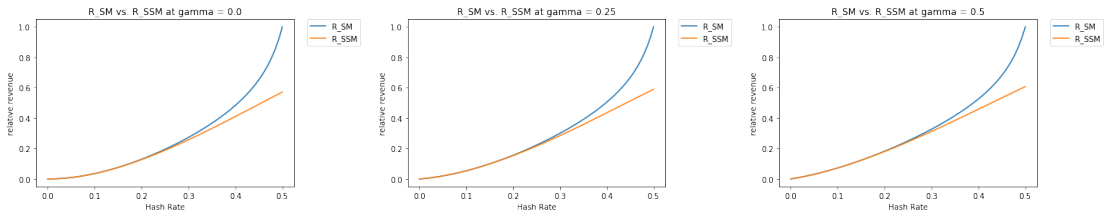

Figure 6.4: R_{SM} and R_{SSM} against honest miners at $\gamma = 0$, 0.25 and 0.5.

$\alpha = 0.38$. At $\gamma = 0.25$ SM becomes profitable at $\alpha = 0.3$ and SSM becomes profitable at $\alpha = 1/3$. Finally, at $\gamma = 0.5$ SM becomes profitable at $\alpha = 1/4$ and SSM becomes profitable at $\alpha = 0.26795$

6.4.2 $M = 2$ Strategic Miners

The benefit of SSM lies in the fact that it can be a rational strategy distinct from honest mining and more importantly, describing it in terms of a Markov chain does not require many states. The simplicity of the state space allows us to explore the scenario where two agents of different hash rates employ SSM and analytically solve for relative revenues.

Markov Chain Analysis

Suppose that $\alpha = (\alpha_1, \alpha_2) \in \mathcal{H}^2$ is the strategic hash rate of the system. Since we have two strategic miners, our state space, S, consists of nine states of the form $S_{i,j}$ where $0 \leq i, j \leq 2$. These represent the relative lead SSM miners 1 and 2 have with respect to the public chain. Given our description of SSM we can describe the state transitions as per Section 6.5.

Transition Matrix and Steady State

The above state space gives rise to an ergodic Markov chain, so there is a unique stationary distribution we can solve for. In order to do so, we define the following transition matrix, P, on $\mathbb{R}^9$, where the coordinate axes of $\mathbb{R}^9$ (in ascending order) represent probability mass in states $S_{0,0}, S_{0,1}, S_{1,0}, S_{0,2}, S_{1,1}, S_{2,0}, S_{1,2}, S_{2,1}$, and $S_{2,2}$ respectively. Each $P_{x,y}$ is

the probability of transitioning to state x from state y in the Markov chain.

$$P = \begin{bmatrix} \beta & \beta & \beta & \beta & \beta & \beta & \alpha_2(1-\alpha_2)+\beta & \alpha_1(1-\alpha_1)+\beta & 1 \\ \alpha_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha_2 & 0 & \alpha_2 & 0 & 0 & \alpha_2^2 & 0 & 0 \\ 0 & \alpha_1 & \alpha_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_1 & 0 & 0 & \alpha_1 & 0 & \alpha_1^2 & 0 \\ 0 & 0 & 0 & \alpha_1 & \alpha_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 & 0 \end{bmatrix}$$

Since this is an ergodic chain, there is a unique steady state distribution, π, such that $P\pi = \pi$, which we can solve for with Gaussian elimination.

Propagation and Revenues

In the original selfish mining paper, much attention was given to the propagation parameter γ. Indeed block dissemination is important because it allows an attacker to persuade other miners to work on their chain in the case of a tie. We also note from the previous section that the steady state distribution π is independent of the propagation of the system. The expected number of blocks published per state however, crucially depends on the propagation of the system, and these two objects specify the relative revenue of agents.

In our work, when there is a single strategic miner employing SSM, the ability to propagate blocks is parametrised by γ as in the original analysis of SM. When there are strategic miners employing SSM however, how propagation is modelled becomes more complicated, since different strategic miners may have a different influence on the P2P network topology. For the rest of the chapter we assume that propagation is uniform. In other words, whenever there is a tie in the public chain (of arbitrary size), all miners not involved in the tie are assumed to have a uniformly random chance of contributing their hash power to any element of the tie. Under the assumption of uniform propagation, we can compute the expected block rate per state of the Markov chain for both strategic miners and honest miners. The following matrix R encodes this information: the first and second row are expected block rates per state for the first and second strategic miners respectively, the third column is the block creation rate for honest miners. If π is a steady state vector for M above, then $R^T\pi \in \mathbb{R}^3$ gives steady state expected block creation rates for all miners.

$$R = \begin{bmatrix} 0 & 0 & \beta \\ \beta\alpha_1 & 2\beta\alpha_2 + \frac{1}{2}\beta(1-\alpha_2) & \frac{1}{2}\beta\alpha_1 + \frac{3}{2}\beta^2 \\ 2\beta\alpha_1 + \frac{1}{2}\beta(1-\alpha_1) & \beta\alpha_2 & \frac{1}{2}\beta\alpha_2 + \frac{3}{2}\beta^2 \\ 0 & \alpha_2 + 2\beta & 0 \\ 2\beta\alpha_1 + \frac{1}{3}\beta^2 & 2\beta\alpha_2 + \frac{1}{3}\beta^2 & \frac{4}{3}\beta^2 \\ \alpha_1 + 2\beta & 0 & 0 \\ 0 & 2\beta + 2\alpha_2^2 + 3\alpha_2(1-\alpha_2) & 0 \\ 2\beta + 2\alpha_1^2 + 3\alpha_1(1-\alpha_1) & 0 & 0 \\ 3\alpha_1 + 3\beta\alpha_1 + \frac{1}{2}\beta^2 & 3\alpha_2 + 3\beta\alpha_2 + \frac{1}{2}\beta^2 & \beta^2 \end{bmatrix}$$

In the following section we include a model for different propagation rates when two strategic miners are involved as well as their effects on relative revenues of all miners.

6.5 Markov Chain Formalism for M Strategic Miners

In this section we delve into the Markov chain governing revenues (block creation rates) when multiple strategic miners employ SSM. In what follows we assume that $\alpha \in \mathcal{H}^M$. This implies that $m_1, ..., m_M$ are strategic miners with hash power $\alpha_1, ..., \alpha_m$, and m_{M+1} is an honest miner with hash power $\beta = 1 - \sum_{i=1}^{M} \alpha_i$.

As in the one miner case, we let $S = \{0, 1, 2\}^M$ be the state space of all possible private leads held by $m_1, ..., m_M$ employing SSM. For a given $x \in S$, x_i denotes the private lead of m_i. In addition, for a given $x \in S$, we let $A_x = \{i \in [M] \mid x_i = 1\}$ and $B_x = \{i \in [M] \mid x_i = 2\}$. Clearly $A_x \cap B_x = \emptyset$, furthermore, we can completely establish transition probabilities from x by looking at A_x and B_x.

6.5.1 State Transitions

Let us suppose $x \in S$ is arbitrary. In what follows we let $e_i \in \{0, 1\}^M$ be the unit vector with 1 in the i-th coordinate. Furthermore, we let $P_{x \to y}$ denote the probability of transitioning from x to y. To fully describe all transitions for any $x \in S$, we look at four different cases depending on A_x and B_x.

Case: $|A_x| \geq 0$**,** $|B_x| = 0$

If any strategic miner m_i obtains a block, they keep it private as per SSM extending their private chain by 1 (which they forcibly have a margin to do so). This results in state $x + e_i$ and occurs with probability α_i. If m_{M+1} finds a block, they publish it as per the honest mining protocol, which occurs with probability β. All m_i such that $x_i \neq 0$ then publish their private chains as per SSM and a race ensues. The conditions of SSM and honest mining dictate that the race is settled in the following turn, and hence we return to state 0. In summary:

- $P_{x \to x+e_i} = \alpha_i$ for all $i \leq M$.
- $P_{x \to 0} = \beta$.

Case: $|A_x| = 0$**,** $B_x = \{j\}$**.**

In this case a single miner has a private lead of 2 and all other miners have no private lead. If any strategic miner m_i such that $i \neq j$ finds a block, SSM dictates that they keep this block private and proceed to having a private chain of length 1. This corresponds to transitioning from x to $x + e_i$, which occurs with probability α_i. If the m_j finds a block, an event which happens with probability α_j, SSM dictates he publish his oldest private block. Since $A_x = \emptyset$, this block will be the longest public chain, and the resulting state will be x again. Finally, if m_{M+1} finds a block, honest mining dictates he publish it. m_j in turn sees his private lead decrease to 1 and hence publishes his entire private chain. As a consequence state 0 ensues, and this transition occurs with probability β. In summary we have the following transitions:

- $P_{x \to x+e_i} = \alpha_i$ for all $i \neq j$.
- $P_{x \to x} = \alpha_j$.
- $P_{x \to 0} = \beta$.

Case: $|A_x| \geq 0$**,** $|B_x| > 1$**.**

Suppose that m_i such that $i \notin B_x$ finds a block, which occurs with probability α_i. As per SSM m_i has a margin to keep this block private, hence state $x + e_i$ ensues. On the other hand, if m_i is such that $i \in B_x$, then by SSM, m_i publishes their oldest private block. As a result, all miners in A_x publish their private leads to start a race, and all miners in B_x publish their private leads to overtake. m_i thus sees his private lead diminish to 1, hence by SSM he publishes his entire private chain. This chain is the longest of all miners, hence we return to state 0. Finally, if m_{M+1} finds a block, which occurs with prability β, he publishes it as per honest mining, all strategic miners with hidden chains once again publish their hidden chains. There is a multi-way race amongst all miners in B_x, but as per SSM and honest mining, this race is decided in the following turn and we return to state 0. In summary we have the following transitions:

- $P_{x \to x+e_i} = \alpha_i$ for $i \notin B_x$.
- $P_{x \to 0} = \beta + \sum_{i \in B_x} \alpha_i$.

Case: $|A_x| > 1$, $B_x = \{j\}$.

If any m_i such that $i \neq j$ finds a block, an event which occurs with probability α_i, then SSM dictates they keep this block private and the resulting state is $x + e_i$. If m_{M+1} finds a block, which occurs with probability β, they publish it as per honest mining, and m_j sees his lead diminished and by the rules of SSM, publishes his private chain to create the longest public chain. The resulting state is thus 0. Finally, if m_j finds the following block, he publishes his oldest private block as per SSM, and consequently the public tie is amongst a prefix of the chain of m_j the chains of all m_i such that $i \in A_x$, since they also publish their private chains. At this point m_j is mining upon his private chain whereas all other miners, including m_{M+1} mine upon some of the chains partaking in the public tie. From here there are two scenarios. Either m_j also finds the following block, in which case SSM dictates he publish it, and the new public prefix of his chain is the longest public chain and the ensuing state is $2e_j$, or any miner other than m_j finds the next block, in which case m_j sees his lead diminished and publishies his entire private chain resulting in state 0. The overall probability of the first scenario is α_2^2 and the overall probability of the second scenario is $\alpha_2(1 - \alpha_2)$. In summary we have the following transitions:

- $P_{x \to x+e_i} = \alpha_i$ for $i \neq j$.
- $P_{x \to 2e_j} = \alpha_j^2$.
- $P_{x \to 0} = \beta + \alpha_j(1 - \alpha_j)$.

6.5.2 Propagation Formalism

In the original analysis of selfish mining, much attention was given to a data propagation parameter γ. Propagation is important because it allows an attacker to persuade honest miners to work on their end the public chain when forks occur.

When there are $M \geq 2$ strategic miners however, propagation intricacies cannot be captured by a single parameter, as different strategic agents have different abilities to convince other miners of their own chains. To encompass this generality, let us suppose that $D \subseteq [M + 1]$ is a subset of miners engaged in a tie (we recall that m_{M+1} is the implicit honest miner in the system). For $j \in D$ and $i \in [M + 1]$ we let $\gamma_{i,j}^D$ be the probability that m_i mines upon the chain of m_j in the tie composed of all D miners. The only restriction we place on these parameters is that $\gamma_{i,i}^D = 1$ for $i \in D$ and $i \neq M + 1$. The reason for this is that a strategic miner will mine upon their public chain in case of a tie. Finally, we note that in the uniform propagation model we use throughout the chapter, we simply let $\gamma_{i,j}^D = \frac{1}{|D|}$ if $i \notin D$ and $i \neq j$ or if $M + 1 \in D$ and $i = j = M + 1$.

6.5.3 Expected Revenue per State with Arbitrary Propagation

For a given state $x \in S$, we compute the expected revenue per agent under the underlying Markov chain governing SSM dynamics. We denote this quantity by $rev(x) \in \mathbb{R}^{M+1}$, where $rev(x)_i$ denotes the expected revenue of m_i when the system is in state $x \in S$.

In order to compute these quantities, it will be useful to define the expected revenue all agents obtain when there is an arbitrary tie involving a set $D \subseteq [M]$ of miners. As we have seen in state transitions, for a given $x \in S$, ties can involve either agents with a private lead of 1 or agents with a private lead of 2. We denote the expected revenue all agents receive when a tie of $D \subseteq [M]$ miners with private lead of $i = 1, 2$ occurs by $T_i(D) \in \mathbb{R}^{M+1}$:

- $T_1(D) = \sum_{i=1}^{M+1} \alpha_i \left(\sum_{j \in D} \gamma_{i,j}^D (e_i + e_j) \right)$.
- $T_2(D) = \sum_{i=1}^{M+1} \alpha_i \left(\sum_{j \in D} \gamma_{i,j}^D (e_i + 2e_j) \right)$.

As with state transitions, for a given $x \in S$, we can characterise $rev(x)$ by looking at A_x and B_x, the indices of strategic miners with a private lead of 1 and 2 respectively.

Case: $|A_x| = |B_x| = 0$**.**

In this case, only m_{M+1} revceives a block if he finds one, which occurs with probability β and we obtain:

- $rev(x) = \beta e_{M+1}$.

Case: $|A_x| > 0$**,** $|B_x| = 0$**.**

In this case, blocks are only won in the event of a tie, which in turn only happens if m_{M+1} originally finds a block with probability β. In such a case, there is a tie amongst the indices $A_x \cup \{M+1\}$. Overall, this yields:

- $rev(x) = \beta \left((T_1(A_x \cup \{M+1\}) \right)$.

Case: $|A_x| = 0$**,** $B_x = m_j$**.**

In this case, if m_j finds a block, he publishes his oldest block as per SSM and thus wins a block in the turn. If m_{M+1} finds and publishes a block as per honest mining, m_j publishes his entire chain as per SSM and wins two blocks in the turn. Overall this yields:

- $rev(x) = \alpha_j(e_j) + \beta(2e_j)$.

Case: $|A_x| \geq 0$**,** $|B_x| > 1$**.**

If any m_j such that $j \in B_x$ finds a block, by SSM they publish their oldest private block. Other miners with indices in B_x thus publish their entire private chains of length 2, and consequently m_j publishes his entire private chain of length 2 (relative to the original fork so still longer than all other private chains), winning 3 blocks overall. If any m_i such that $i \notin B_x$ finds a block, they simply keep it private as per SSM and no blocks are definitively won. Finally, if m_{M+1} finds a block with probability β, then all m_j such that $j \in B_x$ publish their private chains and a tie ensues amongst these agents, which results in $T_2(B_x)$ expected revenue for all miners. Overall, this results in:

- $rev(x) = \sum_{j \in B} \alpha_j(3e_j) + \beta T_2(B_x)$.

Case: $|A_x| > 1$**,** $B_x = m_j$**.**

If m_{M+1} finds a block with probability β, m_j sees his lead diminished and publishes his entire private chain, thus winning two blocks. If any m_i such that $i \notin B_x$ finds a block, they simply keep it private as per SSM and no one immediately wins blocks. Finally, if m_j finds two blocks in a row he publishes a prefix of his private chain and wins two blocks (transitioning to state $2e_j$ in the process). If m_j finds a block (thus leading him to publish his oldest private block as per SSM), and subsequently any other miner finds the next one, m_j sees his lead diminished and publishes his entire private chain, winning 3 blocks overall. Overall, we obtain:

- $rev(x) = \beta(2e_j) + \alpha_j^2(2e_j) + \alpha_j(1 - \alpha_j)(3e_j)$.

6.6 To SSM or not to SSM? A Revenue Analysis

Although our Markov chain analysis gives us a closed-form solution for the relative revenue of both strategic miners when using SSM, the expression is unwieldy. We can however explicitly solve the expression for specific hash values, α_1 and α_2 and use these values to describe a two-player, binary action game governing the decision as to whether a player employs SSM or not.

Suppose that $\alpha = (\alpha_1, \alpha_2) \in \mathcal{H}^2$ describes the hash rates of both strategic miners. We let $R_{SSM}(\alpha) = R_{SSM}((\alpha_1, \alpha_2)) \in [0, 1]^3$ be the revenue ratios of all miners (including the honest miner m_3) when both strategic miners employ SSM. Specifically, $R_{SSM}(\alpha)_i$ is the revenue ratio of m_i for $i = 1, 2, 3$. With this in place we can define a two-player binary action game governing the incentives behind employing SSM or not for m_1 and m_2.

Definition 6.1 (Two-player SSM Games)**.** *Suppose that $\alpha = (\alpha_1, \alpha_2) \in \mathcal{H}^2$ is a strategic hash distribution. We define the* SSM Game*, G_α as a two-player binary action game. In G_α each strategic miner has a binary action set $\{H, S\} \cong \{0, 1\}$, where $H \cong 0$ represents mining honestly and $S \cong 1$ represents employing SSM. We define the utilities of all pure strategy profiles as follows:*

- $U_1(H, H) = \alpha_1$, $U_2(H, H) = \alpha_2$.
- $U_1(H, S) = \frac{\alpha_1}{1-\alpha_2} R_{SSM}((0, \alpha_2))_3$, $U_2(H, S) = R_{SSM}((0, \alpha_2))_2$.
- $U_1(S, H) = R_{SSM}((\alpha_1, 0))_1$, $U_2(S, H) = \frac{\alpha_2}{1-\alpha_1} R_{SSM}((\alpha_1, 0))_3$.
- $U_1(S, S) = R_{SSM}(\alpha)_1$, $U_2(S, S) = R_{SSM}(\alpha)_2$.

For notational convenience, we interchangeably denote a pure strategy profile of all players by either a tuple, as in (H, H) for both miners employing honest mining, or a string, as in HH.

As a first region of interest, in Figure 6.5 we display hash rates where $U_1(S, S) < U_1(H, H) < U_1(S, H)$. For such α, although SSM may be unilaterally rational for the first strategic miner, a larger miner can penalise the first strategic miner for deviating from the honest protocol by retaliating with SSM. As a specific example of this phenomenon, let us consider the hash distribution $\alpha = (0.33, 0.48)$ which leads to G_α with utilities summarised in Table 6.2. The second, larger, strategic miner m_2 can retaliate from SH by deviating to SS, in which case m_1 is worse off by approximately 0.04 in utility than if he had mined honestly at the outset.

Table 6.2: Example of G_α where m_2 can retaliate against SH

$\alpha = (0.33, 0.48)$	HH	HS	SH	SS
U_1	0.33	0.26794954	0.35517387	0.29387121
U_2	0.48	0.57777649	0.46196499	0.61890781

Now that we have defined the game G_α, it is natural to ask about what equilibria it has. Our results suggest that for all values of $\alpha \in \mathcal{H}^2$, G_α has at least one pure Nash equilibrium (PNE), so that if we let PNE(G) denote the PNE of a given game G, PNE(G_α) $\neq \emptyset$ for $\alpha \in \mathcal{H}^2$. In the first image of Figure 6.6 we show which regions of $\mathcal{H}^2$ demonstrate different combinations of PNE. For the most part, hash rates lead to a single PNE in G_α, with distinct regions where each pure strategy profile (HH, HS, SH, and SS) occurs as a sole equilibrium. The most interesting observation however, is that for α

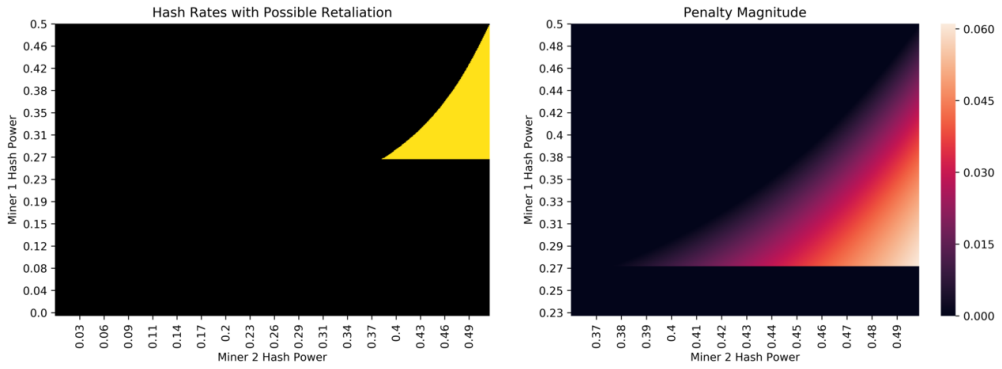

Figure 6.5: Hash rates where $U_1(S,S) < U_1(H,H) < U_1(S,H)$ and subsequent penalty values given by $U_1(S,S) - U_1(H,H)$.

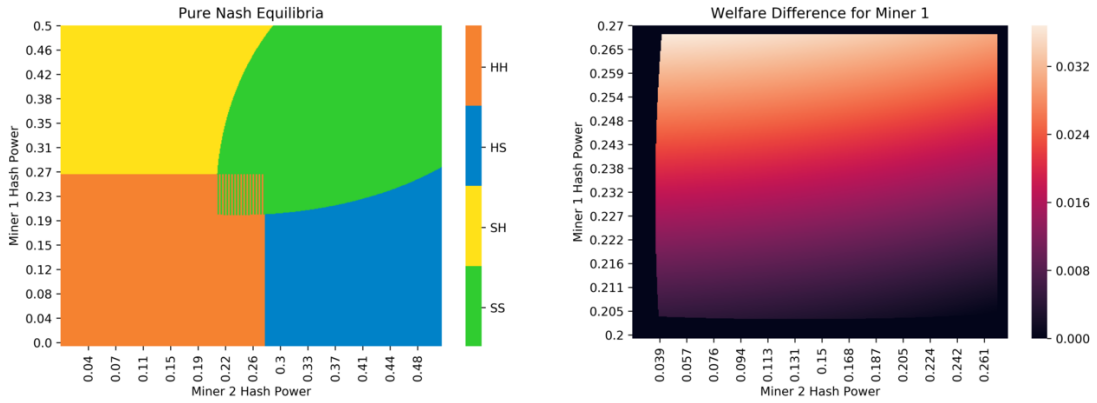

Figure 6.6: PNE types and the welfare surplus of SS over HH for m_1 when both are PNE.

roughly in the region $[0.2, 0.27]^2$, $\text{PNE}(G_\alpha) = \{HH \text{ and } SS\}$. For all of these hash rates, SS Pareto dominates HH as it results in more utility for both agents involved. As a concrete example, consider G_α for $\alpha = (0.24, 0.24)$ with utilities in Table 6.3. Clearly HH and SS are PNE in G_α, and the utility surplus between SS and HH is approximately 0.02 for m_1 and m_2.

Table 6.3: Example of G_α with HH and SS as PNE

$\alpha = (0.24, 0.24)$	HH	HS	SH	SS
U_1	0.24	0.24293956	0.23069139	0.25911617
U_2	0.24	0.23069139	0.24293956	0.25911617

The second image in Figure 6.6 focuses on $[0.2, 0.27]^2 \subset \mathcal{H}^2$ and visualises the difference in utility between SS and HH for m_1. The difference in utility for m_2 is symmetric since G_α is an anonymous game, meaning the role m_1 and m_2 can be interchanged.

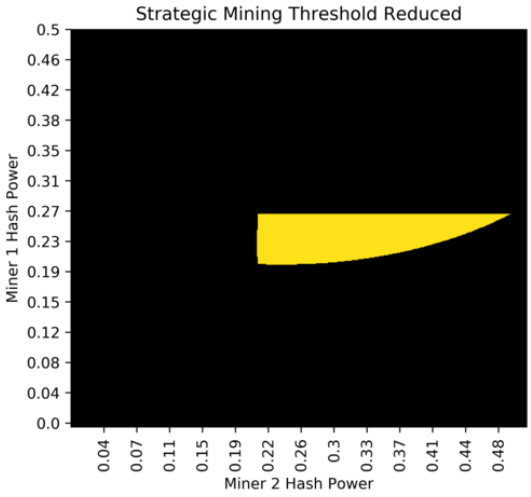

Figure 6.7: Hash rates where the profitability threshold of SSM is reduced.

Interestingly, there are hash rates $\alpha \in \mathcal{H}^2$ where SS is an equilibrium, yet SH is not profitable relative to HH for m_1. This means that the existence of another strategic miner can make mining with SSM profitable and stable for m_1 whereas this is not the case when m_1 with hash power α_1 is the only strategic miner in the system. For these α we say the profitability threshold of SSM has decreased. The set of $\alpha \in \mathcal{H}^2$ such that the profitability threshold of SSM decreases is graphed in Figure 6.7. Furthermore, there are hash rates in this region where SS is the only PNE, such as $\alpha = (0.235, 0.345)$ which leads to G_α with utilities in Table 6.4.

Table 6.4: Example G_α where SSM Profitability Threshold Decreases

$\alpha = (0.235, 0.345)$	HH	HS	SH	SS
U_1	0.235	0.22352621	0.22418585	0.23160125
U_2	0.345	0.37698013	0.34987697	0.42917647

The logical next step is to ask about mixed Nash equilibria in G_α, however the meaning of mixed strategies is not well-suited for selfish mining attacks. For example, what would the mixed strategy $0.2H + 0.8S$ represent? One interpretation could be a randomised commitment, where with probability 0.2 a miner commits to H and with probability 0.8 a miner commits to SSM. This however does not make much sense for selfish mining attacks, since their profitability takes time (due to adjustments in the block difficulty of the system), meaning that an opposing agent would have ample time to perform a best response to the realised commitment over the initial randomisation.

Another approach is to have $0.2H + 0.8S$ mean that a miner partitions his hash power into honest mining and SSM mining and commits to this partition henceforth. Although utilities of mixed strategies do not directly correspond to convex combinations of utilities, we use this approach to study an extended action space for miners.

6.7 Partition Games and Strong Stackelberg Equilibria

As mentioned at the end of the previous section, we also study incentives when miners are given a richer set of pure strategies beyond that of choosing between honest mining and SSM. In particular, we now allow a given miner with hash power α_i to partition his computational power into a portion following SSM and a portion using honest mining. Before continuing we also clarify notation: for $x, y \in \mathbb{R}^n$, we use $x \circ y$ to denote the Hadamard product of x and y.

Definition 6.2 (Two-player Partition Games)**.** *Suppose that $\alpha = (\alpha_1, \alpha_2) \in \mathcal{H}^2$ is a strategic hash distribution. We define the* Partition Game*, G_α^P, as a two-player game, where each player has the same action set $[0, 1]$, representing the proportion of their hash power dedicated to employing SSM. For a given pure strategy profile $s = (s_1, s_2) \in [0, 1]^2$, we define the utilities of G_α^P as follows:*

- $U_1(s_1, s_2) = s_1 R_{SSM}(s \circ \alpha)_1 + (1 - s_1)\frac{(1-s_1)\alpha_1}{1-s\cdot\alpha} R_{SSM}(s \circ \alpha)_3$.
- $U_2(s_1, s_2) = s_2 R_{SSM}(s \circ \alpha)_2 + (1 - s_2)\frac{(1-s_2)\alpha_2}{1-s\cdot\alpha} R_{SSM}(s \circ \alpha)_3$.

In Figure 6.8, for $\alpha = (0.46, 0.25)$ we graph the pure strategy utilities of m_1 and m_2 as a function of $s \in [0, 1]^2$. The most glaring observation is that for fixed s_{-i}, $U_i(s_i, s_{-i})$ is a convex function of s_i, attaining local maxima at $s_i = 0$ and $s_i = 1$. This is clear from the fact that the blockchain eventually has one common history, so both sides of a miner's partition inherently compete with one another.

Game theoretically, this means best responses for any m_i are always from the set $\{0, 1\}$. Immediately, this tells us that the set of pure Nash equilibria of G_α^P are the same as those in G_α, since G_α^P restricted to pure strategy profiles in $\{0, 1\}^2$ is isomorphic to G_α (recall that $H \cong 0$ and $S \cong 1$ in G_α). It may thus seem the augmented strategy space of G_α^P buys us nothing. However, if we treat G_α^P as a leadership game, where m_1 gets to commit to a pure strategy, s_1, to which m_2 retaliates, then we get a different story.

To formally treat G_α^P as a leadership game, we let m_1 be the leader and m_2 the follower. For a given pure strategy $s_1 \in [0, 1]$ of m_1, we let $BR(s_1)$ denote the best response m_2 has to s_1. Since we have observed that best responses for any m_i are always from the

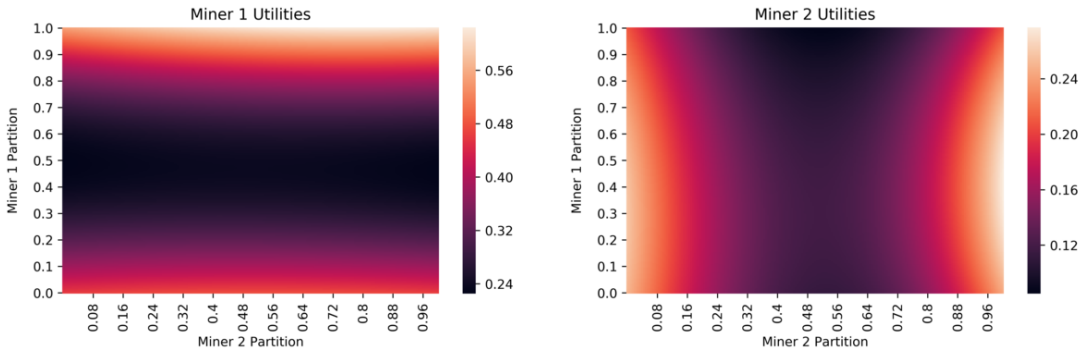

Figure 6.8: Utilities in G_α^P for $\alpha = (0.46, 0.25)$.

set $\{0, 1\}$, it follows that $BR(s_1) = \text{argmax}_{x\in\{0,1\}} U_2(s_1, x)$. If $U_2(s_1, 0) = U_2(s_1, 1)$, then we let $BR(s_1) = \text{argmax}_{x\in\{0,1\}} U_1(s_1, x)$, so that m_2 breaks ties in favour of m_1. The value of commitment s_1 for m_1 is denoted by $v_1(s_1) = U_1(s_1, BR(s_1))$ and for the value of commitment s_1 for miner 2 is denoted by $v_2(s_1) = U_2(s_1, BR(s_1))$.

In a leadership game, a common solution concept is that of a *Strong Stackelberg Equilibrium (SSE)*, which is a strategy pair (s_1^*, s_2^*) such that $s_1^* \in \text{argmax}_x v_1(x)$ and $s_2^* = BR(s_1^*)$. This can be seen as a subgame perfect equilibrium of G_α^P, or the optimal commitment under v_1. Furthermore, we let $\text{SSE}(G)$ denote the SSE of a game G.

In Figure 6.9 we graph (optimal) commitment values for m_1 at the SSE of G_α^P for different values of $\alpha \in \mathcal{H}^2$. Furthermore, we graph the value of these optimal commitments when compared to utility players obtain at their respectively optimal PNE of G_α at the given hash rate.

6.7.1 Non-trivial SSE

Since G_α^P can be seen as an augmented action space to G_α, we categorise $\alpha \in \mathcal{H}^2$ depending on how the sets $\text{PNE}(G_\alpha) = \text{PNE}(G_\alpha^P)$ and $\text{SSE}(G_\alpha^P)$ compare.

Definition 6.3 (Commitment/SSE Types)**.** *For every $\alpha \in \mathcal{H}^2$ we associate a* commitment type *denoted $com(\alpha) \in \{0, 1, 2, 3\}$ defined as follows:*

- *If $SSE(G_\alpha^P) = PNE(G_\alpha)$, then we define $com(\alpha) = 0$.*
- *If $SSE(G_\alpha^P) \subset PNE(G_\alpha)$, then we define $com(\alpha) = 1$.*
- *If $SSE(G_\alpha^P) \not\subset PNE(G_\alpha)$, and $\exists s^* \in SSE(G_\alpha^P)$ such that $s_1^* \in \{0, 1\}$, then we define $com(\alpha) = 2$.*
- *If $SSE(G_\alpha^P) \not\subset PNE(G_\alpha)$, and $\nexists s^* \in SSE(G_\alpha^P)$ such that $s_1^* \in \{0, 1\}$, then we define $com(\alpha) = 3$.*

If $com(\alpha) = 0$ we say $\alpha \in \mathcal{H}^2$ gives rise to a trivial commitment and that the collection of SSE in G^P_α are trivial. Accordingly, if $com(\alpha) \neq 0$, we say α gives rise to a non-trivial commitment and the collection of SSE in G^P_α is non-trivial. Furthermore, we also say that if $\alpha \in \mathcal{H}^2$ is such that $com(\alpha) = i$, then all $s^* \in \text{SSE}(G^P_\alpha)$ are of type i as well. In the two-miner scenario, we make the following observations about $\alpha \in \mathcal{H}^2$ with non-trivial commitment types:

- $com(\alpha) = 1$ occurs at hash values such that the PNE of G_α are HH and SS. m_1 commits to S to nudge the system to converge to the SS equilibrium which Pareto-dominates HH in G_α.

- $com(\alpha) = 2$ occurs at hash rates where there is one SSE of G^P_α, $s^* = (s^*_1, s^*_2) \in \{0, 1\}^2$, yet s^* does not correspond to a PNE of G_α. s^* is unstable in G_α from the perspective of m_1, who would prefer deviating from s_1 when pitted against s_2. These SSE make use of the sequentiality of G^P_α but not of the extended action space given by partitioning.

- $com(\alpha) = 3$ occurs at hash rates such that HS is the only PNE of G_α, but where α is close to the region in $\mathcal{H}^2$ where SS arises as the sole PNE of G_α. At these values, m_2 only slightly prefers SH to SS, hence m_1 can bait m_2 into playing S by reserving a small portion of hash power to mine honestly.

For any non-trivial SSE, $v_1(s^*_1)$ is lower bounded by the lowest-utility m_1 obtains amongst PNE in G_α. On the other hand, if $com(\alpha) = 1, 3$, the SSE of G^P_α are such that $v_1(s^*_1)$ is strictly greater than the highest utility m_1 obtains amongst PNE in G_α. This strict surplus in utility is visible in the latter graphs of Figure 6.9, and we can see that these non-trivial commitments also benefit m_2 in spite of being the follower.

6.7.2 Plots of Non-Trivial SSE by Type

We now focus on plotting optimal $\mathcal{H}^2$ that exhibit SSE of types 1, 2 and 3. For SSE of type 1, it suffices to look at Figure 6.6 and Table 6.3 for visualisation of the benefit of SSE over PNE (Since it is just the difference in welfare between PNE in this case).

As for SSE of type 2, these are plotted in more detail in Figure 6.10. For these values of α, we can see that HH is the only PNE in G_α, but SS is the SSE of G^P_α, which is forcibly unstable in the one shot game, G_α, as m_1 prefers HS to SS. Table 6.5 shows the utilities for G_α at a specific value of α exhibiting this behaviour. Note that in this example, the leader, m_1, has a hash rate of $\alpha_1 = 0.2$, at which normally they would not be incentivised to unilaterally employ SSM in the one-shot SSM game. The power to commit makes SSM viable at smaller hash rates than in the one-shot game.

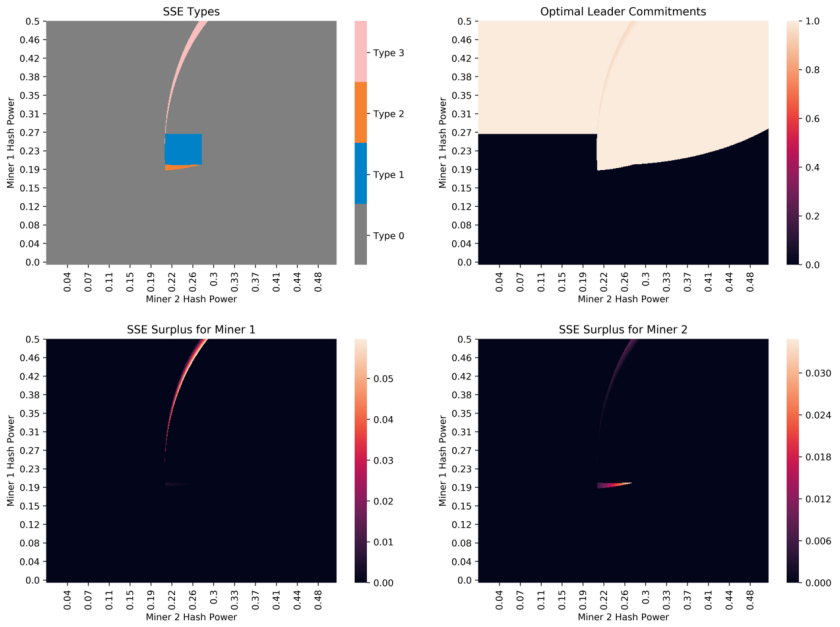

Figure 6.9: SSE types for m_1, optimal commitments for m_1, and relative surplus of SSE against best PNE for m_1 and for m_2 respectively.

Table 6.5: Example G_α where the SSE in G_α^P is of type 2

$\alpha = (0.2, 0.225)$	HH	HS	SH	SS
U_1	0.2	0.20352746	0.18016529	0.20179681
U_2	0.225	0.21133109	0.23057851	0.23905979

Figure 6.11 focuses on hash rates where SSE are of type 3. Furthermore, Figure 6.12 looks specifically at $\alpha = (0.431, 0.239)$, which is a hash rate such that G_α^P has an SSE of type 3, and graphs utilities and best responses as a function of the leader commitment in G_α^P. This gives a better way of visualising how $s_1 = 0.98$ is an optimal commitment where m_2 is rendered indifferent between S and H.

6.8 Game-theoretic Formalism for $M > 2$ Strategic Miners

Our analysis from Section 6.4.2 extends in a straightforward fashion to when there are $M > 2$ strategic miners. Consequently, for any hash distribution $\alpha \in \mathcal{H}^M$, we can compute $R_{SSM}(\alpha) \in [0,1]^{M+1}$, the revenue ratio of all M strategic miners and all other honest miners, when all strategic miners of hash power α_i employ SSM. In this section, we

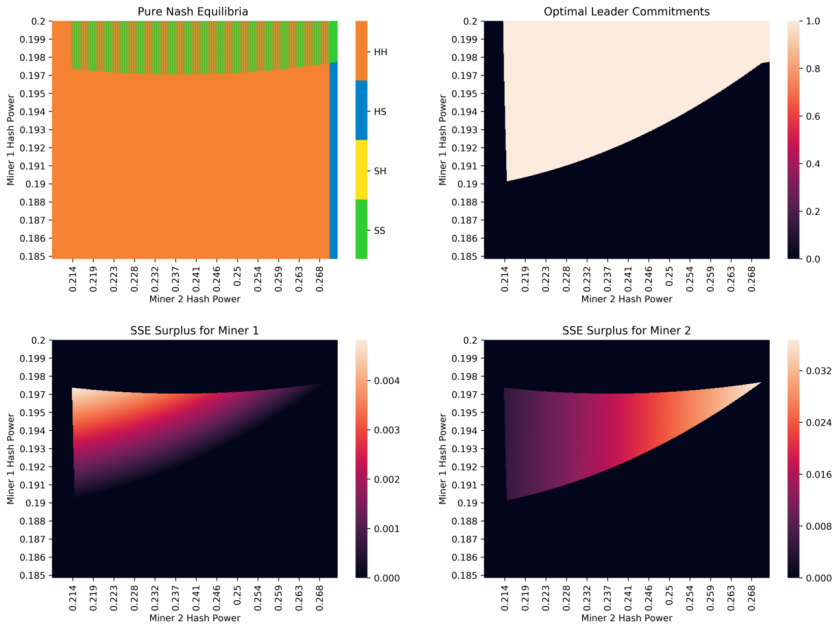

Figure 6.10: Optimal Commitments for m_1, as well as SSE surplus against best PNE for m_1, and for m_2 respectively in the region $[0.185, 0.2] \times [0.21, 0.27]$. This region exhibits SSE of type 2.

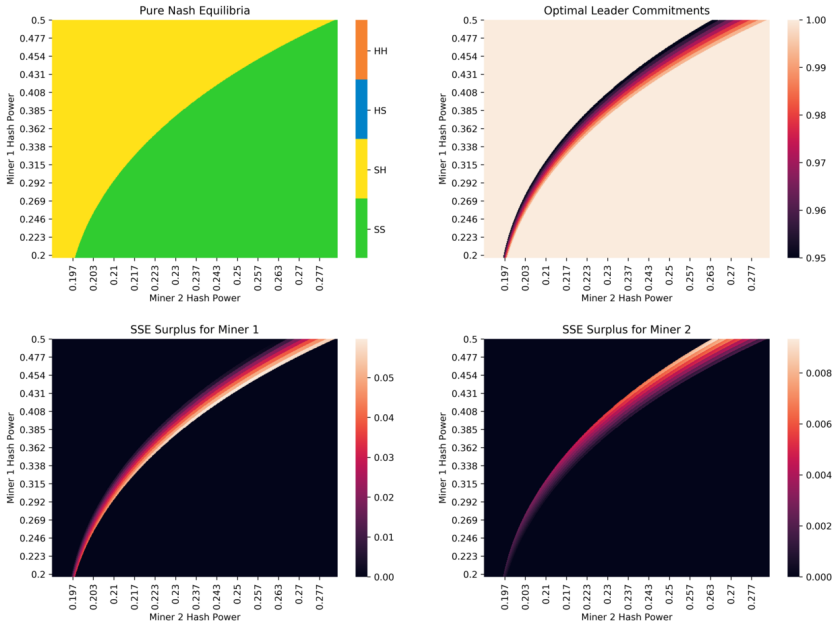

Figure 6.11: Optimal Commitments for m_1, as well as SSE surplus against best PNE for m_1, and for m_2 respectively in the region $[0.2, 0.5] \times [0.19, 0.28]$. This region exhibits SSE of type 3.

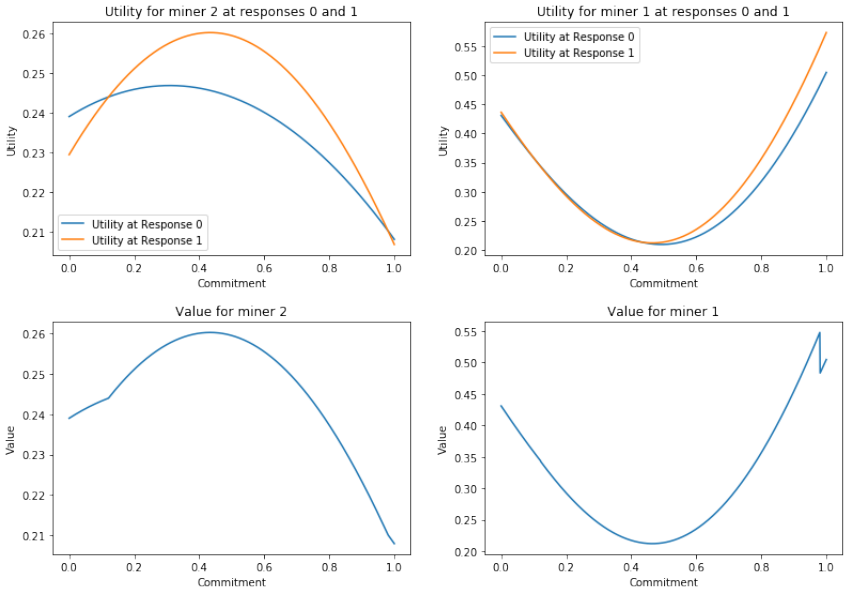

Figure 6.12: Partition Game Analysis for $\alpha = (0.431, 0.239)$. The top left image plots follower utilities when playing H or S against a leader commitment partition. The bottom left image plots follower utility when best responding to a leader commitment. The best response at a given commitment dictates which of the two utilities the leader obtains in the top right plot. Putting everything together, the bottom right plot gives the value of a leader commitment (for the leader) as a function of their commitment. Note how this function is maximised at approximately 0.98, where the follower is indifferent between H and S.

extend the game-theoretic formalism of Section 6.4.2 to to study incentives when $M > 2$ strategic miners interact.

6.8.1 To SSM or not to SSM in the Multiplayer Setting

We recall that Section 6.4.2 introduced SSM games, a family of binary action games G_α that governed the incentives behind choosing to employ SSM or honest mining. We extend this game to the multiplayer setting in a natural way. Suppose that $\alpha \in \mathcal{H}^M$ is a hash rate of all strategic miners. Once again, we let $R_{SSM}(\alpha) \in [0,1]^{M+1}$ be the revenue ratios of all miners. Just as before, $R_{SSM}(\alpha)_i$ is the relative revenue of m_i.

Definition 6.4 (Multi-player SSM Games)**.** *For every $\alpha \in \mathcal{H}^M$, we define the* SSM Game*, G_α as a M-player binary action game. Each strategic miner has a binary action set $\{H, S\}$, where H represents mining honestly and S represents employing SSM. For convenience, we associate this action space with $\{0,1\}^M$, where action 0 denotes honest mining and action 1 denotes employing SSM. In order to specify utilities, we suppose that $x \in \{0,1\}^M$ is a pure action profile such that $x_i = 1$ and $x_j = 0$:*

- $U_i(x) = R_{SSM}(\alpha \circ x)_i$.
- $U_j(x) = \frac{\alpha_j}{1-\alpha \cdot x} R_{SSM}(\alpha \circ x)_{M+1}$.

Penalising Coalitions

In Section 6.4.2 we explored scenarios where an agent might be unilaterally incentivised to use SSM, but a second larger agent can retaliate by employing SSM to make the original agent worse off than when everyone mines honestly. In the multiplayer setting, any subset of agents can retaliate in a similar fashion, thus we formally define what constitutes a penalising coalition. In what follows, we use the notation $\vec{\chi}_C \in \{0,1\}^M$ to denote an indicator vector for a subset $C \subset [M]$.

Definition 6.5 (Penalising Coalition)**.** *Suppose that $\alpha \in \mathcal{H}^M$ is a distribution of hash power amongst M strategic miners. We say that $C \subset \{2, ..., M\}$ is a penalising coalition for miner 1 if the following hold:*

- $U_1(\vec{\chi}_1) > U_1(\vec{0})$.
- $U_i(\vec{\chi}_{1 \cup C}) > U_i(\vec{\chi}_{1 \cup C \setminus i})$ *for all* $i \in C$.
- $U_1(\vec{\chi}_{1 \cup C}) < U_1(\vec{0})$.

Furthermore, we say that C incurs a penalty of $U_1(\vec{0}) - U_1(\vec{\chi}_{1 \cup C})$ on miner 1 when retaliating.

The first condition ensures that miner 1 has a unilateral incentive to deviate and employ SSM. The second condition ensures that each miner in the penalising coalition is better off retaliating than defecting from the retaliation (ensuring retaliation is in a loose sense a credible threat), and finally the third condition ensures that miner 1 is worse off when being retaliated against than when everyone is honest.

6.8.2 Partition Games in the Multiplayer Setting

In Section 6.7, we introduced the notion of a partition game, G_α^P, which extended the action space of G_α to allow miners to partition their hash power into honest mining and employing SSM. This definition extends naturally to the M-player case.

Definition 6.6 (Multi-player Partition Games)**.** *Suppose that $\alpha \in \mathcal{H}^M$ is a hash distribution for M strategic miners. We define the* Partition Game*, G_α^P, as a M-player game, where each player has the same action set $[0,1]$, representing the proportion of their hash power dedicated to employing SSM. For a given pure strategy profile $s \in [0,1]^M$, we define the utility of the i-th player in G_α^P as follows:*

- $U_i(s) = s_i R_{SSM}(s \circ \alpha)_i + (1 - s_i)\frac{\alpha_i(1-s_i)}{1-\sum \alpha_i s_i} R_{SSM}(s \circ \alpha)_{M+1}$.

Optimal Commitments in G_α^P

As in the $M = 2$ miner case, for any miner i, every action $s_i \in (0,1)$ is dominated by either $s_i = 0$ or $s_i = 1$ if G_α^P is treated as a one shot game. The reason for this is that partitioning hash power results in unnecessary self competition, hence it will never be a best response to fixed opponent strategies. Consequently, the PNE of G_α^P as a one shot game are identical to the PNE of G_α.

On the other hand, we can once again treat G_α^P as a full information sequential game where m_1 commits to a strategy and all other $M-1$ players react. The subgame perfect Nash equilibria (SGPNE) of this game are generalisations of the Stackelberg equilbria of Section 6.7. The most subtle issue with generalising SSE however arises in tie-breaking. The assumption in SSE for two player games is that the follower will break ties in favour of the leader. This is a fair assumption in the two-player setting, because it is often the case that commitments that lead to indifference in responses are of lower measure than those that invoke unique best responses. For this reason a leader can commit to strategies in an arbitrarily small neighbourhood of an SSE to ellicit the desired best response in the case of a tie for the follower.

In the multi-player setting however, it can be the case that a non-trivial neighbourhood of leader commitments give rise to subgames with multiple PNE. For this reason it may

be unfeasible to assume that follower agents converge to a PNE that maximises the welfare of the leader, as there is nothing in the power of the leader to even approximately guarantee this behaviour. For this reason, we take a pessimistic approach to SGPNE of G_α^P. In particular, we assume that for a leader commitment, all other agents will settle on a PNE that minimises welfare for the leader. To be precise, for a given pure strategy $s_1 \in [0,1]$, we let $G_\alpha^P(s_1,-)$ denote the $(M-1)$-player subgame for miners $2,...,M$ conditioned on miner 1 committing to s_1. Furthermore, we let $WSN(s_1)$ (Worst sub-Nash) be the lowest utility pure Nash equilibrium of $G_\alpha^P(s_1,-)$ for miner 1. The value of commitment s_1 in the leadership game G_α^P for miner 1 is $v_1(s_1) = U_1(s_1, WSN(s_1))$, and for any other miner $i = 2,...,M$, $v_i(s_1) = U_i(s_1, WSN(s_1))$. We call the family of all such pure strategy profile the collection of *Pessimistic Sub-game Perfect Nash Equilibria*, (P-SGPNE). In particular, we are interested in values of α where the set of P-SGPNE of G_α^P result in strictly larger welfare for m_1, implying that either the possiblity of commitment or partitioining strictly benefits m_1 in the worst case.

In a similar fashion to the two-player case, we study how different values of $\alpha \in \mathcal{H}^M$ give rise to different P-SGPNE(G_α^P) vs PNE(G_α) = PNE(G_α^P).

Definition 6.7 (Multiplayer Commitment/ SGPNE Types)**.** *Suppose that $\alpha \in \mathcal{H}^M$, we classify its commitment type, com(α), depending on the relationship between the sets P-SGPNE(G_α^P) and PNE(G_α)*

- *If P-SGPNE(G_α^P) = PNE(G_α), then com(α) = 0.*
- *If P-SGPNE(G_α^P) $\subset$ PNE(G_α), then com(α) = 1.*
- *If P-SGPNE(G_α^P) $\not\subset$ PNE(G_α) and $\exists s^* \in$ P-SGPNE such that $s_1^* \in \{0,1\}$, then com(α) = 2*
- *If P-SGPNE(G_α^P) $\not\subset$ PNE(G_α) and $\nexists s^* \in$ P-SGPNE such that $s_1^* \in \{0,1\}$, then com(α) = 3.*

As in the two-player case, if $com(\alpha) = 0$ we say $\alpha \in \mathcal{H}^M$ gives rise to a trivial commitment and that the collection of P-SGPNE in G_α^P are trivial. Accordingly, if $com(\alpha) \neq 0$, we say α gives rise to a non-trivial commitment and the collection of P-SGPNE in G_α^P is non-trivial. Furthermore, we also say that if $\alpha \in \mathcal{H}^M$ is such that $com(\alpha) = i$, then all $s^* \in$ P-SGPNE(G_α^P) are of type i as well.

6.9 Visualising Incentives of $M = 3$ Strategic Miners

In order to visualise results for $M = 3$ miners, we fix the hash rate of the third player, α_3 and repeat our analysis for Section 6.4.2 when α_1 and α_2 are allowed to vary. We observe qualitative difference in the family of games G_α for four different regions of α_3 values: $R_1 = [0, 0.17]$, $R_2 = [0.175, 0.203]$, $R_3 = [0.208, 0.27]$ and $R_4 = [0.274, 0.5]$.

6.9.1 Pure Nash Equilibria in G_α

As mentioned in the previous section, our results show four main regimes of results as a function of α_3. In terms of PNE, when $\alpha_3 \in R_1$, H strictly dominates S for miner 3, hence the three player games G_α and G_α^P reduce to a two player game conditioned on player 3 playing H. For $\alpha_3 \in R_2$, we see the emergence of SSS as a PNE near the centre of the hash space, and the size of this region grows as a function of α_3. For $\alpha_3 \in R_3$, SSS is still a PNE for central values of α, however we see the emergence of distinct regions where SHS and HSS are PNE. Finally, when $\alpha_3 \in R_4$, S strictly dominates H for player 3, and once again G_α and G_α^P reduce to subgames conditioned on miner 3 playing S.

In Figure 6.13, we visualise this phenomenon by picking representative values of α_3 in R_1, R_2, R_3 and R_4 and graphing regions where distinct PNE occur in G_α as well as the difference in welfare between the best and worst PNE for each player respectively.

6.9.2 SSM Profitability Threshold Diminished

As in the two-player case, we find that there are hash rates where m_1 is not unilaterally incentivised to employ SSM, yet there exist equilibria where m_1 employs SSM. In Figure 6.14 we visualise the hash rates where this happens for all R_i relevant regions of α_3 values. In particular, for α_3 values in R_2, R_3 and R_4, we see that the emergence of SSS as a PNE can occur when m_1 has much smaller hash power than the 0.26795 necessary to make SSM profitable unilaterally.

6.9.3 Optimal Commitments

As mentioned in Section 6.8, we treat G_α^P as a full information sequential game where m_1 commits to a strategy and all other miners subsequently act. We recall that for a given pure strategy commitment $s_1 \in [0, 1]$ for m_1, the worst Nash equilibrium of the resulting subgame $G_\alpha(s_1, -)$ for m_1 (so in terms of U_1) is denoted by $WSN(s_1)$. In addition, $v_i(s_1) = U_i(s_1, WSN(s_1))$ for $i = 1, ..., M$, denotes the utility obtained by each miner at $(s_1, WSN(s_1))$.

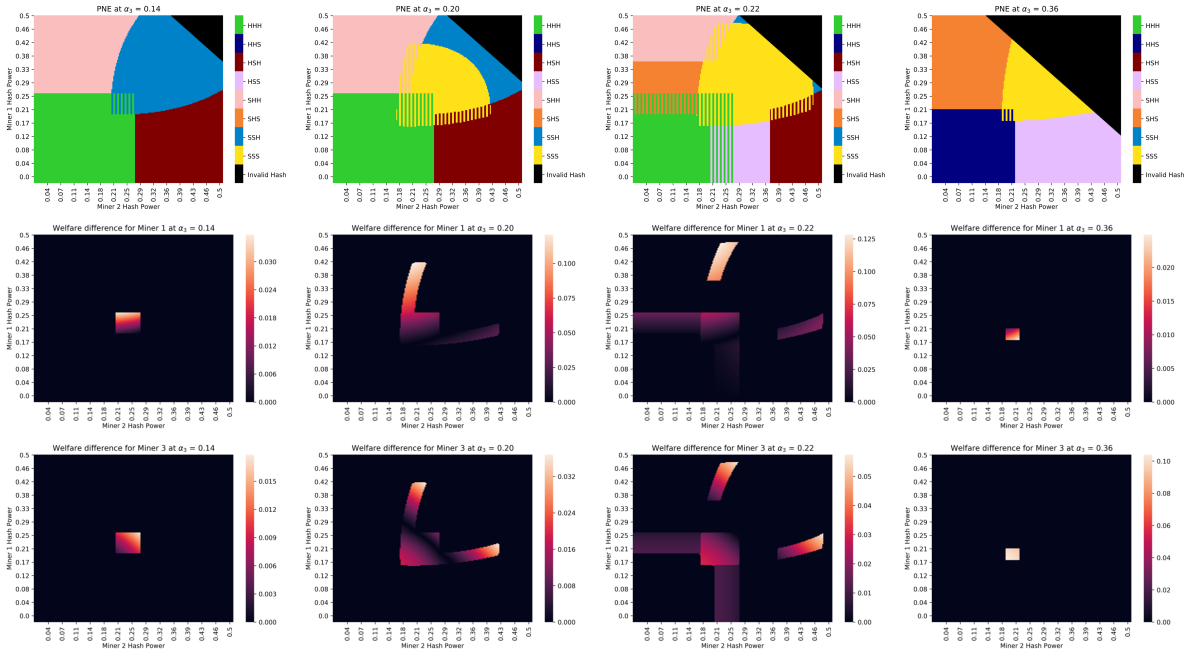

Figure 6.13: PNE for $\alpha_3 \in \{0.14, 0.2, 0.22, 0.36\}$. For the areas that have multiple PNE, the difference in welfare at the best PNE and worst PNE for miner 1 and miner 3 are mapped in the second and third rows respectively. Since G_α is an anonymous game, the difference in welfare for miner 2 is the same as that of miner 1 reflected about the axis $y = x$.

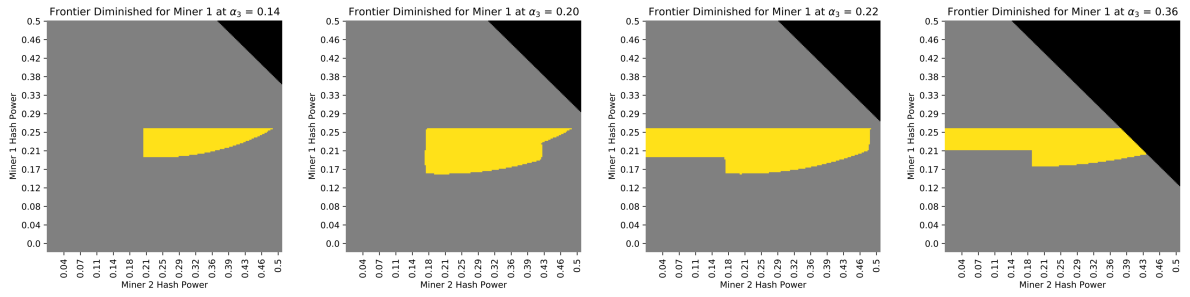

Figure 6.14: Profitability Threshold Diminished for $\alpha_3 \in \{0.14, 0.2, 0.22, 0.36\}$

For $\alpha_3 \in R_1, R_2, R_3, R_4$, we look at what values of s_1 optimise $v_1(s_1)$ as optimal commitments from the leader of G^P_α, m_1. As mentioned in Section 6.8, any $s^* = (s^*_1, WSN(s^*_1))$ that optimises v_1 is necessarily a (pessimistic) subgame perfect Nash equilibrium. In Figure 6.15 we plot these optimal commitments with fixed α_3 as a function of (α_1, α_2). Furthermore, we also plot $com(\alpha)$ as per Definition 6.7, and the subsequent surplus between v_1 at the aforementioned pessimistic SGPNE and the worst lowest utility PNE for m_1. Similar observations can be made as in the two-player case of Section 6.7:

- When $com(\alpha) = 0$, pessimistic SGPNE of G^P_α are identical to PNE of G_α, so the ability to partition and the ability to commit to strategies do not give m_1 an undue advantage in the worst case.

- When $com(\alpha) = 1$ it is generally the case that G_α has multiple PNE, and the commitment of m_1 "nudges" other players to a PNE that Pareto-dominates the worst PNE in G_α.

- The only exception to the previous observation is the left-most region of $com(\alpha) = 1$ when $\alpha_3 = 0.2$ is fixed (second column of Figure 6.15). In this area, the optimal commitment for m_1 is $s_1 = 0$. In response to this, the subgame $G_\alpha(0, -)$ only has HH as a PNE. As a consquence, the only pessimistic SGPNE at these α values is $(0, 0, 0)$, yet both $(0, 0, 0)$ and $(1, 1, 1)$ are PNE in G_α. The reason for this however, is that if we consider the commitment $s_1 = 1$ (i.e. m_1 employing SSM), then $G_\alpha(1, -)$ actually has two PNE: HH and SS. The worst of these two equilibria however is HH, and thus the strategy profile $(1, WSN(1)) = (1, 0, 0)$, which is strictly worse than $(0, 0, 0)$ fpr m_1.

- When $com(\alpha) = 2$ there exist pessimistic SGPNE, $s^* = (s^*_1, WSN(s^*_1)) \notin \text{PNE}(G_\alpha)$ such that $s_1 \in \{0, 1\}$ and s^*_1 is not a best response to $WSN(s^*_1)$ for m_1.These non-trivial commitments make use of sequentiality of G^P_α but not of the augmented action space granted by partitioning.

- When $com(\alpha) = 3$, m_1 has enough hash power that $\text{PNE}(G_\alpha)$ only has strategy profiles that exemplify m_2 and m_3 being disincentivised to use SSM. That being said, at these values of α, m_2 and m_3 are almost indifferent between employing SSM and honest mining (hence the reason $com(\alpha) = 3$ occurs along boundaries of where $\text{PNE}(G_\alpha)$ changes values), hence m_1 can bait them into employing SSM by judiciously giving away some hash power to honest mining in a partition.

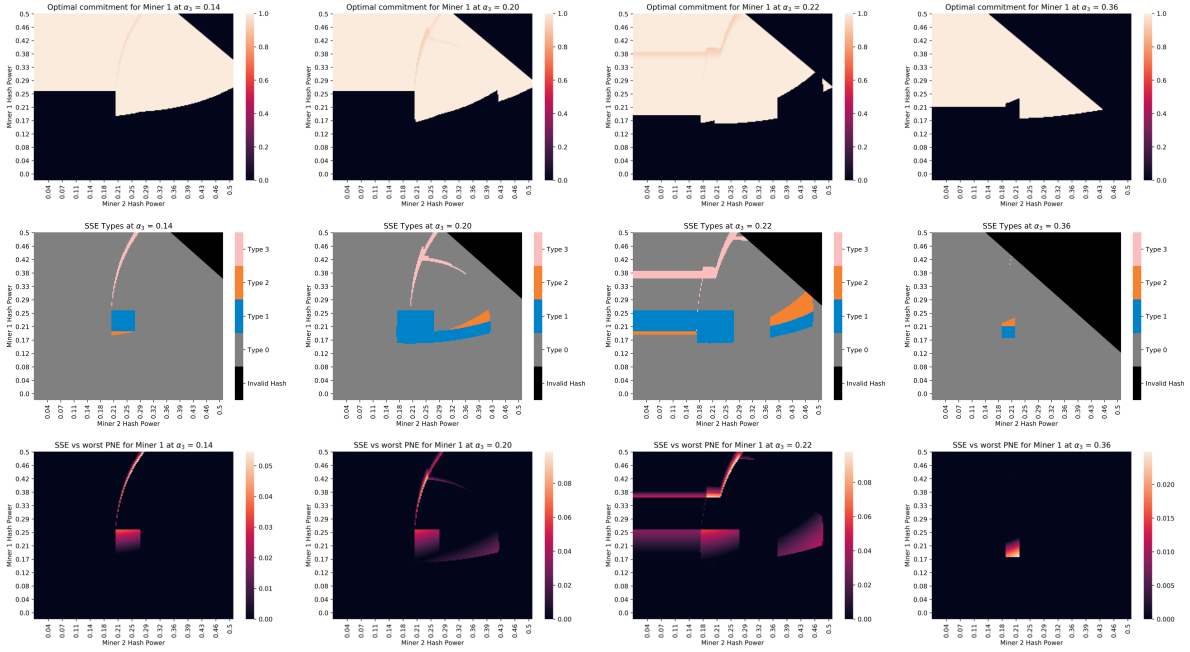

Figure 6.15: Optimal m_1 commitment for $\alpha_3 \in \{0.14, 0.2, 0.22, 0.36\}$, commitment types, and utility surplus in P-SGPNE vs. worst PNE for m_1.

6.9.4 Penalising Coalitions

In Figure 6.16 we plot hash rates where there exist penalising coalitions against miner 1 along with the smallest given penalty they can incur on miner 1. Furthermore, in the top row of the plot, we specify precisely which coalitions $C \subset [2, 3]$ satisfy the conditions of Definition 6.5. The plots show that for $\alpha_3 \in \{0.14, 0.36\}$ there is only one kind of penalising coalition ($C = \{2\}$ or $C = \{3\}$ respectively), but for $\alpha_3 \in \{0.2, 0.22\}$, $C = \{2\}, \{3\}$ and $\{2, 3\}$ are all penalising coalitions at different hash rates and for some values of α.

6.10 Visualising Incentives of $M > 3$ Strategic Miners

When $M > 3$ it becomes difficult to visualise how aspects of G_α and G_α^P precisely vary with α. That being said, we do find very similar structures as in the $M = 2$ and $M = 3$ case, such as: penalising coalitions, existence of PNE, and for some regions multiple PNE, in G_α, non trivial commitments in G_α^P, and finally, hash rates α where the SSM profitability threshold decreases with the existence of other strategic miners. We expand upon this final point to specifically see how the number of strategic miners M affects the profitability threshold of SSM.

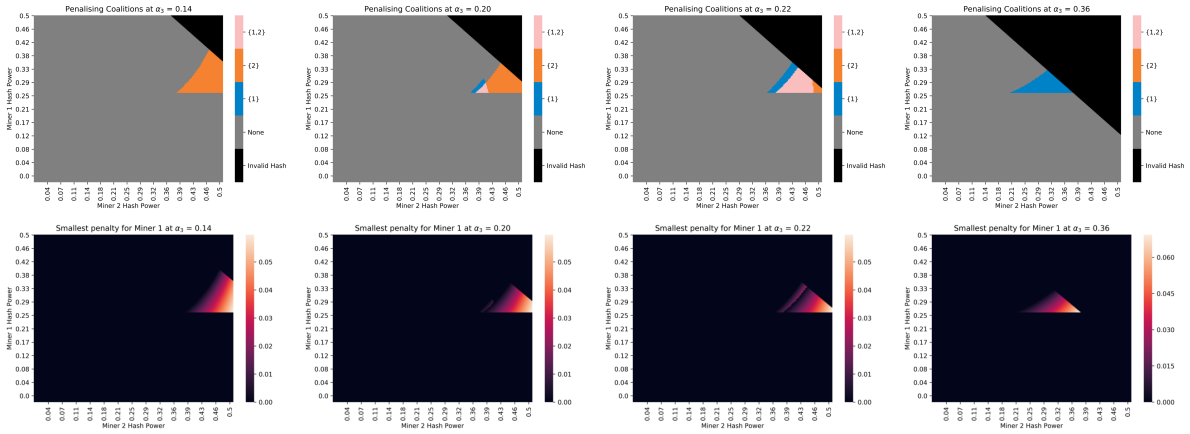

Figure 6.16: All Penalising Coalitions for miner 1 when $\alpha_3 \in \{0.14, 0.2, 0.22, 0.36\}$, and the smallest penalty they incur.

6.10.1 Decreasing SSM Profitability Threshold

To study the effect of the number of miners on the profitability threshold of SSM, we define the following:

Definition 6.8 (Uniform Profitability Threshold for SSM)**.** *For $M \geq 1$ miners we say the* uniform profitability threshold for SSM *is the smallest $\eta \in [0, 1]$ such that $\alpha = \eta\vec{1} \in \mathcal{H}^M$ and $\vec{1} \in PNE(G_\alpha)$ (all players employing SSM is a PNE in G_α).*

With our methods from Section 6.8, we can approximate the uniform SSM profitability threshold for various values of M. In particular, Figure 6.17 shows these threshold values for $M = 1, ..., 8$. Furthermore, the second plot takes the uniform SSM profitability threshold, η, and for $\alpha = \eta\vec{1}$, computes the utilities of both $\vec{0}$ and $\vec{1}$ which are both PNE in G_α. Interestingly, for $M = 1, .., 8$, not only does the uniform profitability threshold decrease as a function of M, but all miners employing SSM is a PNE that Pareto dominates all miners being honest. These results thus show that the presence of multiple strategic miners may have more of an impact on the stability of Bitcoin than previously thought.

6.11 Conclusion and Future Directions

In this chapter we have described a specific miner strategy, semi-selfish mining (SSM) that is a truncated variant of Selfish Mining (SM). SSM has the benefit of being a profitable strategy for large enough miners (in the same way as SM), and also structured enough for us to explicitly solve for relative revenues when more than one strategic miner employs SSM. With this in hand, we have been able to use a game-theoretic lense to glean some information on miner incentives when more than one miner is strategic within the bitocin system.

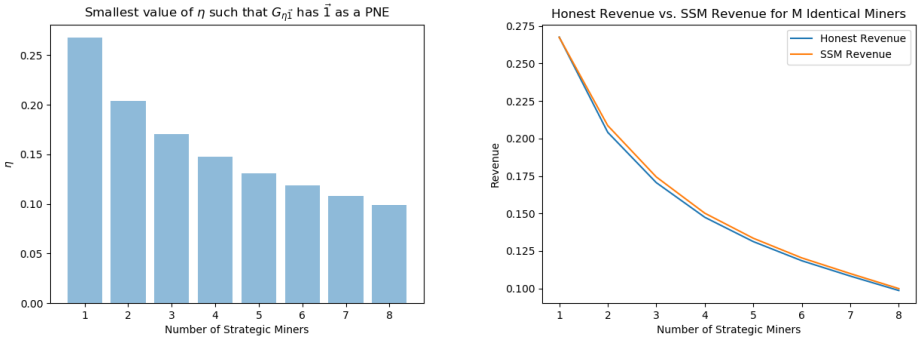

Figure 6.17: Upper bounds on the uniform profitability threshold for SSM as a function of the number of strategic miners. We also plot the welfare of $\vec{1}$ (all SSM) versus $\vec{0}$ (all honest).

In particular, for any $\alpha \in \mathcal{H}^M$, we define the SSM game G_α which governs strategic miner incentives in choosing to employ SSM or mine honestly, and the partition game G_α^P, which extends the action space of G_α to allow miners to partition their hash power between honest mining and SSM. For $M > 1$ strategic miners we find the following main takeaways from studying G_α and G_α^P:

- All $\alpha \in \mathcal{H}^M$ seem to lead to G_α with pure Nash equilibria. Furthermore, there are regions in $\mathcal{H}^M$ such that G_α has multiple PNE.
- A single miner might prefer to use SSM over honest mining in G_α, but there can exist a coalition of miners who may retaliate against this action and punish the original SSM miner into receiving less utility than their hash power.
- Though the set of PNE in G_α^P is identical to those of G_α, when treating G_α^P as a sequential game leads to non-trivial commitments, some of which involve a miner employing SSM even though SSM is not rational in the one-shot SSM game.
- Finally, there exist hash rates, $\alpha \in \mathcal{H}^M$ such that m_1 does not unilaterally prefer to employ SSM, but some PNE of G_α includes m_1 employing SSM, effectively reducing the profitability threshold of SSM and consequently affecting the stability of Bitcoin.

The action spaces in G_α and G_α^P may seem limited due to the fact that they only interpolate between honest mining and SSM, but there is nothing barring a variant G_α and G_α^P from studying the choice of employing other subversive mining strategies over honest mining. In fact, G_α and G_α^P can be defined by using empirical estimates to steady state payoffs instead of closed form solutions, which could glean some information into how mining dynamics change when a larger palette of subversive strategies is available to

interdependent strategic miners. In fact, G_α^P could be extended so that the action space of miners is no longer simply partitioning mining power between honest mining and SSM, but any partition of mining power amongst a given list of subversive mining strategies.

In addition, the fact that penalising coalitions exist hints at the possibility of modelling such structures in a repeated game framework. The issue of course comes in modelling how much utility a penalising coalition gains in maintaining everyone honest, but there could be interesting subgame perfect Nash equilibria in an appropriate model. Finally, along the same vein of penalising coalitions, there is also scope for a more fine-grained cooperative game theoretic analysis of SSM and Partition games.

Chapter 7

RPPLNS: Pay-per-last-N-shares with a Randomised Twist

In this chapter we present a novel twist to the already popular "Pay-per-last-N-shares" (PPLNS) mining pool scheme used by the majority of the Bitcoin network. By suitably randomising PPLNS we are able to maintain its strengths (fairness, variance reduction, robustness to pool hopping) while reducing the underlying memory usage of the protocol and proving robustness guarantees against a richer class of strategic mining than before.

7.1 Introduction

In Bitcoin's proof-of-work (PoW) consensus protocol, the miner is essential. These agents take upon themselves the task of maintaining the ledger of transactions that form the foundation of Bitcoin. This upkeep is not done out of good will however, as the judicious design of Bitcoin pays miners for their efforts, with the aim of incentivising honest behaviour. Bitcoin dynamically adjusts its difficulty, and as mining becomes more specialised, miners find that the variance in their rewards is undesirably high (indeed mining a single block on a small CPU could take years in expectation). For this reason, mining pools have emerged as a way of reducing payoff variance, as miners pool together resources and distribute earnings accordingly.

Given the existence of pools, however, the choice of which payment protocol to use becomes an important mechanism design problem, as any such choice directly affects pool miner behaviour. As of now, the most popular pool payment protocol is Pay-per-last-N-shares (PPLNS), in which a pool manager maintains a queue of the N most recently reported shares (near misses to the hash threshold of the protocol), and if a block is found by the pool, its reward is distributed evenly amongst the miners who own those N most recent shares. The popularity of PPLNS can be attributed to the following properties:

- Fairness: PPLNS miners earn the same block reward in expectation than solo mining.
- Variance reduction: PPLNS miner earnings have lower variance than in solo mining.
- Robustness against Pool-hopping: Unlike other payment schemes (such as proportionally paying per share), there are no "beneficial" states of the pool where miners are better off expending their computational resources in another PPLNS pool.

In this work, we present a novel stochastic twist to PPLNS which we call Randomised PPLNS (RPPLNS). We show that RPPLNS enjoys the same aforementioned benefits of PPLNS, in addition to the following:

- Robustness against a richer class of strategic mining.
- An exponential reduction in the protocol state space.

The main structure of the chapter is as follows: in Section 7.2 we define a formal framework for pool mining protocols that encompasses PPLNS and RPPLNS in a language that is amenable to a rigorous mathematical analysis of their properties. Section 7.3 holds the key results of the chapter, where we describe RPPLNS and prove its desirable properties. In Section 7.5 we theoretically justify scenarios when strategic hoarding of blocks occurs in both PPLNS and RPPLNS. Finally, in Section 7.6 we describe possible further areas of research.

7.2 State Machine Mining Pools

In this section we describe a broad class of pool payment mechanisms that encompass many of the most popular pool payment schemes in use today. In general, a pool prepares blocks in advance to send to its pool miners, and pool miners try different nonce values in an attempt to obtain a valid block (with a low enough hash value) that will reward the pool with BTC to be redistributed to pool miners.

7.2.1 Partial Proof-of-Work

As mentioned before, when mining solo (not as a part of a pool), miners obtain an expected reward proportional to their hash power. In order to maintain fairness, a pool needs a proxy to measure the computational resources a miner is contributing to the pool's operations. This is done by allowing pool miners to send the pool operator near-misses to the Bitcoin difficulty threshold.

Definition 7.1 (Valid Shares/Blocks)**.** *For a given value $D \geq 0$ and Bitcoin difficulty threshold $\tau \geq 0$, we say that a block attempt B is a valid share if $H(B) \leq D \cdot \tau$, where $H(B)$ is the hash of the block. If $H(B) \leq \tau$, we say the block is valid (or simply call it a block rather than a share if the context is clear).*

Thus the first choice of a pool is which difficulty parameter, D, it uses for the definition of its valid shares. Note also that valid blocks are themselves considered to be valid shares as well.

7.2.2 Full Generality of Mining Pool Rules

In our full exposition of mining pool rules, we assume the following: that there are n inter-pool strategic miners, $m_1, ..., m_n$, and one honest extra-pool miner, m_0. Furthermore, we define $M_s = \{(i, 0) \mid i \in [n]\}$ and $M_b = \{(i, 1) \mid i \in [n]\}$ such that $M = M_s \cup M_b \cup (0, 1)$ is the set of all valid messages that can be sent to the mining protocol. At any given moment, a message $x \in M$ is sent to the pool. If $x = (i, 0) \in M_s$, then miner i has reported a share to the mining protocol and if $x = (i, 1) \in M_b \cup (0, 1)$, then miner i has reported a block to the mining protocol[1]. In what follows, we also let $\Delta^k = \{\vec{x} \in \mathbb{R}^k \mid x_i \geq 0, \text{ and } \sum x_i = 1\}$ be the k-dimensional simplex.

Definition 7.2 (State Machine Mining Pool)**.** *We say that $\mathcal{M} = (D, T, S, P)$ is a* state machine mining pool protocol *where:*

- *D is the share difficulty.*
- *S is a discrete state space with a distinguished initial state, $s_0 \in S$.*
- *$T : S \times M \to S$ is a state transition function.*
- *$P : S \times M_b \to \Delta^n$ is a payment function.*

The above definition gives rise to two simple payment dynamics which we will shortly define. Interpool miners report shares or blocks to $\mathcal{M}$ which update the internal state $s \in S$ of $\mathcal{M}$ via T. If ever an inter-pool miner reports a valid block via a message $x \in M_b$, then one of two scenarios happens: In a *push-pay* pool, $\mathcal{M}$ transitions to state $s' = T(s, x)$ first and subsequently re-distributes block reward to miners via $P(s', x)$ (block rewards are normalised to 1 unit), and in a *pay-push* pool, $\mathcal{M}$ re-distributes block reward to miners via $P(s, x)$ and then transitions to $s' = T(s, x)$. Finally, if an extra-pool miner finds a block (which is visible to $\mathcal{M}$ by virtue of folllowing the global blockchain), then

[1] m_0 only sends block messages to the pool due to the fact that the concept of a share only makes sense for inter-pool miners. On the other hand, a block message from m_0 corresponds to the pool becoming aware of said block on the global blockchain itself.

$\mathcal{M}$ transitions to state $T(s,(0,1))$. In what follows we cast common mining pool schemes as state machine mining pools. Before doing so, we define what honest miner behaviour is in state machine mining pools.

Definition 7.3 (Honest Mining)**.** *For all pool mining schemes, we say an inter-pool miner is honest if they mine upon the block given by the pool operator and if they report shares/blocks immediately. For solo miners, we say they are honest if they publish blocks they find immediately.*

7.2.3 Examples of State Machine Mining Pools

Proportional-per-share (PPS)

PPS mining pools with share difficulty D can be described succinctly as a state machine mining pool: $\mathcal{M}_p = (D, S_p, T_p, P_p)$ with pay-push dynamics. The state space is $S_p = (\mathbb{Z}_{\geq 0})^n$ with starting state as the zero vector $\vec{0} \in S_p$. To define transition and payment functions in what follows, we let e_i be the i-th unit vector and $\mathbb{I}(\cdot)$ be the indicator function of its argument (i.e $\pi_j(e_i) = \mathbb{I}(i = j)$ where π_j is the j-th projection of a vector).

$$T_p(s,x) = \begin{cases} s + e_i, & \text{if } x = (i,0) \in M_s, \\ \vec{0}, & \text{if } x \in M_b, \\ s, & \text{if } x = (0,1). \end{cases}$$

$$P_p(s,x) = \begin{cases} \frac{(s+e_i)}{\|s+e_i\|_1}, & \text{if } x = (i,1) \in M_b, \\ \vec{0} & \text{otherwise.} \end{cases}$$

Pay-per-last-N-shares (PPLNS)

In PPLNS, pools have an extra parameter, N, designating the length of queue to be used and they use push-pay dynamics. For this reason, we let $\mathcal{M}^N_{PPLNS} = (D, S, T, P)$ be the specific protocol for a given value of N.

- $S = (\{*\} \cup [n])^N$. Here "$*$" represents an empty value while the queue is filled in the first N messages $\mathcal{M}^N_{PPLNS}$ receives. The queue is filled from the left to the right and the starting state is $s_0 = (*)^N_{i=1}$.
- for $x = (i,b) \in M_s \cup M_b$, $T(s,x) = i : s_{<n}$, which is the concatenation of i with the first $n-1$ elements of s. $T(s,(0,1)) = s$.
- $P(s,x)_i = \frac{|\{j \mid T(s,x)_{j=i}\}|}{N}$.

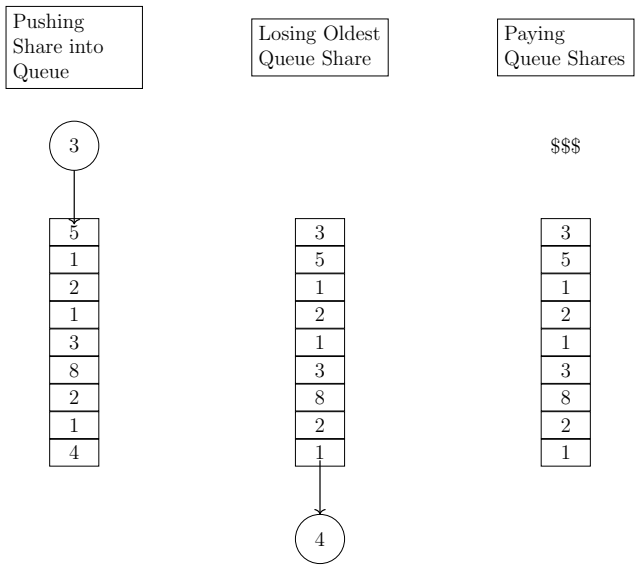

Figure 7.1: A single transition of state for PPLNS. Share 3 is pushed into the queue, causing share 4 at the end of the queue to exit. Upon this transition, all owners of shares in the queue are paid $1/N$ per share in the queue.

PPLNS maintains a sliding window history of shares that have been reported to the pool manager, and whenever a valid block is found, it is pushed into the queue as a share and then the block reward is distributed evenly amongst the N most recent shares sent to the pool manager as shown in Figure 7.1.

$$T(s,x) = \begin{cases} (i : s_{<n}), & \text{if } x = (i,b) \in M_s \cup M_b, \\ s, & \text{if } x = (0,1). \end{cases}$$

$$P(s,x) = \begin{cases} \left(\frac{|\{j \mid T(s,x)_{j=i}\}|}{N} \right)_{j=1}^{n}, & \text{if } x = (i,1) \in M_b, \\ \vec{0} & \text{otherwise.} \end{cases}$$

Score-based Rules.

It is clear from Section 7.2.3, that state machine mining pool protocols are able to maintain a sliding window of information regarding the most recent shares that are sent to the pool operator. With minor tweaks to the structure of PPLNS, one can show that a state machine pool protocol can also maintain information of all shares published between subsequent blocks found by pool miners. Many score-based protocols take this sequence of shares between blocks along with time stamps to decide how much to pay each pool miner, consequently state machine mining pools can also implement general score-based rules.

7.2.4 Probabilistic State Machines

Just as before, we assume that there are n inter-pool strategic miners, $m_1, ..., m_n$, and one honest extra-pool miner, m_0. In an identical fashion, we let $M_s = \{(i, 0) \mid i \in [n]\}$ and $M_b = \{(i, 1) \mid i \in [n]\}$ such that $M = M_s \cup M_b \cup (0, 1)$ is the set of all valid messages that can be sent to the mining protocol. We augment the class of pools we study to include those with probabilistic state transitions and payments:

Definition 7.4 (Probabilistic State Machine Mining Pool)**.** *We say that $\mathcal{M} = (D, T, S, P)$ is a* probabilistic state machine mining pool protocol *if the following hold:*

- *D is the share difficulty.*
- *S is a discrete state space with a distinguished initial state $s_0 \in S$.*
- *$\forall x \in M$ and $\forall s \in S$, $T(s, x)$ is a distribution over S and $P(s, x)$ is a distribution over Δ^m.*

As in the deterministic case, the above definition gives rise to two simple payment dynamics. Interpool miners report shares or blocks to $\mathcal{M}$ which stochastically updates the internal state $s \in S$ of $\mathcal{M}$. If ever an inter-pool miner reports a valid block via a message $x \in M_b$, then one of two scenarios happens: In a *push-pay* pool, $\mathcal{M}$ follows a stochastic transition governed by $T(s, x)$ to s' and subsequently re-distributes block reward to miners via $P(s', x)$, and in a *pay-push* pool, $\mathcal{M}$ re-distributes block reward to miners via $P(s, x)$ and then stochastically transitions to $s' = T(s, x)$. Once more, if an extra-pool miner finds a block (which is visible to $\mathcal{M}$ by virtue of folllowing the global blockchain), then $\mathcal{M}$ stochastically transitions to state $T(s, (0, 1))$. With this definition in hand, we can delve into the definition and study of RPPLNS.

7.3 RPPLNS

Our definition of probabilistic state pool protocols allowed for payments to be stochastic, but RPPLNS will only need for transitions to be stochastic.

7.3.1 Description of Protocol

As in PPLNS, in RPPLNS pools have an extra parameter, N, designating the size of bag to be used and employs push-pay dynamics. For this reason, we let $\mathcal{M}_R^N = (D, S, T, P)$ be the specific protocol for a given value of N.

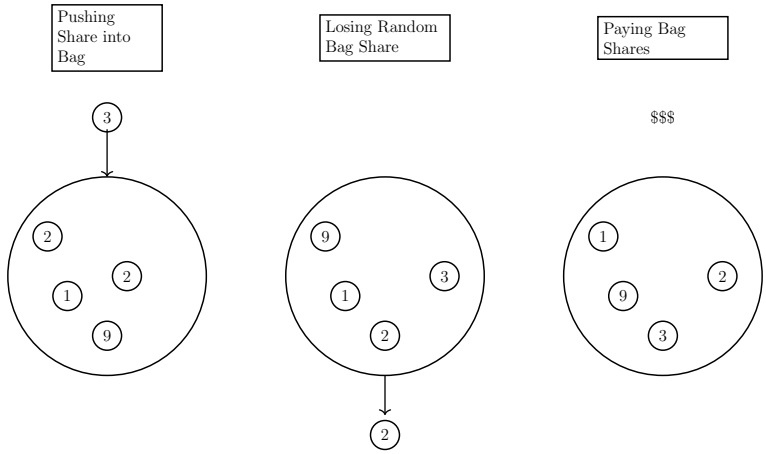

Figure 7.2: A single turn's randomised transition for RPPLNS. The pool receives share 3 and pushes it into the bag by randomly selecting an existing share in the bag to kick out. In this case, share 2 is picked to leave. Upon the randomised push, all owners of shares in the bag are paid $1/N$ per share in the bag.

- $S = \{s \in [N]^m \mid \sum s_i \leq N\}$. s_i represents the number of shares that m_i owns in the bag of $\mathcal{M}_R$. Furthermore, $s_0 = \vec{0} \in [N]^m$. Note $\sum s_i < N$ only occurs for the first N turns of the protocol while the bag is being filled.
- for $s \in S$ and $x = (i, b) \in M_s \cup M_b$, $\mathbb{P}\left(T(s, x) = s - e_j + e_i\right) = \frac{s_j}{N}$. If $\sum s_i < N$, then $T(s, x) = s + e_i$.
- If $x = (i, 1) \in M_b$, then $P(s, x)_j = \frac{s_j + \mathbb{I}(i=j)}{N}$.

Essentially, RPPLNS maintains a bag of N shares that have been reported to the pool manager. Whenever a share is found, a random bag share is chosen to be displaced to make room for the new share, and whenever a valid block is found, this randomised transition is followed by a payment of $1/N$ for each share that a miner may have in the bag. This process is visualised in Figure 7.2.

7.3.2 Notation

In the analysis of RPPLNS, it suffices to consider a single strategic inter-pool miner, m_0 with hash power α, a single honest inter-pool miner m_1 with hash power β, and a single honest extra-pool miner m_2 with hash power γ. Indeed, m_1 and m_2 could be composed of multiple honest miners, but if they are honest, we can model their behaviour as that of a single miner of their aggregate hash power. In addition, to model revenues, we consider a turn-based process. Every turn, either m_0, m_1 or m_2 find a share with probability α, β and γ respectively, and each share has a further $\frac{1}{D}$ probability of being a full block. We wish to point out that we say that m_2 finds shares in the sense that it computes a block

with a hash that is a near-miss to the target hash (by a factor of D). m_2 does not actually report this near miss to the pool since it is not a part of the pool. However, m_2 does publish blocks immediately to all agents of the Bitcoin ecosystem, so we consider this as a message to the pool operator. Finally, since m_1 is an honest inter-honest pool miner, whenever they find a share/block they communicate this immediately to the RPPLNS pool.

7.3.3 Fairness and Variance Reduction

We show that if m_0 is honest, then their expected block reward per turn is precisely α/D. Since each share has a $\frac{1}{D}$ probability of being a block, this coincides with the expected α block reward m_0 would get (per block mined by the system) mining solo. In addition, we demonstrate that RPPLNS enjoys similar variance reduction in block reward to what characterises PPLNS.

Theorem 7.1. *Suppose that m_0 is honest with hash power α, then their expected per-turn block reward is $\frac{\alpha}{D}$ in an RPPLNS mining pool. In addition, the variance of their per-turn block reward is $\frac{1}{D^2}(\alpha - \alpha^2) + \frac{\alpha}{ND}$*

Proof. Suppose that m_0 finds a share and sends it to the pool manager. Let X be the random variable that specifies how much revenue that share makes over its lifetime in the bag, Z. Since RPPLNS is a push-pay protocol, this means that $Z - 1$ is a geometric random variable with mean $N - 1$, since the first turn a share is sent to the protocol it is automatically added to the bag and thus elligible for payment, whereas once the share is kicked out of the bag it is not elligible for the subsequent turn's payment. At each turn over the lifetime of the the share in the bag, let Y_i be the indicator random variable for whether any miner in the pool (i.e. m_0 or m_1) finds a full solution at the i-th turn (including the initial turn when the share is found, as a share can be a full solution as well). This means that the revenue of the share at the i-th turn is Y_i/N. Consequently, we get:

$$X = \sum_{i=1}^{Z} (Y_i/N).$$

Clearly each of the Y_i is iid, hence we can use Wald's equation to get

$$\mathbb{E}(X) = \mathbb{E}(Z)\mathbb{E}(Y_i) = (N)(1/ND) = 1/D.$$

With this in hand, we know that the expected revenue of an honest miner in any given turn is the previous expression multiplied by the probability of getting a share (including a share that is a full solution). Thus the expected revenue of the m_0 is

$$\mathbb{E}(R_0) = \alpha/D.$$

As for the second part of the theorem, we wish to compute the variance of X. To do so, we compute $\mathbb{E}(X^2)$:

$$\mathbb{E}(X^2 \mid Z) = \frac{1}{N^2}\mathbb{E}\left(\sum_{i=1}^{Z} Y_i^2 \mid Z\right).$$

The inner sum can be expressed as

$$\sum_{i<j} \mathbb{E}(Y_i Y_j) + \sum_{i=1}^{Z} \mathbb{E}(Y_i).$$

Since these are iid we get

$$\binom{Z}{2}\frac{1}{D^2} + \frac{Z}{D}.$$

Thus,

$$\mathbb{E}(X^2 \mid Z) = \frac{Z^2 - Z}{2N^2D^2} + \frac{Z}{N^2D}.$$

Now we can use the fact that $\mathbb{E}(Z) = N$ and $\mathbb{E}(Z^2) = 2N^2 - N$

$$\mathbb{E}(X^2) = \frac{(2N^2 - N) + N}{2N^2D^2} + \frac{N}{N^2D} = \frac{1}{D^2} + \frac{1}{ND}.$$

Finally, this means that the second moment of the revenue is

$$\mathbb{E}(R_0^2) = \frac{\alpha}{D^2} + \frac{\alpha}{ND},$$

which in turn tells us the variance is

$$var(R_0) = \mathbb{E}(R_0^2) - \mathbb{E}(R_0)^2 = \frac{1}{D^2}(\alpha - \alpha^2) + \frac{\alpha}{ND}.$$

□

In deterministic PPLNS, block reward variance can be computed in an identical fashion, and it is $\frac{\alpha}{2D^2} + \frac{\alpha}{ND} - \frac{\alpha^2}{D^2} - \frac{\alpha}{2ND^2}$. Typically, pools have $N = 2D$, in which case the PPLNS variance becomes $\frac{1}{D^2}(\alpha - \alpha^2) - \frac{\alpha}{4D^3}$. For this difficulty setting, RPPLNS block reward variance becomes $\frac{1}{D^2}(\alpha - \alpha^2) + \frac{\alpha}{2D^2}$. Though this is more than with standard PPLNS, this still vanishes at the same asymptotic rate of $O(1/N^2)$ when $N = \Theta(D)$.

7.3.4 State Space Reduction

In this section, we prove that RPPLNS generally results in an exponential reduction in the state space $\mathcal{M}$ needs to implement the protocol.

Theorem 7.2. *Suppose that $\mathcal{M}$ is a mining pool with m miners. In order to implement PPLNS with parameter N, $\mathcal{M}$ needs at least m^N states, whereas $\mathcal{M}$ can implement RPPLNS with parameter N using at most $N\frac{(N+m-2)^{m-1}}{(m-1)!}$ states.*

Proof. The first part of the theorem is clear from the definition of PPLNS. We focus on the state that PPLNS needs once the queue is full after the initial N turns. Essentially $\mathcal{M}$ needs to keep track of a sliding window of the ownership of the shares sent to it. Since the messages sent to $\mathcal{M}$ can only come from $[m]$, there are thus m^N many such sequences.

As for the second part of the claim, we begin by focusing on the states used once the bag is full after N turns. We notice that the set of all possible partitions of N bag shares into m owners is in one-to-one correspondence with non-negative integral points of the $m-1$ simplex scaled by a factor of N: $N\Delta^{m-1} = \{x \in \mathbb{R}^{m-1} \mid \forall i\ x_i \geq 0, \text{ and } \|x\|_1 \leq N\}$. In [74], this is shown to be upper bounded by $\frac{(N+m-2)^{m-1}}{(m-1)!}$. For the first N transitions of the protocol, we notice that the set of all possible partitions of $k < N$ bag shares into m owners is in one-to-one correspondence with non-negative integral points of $k\Delta^{m-1} \subsetneq N\Delta^{m-1}$, hence by a union bound we get $|\bigcup_{k=1}^{N} k\Delta^{m-1}| \leq N|N\Delta^{m-1}| \leq N\frac{(N+m-2)^{m-1}}{(m-1)!}$ as desired. □

To put Theorem 7.2 into perspective, it is often the case that $m << N$, so if we let $m = O(1)$ and consider the cardinality of the state space as a function of N, we get that for PPLNS this is exponential in N: $O(m^N)$ and for RPPLNS this is only polynomial in N: $O(N^m)$. Similarly, notice in addition to this, how in PPLNS, if $\mathcal{M}$ wishes to communicate to a specific miner the state of his shares, this requires N bits, as the miner needs to know the position of all his shares in the queue. On the other hand, RPPLNS only needs to communicate $\log N$ bits, due to the fact that it suffices to tell miners how many shares they have in the bag, as there is no sequentiality in the bag.

7.3.5 Robustness to Pool-hopping

In this section we show that if a miner is given the choice between mining with two different RPPLNS pools, then in expectation he will always earn the same block reward, irrespective of the initial state of his shares in each pool and how he may choose to partition his mining between said pools. In order to prove this, suppose that there is a single miner m_0 of hash power α, and two RPPLNS pools, M_i for $i \in \{1, 2\}$. Furthermore, suppose that each M_i has bag size N_i, hash power β_i and difficulty D_i, so that on average, M_i mines one block every D_i shares.

Previously we studied fairness of a single mining pool, and hence we could model share/block mining as a turn-based process, where each turn a share is found, and ownership is dictated by agent hash rates. Having a turn-based approach with multiple pools is more fickle however, since pools may have different share difficulties, and hence the expected duration of such a turn in the continuous-time mining process may be different. For this reason, in this section we instead consider a continuous-time mining process.

As is standard, we assume that share/block mining follows a Poisson process. We further assume that time units are normalised so that the expected time it takes for a block to be mined by the entire mining ecosystem is one time unit. Given this assumption, it follows that if a $\eta \in [0,1]$ proportion of the global mining hash power is dedicated to M_i for T time units, then in expectation $\eta D_i T$ shares are found in M_i for $i \in \{1,2\}$. Given these observations, we are in a position to prove that RPPLNS is pool-hop-proof.

Theorem 7.3. *Suppose that m_0 has A_i shares in M_i for $i \in \{1,2\}$ at time $t = 0$. Furthermore, suppose that $\{I_1, ..., I_k\}$ is an arbitrary finite disjoint collection of closed intervals of $[0,T]$, such that $I_\ell = [x_\ell, y_\ell]$, where T is arbitrary as well. Let $T_2 = \cup_{i=1}^{k} I_i$ and $T_1 = [0,T] \setminus T_2$ and suppose that m_0 mines for M_i for each T_i, $i \in \{1,2\}$, respectively. Then the expected lifetime block reward of m_0 for residual shares A_1, A_2 and shares/blocks mined in $[0,T]$ is $\frac{A_1}{D_1} + \frac{A_2}{D_2} + \alpha T$.*

Proof. We first derive the expected payoff m_0 obtains from A_i residual shares in M_i. Since the pool manager will remove one share out of queue randomly whenever a new share arrives, the expected number of these A_i residual shares will shrink exponentially as new shares come in. After n new shares, the expected number of residual shares will be $A_i \left(\frac{N_i - 1}{N_i}\right)^{n-1}$. Furthermore, every time a share is found, it has a $\frac{1}{D_i}$ chance of being a full solution, in which case every share m_0 has in the bag pays $\frac{1}{N_i}$ block reward. This means that the total expected lifetime payoff from these A_i shares can be written as,

$$\begin{aligned}
&\frac{1}{D_i}\frac{A_i}{N_i}\left(1 + \frac{N_i - 1}{N_i} + \left(\frac{N_i - 1}{N_i}\right)^2 + \dots\right), \\
&= \frac{1}{D_i}\frac{A_i}{N_i}\frac{1}{1 - \frac{N_i - 1}{N_i}} = \frac{A_i}{D_i}.
\end{aligned}$$

It only remains to compute the expected lifetime block reward m_0 obtains from shares/blocks mined during $[0,T]$. In what follows we let $|T_2| = \sum_{i=1}^{k} |y_i - x_i|$ and $|T_1| = T - |T_2|$ denote the amount of time m_0 mines in M_i during $[0,T]$. We note that during T_1, a total of $\alpha + \beta_1$ hash power is being contributed to M_1. This means that M_1 mines $D_1|T_1|(\alpha + \beta_1)$ shares in expectation during during this time. Furthermore, every share has a $\frac{\alpha}{\alpha+\beta}$ probability of belonging to m_0, therefore m_0 finds $D_1\alpha|T_1|$ shares for M_1 in T_1. Furthermore, as we saw in Theorem 7.1, each of these shares earns $\frac{1}{D_1}$ block reward over its lifetime in expectation, therefore the total lifetime block rewards from shares found for M_1 in $[0,T]$ is $\alpha|T_1|$ in expectation. In an identical fashion we can show that the lifetime rewards for shares found for M_2 in $[0,T]$ is $\alpha|T_2|$. Putting everything together, this means that the total expected lifetime reward for the residual shares A_1, A_2 and newly found shares in $[0,T]$ is $\frac{A_1}{D} + \frac{A_2}{D} + \alpha T$ as desired.

□

The fact that the expected lifetime reward is an expression that is independent of the partition, T_1, precisely implies that RPPLNS is robust to pool-hopping. Furthermore, it is straightforward to generalise Theorem 7.3 to encompass a choice of hopping between an arbitrary number of pools, yielding the same robustness to pool-hopping in this setting.

7.3.6 Steady State of Honest Pool for RPPLNS

In this section we show that the number of shares m_0 has in the bag of size N can be modelled as an ergodic Markov chain. We explicitly derive the steady state distribution and use it to compute the expected number of shares the miner has in the bag in the honest steady state.

The Markov Chain

Let us suppose that miner m_0 currently has $i \in \{0, ..., N\}$ shares in the bag and is honest. With probability α, m_0 finds a block/share and publishes it to the pool operator. Conditional to this, with probability $\frac{i}{N}$, the number of shares stays the same, $i \to i$, and with probability $\frac{N-i}{N}$, the number of shares increases by one, $i \to i+1$. On the other hand, m_1 finds a block/share and publishes it with probability β. Conditional to this, with probability $\frac{i}{N}$, the number of shares goes down by one, $i \to i-1$, and with probability $\frac{N-i}{N}$, the number of shares stays the same. Clearly, this induces an ergodic Markov chain on the state space $S = \{0, ..., N\}$. Suppose that $\pi = (\pi_i)_{i=1}^N$, is the unique honest steady state distribution over S.

Theorem 7.4. *For the steady state distribution π of miner shares in the RPPLNS bag:*

$$\pi_k = \frac{\binom{N}{k}\left(\frac{\alpha}{\beta}\right)^k}{\left(1+\frac{\alpha}{\beta}\right)^N}, \text{ for } k = 0, ..., N.$$

Furthermore, the expected number of shares in the bag under π is

$$N\left(\frac{\alpha}{\alpha+\beta}\right).$$

Proof. From the topology of the chain, we can make a cut-set between $S_i = \{0, ..., i\}$ and $S_i^c = \{i+1, ..., N\}$ for $i = 0, ..., N-1$. From steady state conditions it follows that π must satisfy $\pi_i P_{i\to i+1} = \pi_{i+1} P_{i+1\to i}$ for all such cut-sets. From before, we know that $P_{i\to i+1} = \alpha\frac{N-i}{N}$, and $P_{i+1\to i} = \beta\frac{i+1}{N}$. As a consequence, we obtain $\frac{\pi_{i+1}}{\pi_i} = \frac{\alpha}{\beta}\left(\frac{N-i}{i+1}\right) = f(i)$. Using this expression we know that $\pi_{i+1} = f(i)\pi_i$ for all $i = 0, ..., N-1$, consequently $\pi_k = \left(\prod_{j=0}^{k-1} f(j)\right)\pi_0 = \pi_0\binom{N}{k}(\frac{\alpha}{\beta})^k$ for all k. The final step in obtaining the steady state is normalising the sum of all terms, which corresponds to $\sum_{k=0}^N \pi_k = \pi_0 \sum_{k=0}^N \binom{N}{k}(\frac{\alpha}{\beta})^k$.

Using the well-known expansion for $(1+x)^n$, we get that this is $\pi_0 \left(1+\frac{\alpha}{\beta}\right)^N$. As a consequence, it follows that for π to be a steady state, we need $\pi_0 = \left(1+\frac{\alpha}{\beta}\right)^{-N}$ and consequently:

$$\pi_k = \frac{\binom{N}{k}\left(\frac{\alpha}{\beta}\right)^k}{\left(1+\frac{\alpha}{\beta}\right)^N},\ k = 0, ..., N.$$

Now we wish to compute the expected number of shares in the bag in the steady state by using our expression π. To do so, let us first define $f(x) = (1+x)^n$ for $n \in \mathbb{N}$ and $x \in \mathbb{R}$. We know that $f(x) = \sum_{k=0}^{n} \binom{n}{k} x^k$. Therefore, $nx(1+x)^{n-1} = x\frac{d}{dx}f(x) = \sum_{k=0}^{n} \binom{n}{k} k x^k$. We use this formula to compute the expectation over π, $\mathbb{E}_\pi = \sum_{k=0}^{N} k\pi_k$.

$$\begin{aligned}
\mathbb{E}_\pi = \sum_{k=0}^{N} k\pi_k &= \left(1+\frac{\alpha}{\beta}\right)^{-N} \sum_{k=0}^{N} \binom{N}{k} k \left(\frac{\alpha}{\beta}\right)^k, \\
&= \left(1+\frac{\alpha}{\beta}\right)^{-N} N\left(\frac{\alpha}{\beta}\right)\left(1+\frac{\alpha}{\beta}\right)^{N-1}, \\
&= N\left(\frac{\alpha}{\alpha+\beta}\right).
\end{aligned} \tag{7.1}$$

□

As a first consequence, note that this also constitutes a proof that RPPLNS is fair if everyone is honest. The reason for this is that in each turn, we suppose that one of either m_0, m_1 or m_2 find a share with probability α, β, γ respectively, and each share has a further probability of $\frac{1}{D}$ of being a full solution. Thus the probability that a single turn ends up paying agents from the pool is precisely $\frac{\alpha+\beta}{D}$. As a consequence, the expected payment to m_0 in the steady state is $(\mathbb{E}_\pi)\left(\frac{\alpha+\beta}{ND}\right) = \frac{\alpha}{D}$, which is precisely the expected per-turn payment of m_0 mining solo.

7.4 When is Honest Mining a Dominant Strategy

We recall that we are in the setting of a single pool miner being strategic. In other words we have m_0, m_1 and m_2 of hash powers α, β and γ respectively. We wish to find conditions such that m_0 is honest (i.e. publishes shares and blocks to $\mathcal{M}_R$ immediately). Given this setting, at any given moment, we describe the state of m_0 within the system with a tuple $(\ell, s, b) \in [N]^2 \times \{0, 1\}$. ℓ represents the shares m_0 has in the bag of the pool, s represents the private shares of m_0, and b represents whether m_0 has a private block or not. Note that $b \leq 1$ is without loss of generality, as the extra blocks mined are invalidated once one of them is released. Requiring $s \leq N$ is not without loss of generality, since technically

a selfish miner could hoard an infinite amount of shares. However, it is reasonable to assume that $\alpha < 0.5$ (to prevent scenarios where double spend attacks are possible in the first place). Hence it is exceedingly unlikely that s will reach values higher than N, given a typical difficulty such as $D = N/2$. Furthermore, as we will soon see, any time an agent other then m_0 finds a block, private shares are invalidated, hence further making it difficult to hoard a large amount of private shares. The benefit of upper bounding s and b is significant, as the state space becomes finite.

Recurrence Relation

Let $g_k(\ell, s, b)$ be the maximum revenue a strategic miner can obtain when acting optimally starting at $g(\ell, s, b)$, after exactly k shares in total have been mined by his pool. The parameter k is necessary for the recursion to prevent it from being open ended.

$$g_k(\ell, s, b) = max \begin{cases} \text{A}, & (m_0 \text{ waits and } s < N), \\ \text{B}, & (m_0 \text{ waits and } s = N), \\ \text{C}, & (m_0 \text{ publishes a share}), \\ \text{D}, & (m_0 \text{ publishes a block}), \end{cases} \tag{7.2}$$

We present the expressions for A, B, C and D in the following sections.

m_0 Waits and $s < N$

When m_0 waits and $s < N$, there are 6 immediate outcomes:

- m_0 gets a block. Call this event "a" which happens with probability $\alpha\left(\frac{1}{D}\right)$. The resulting state is $(\ell, s, 1)$.
- m_0 gets a share. Call this event "b" which happens with probability $\alpha\left(\frac{D-1}{D}\right)$. The resulting state is $(\ell, s+1, b)$.
- m_1 gets a block. Call this event "c" which happens with probability $\beta\left(\frac{1}{D}\right)$. Since m_1 is honest, he publishes this block and the private shares and blocks of m_0 are rendered obsolete. This means that with probability $\frac{\ell}{N}$ a share of m_0 is kicked out of the bag resulting in state $(\ell - 1, 0, 0)$ and m_0 getting paid $\frac{\ell-1}{N}$. With probability $\frac{N-\ell}{N}$ a share of m_1 is kicked out of the bag resulting in state $(\ell, 0, 0)$ and m_0 getting paid $\frac{\ell}{N}$.
- m_1 gets a share. Call this event "d" which happens with probability $\beta\left(\frac{D-1}{D}\right)$. Since m_1 is honest, he publishes this share. This means that with probability $\frac{\ell}{N}$ a share of m_0 is kicked out of the bag and the resulting state is $(\ell - 1, s, b)$, and with

probability $\frac{N-\ell}{N}$ a share of m_1 is kicked out of the bag and the resulting state is (ℓ, s, b).

- m_2 gets a block. Call this event "e" which happens with probability $\gamma\left(\frac{1}{D}\right)$. Since m_2 is honest, he publishes this block and the private shares and blocks of m_0 are rendered obsolete. This means that the resulting state is $(\ell, 0, 0)$.
- m_2 gets a share. Call this event "f" which happens with probability $\gamma\left(\frac{D-1}{D}\right)$. In this case the state remains at (ℓ, s, b)

We can now get an expression for A in the recursion for g_k:

$$\begin{aligned} A = \quad & \alpha\left(\frac{1}{D}\right) g_{k-1}(\ell, s, 1) + \alpha\left(\frac{D-1}{D}\right) g_{k-1}(\ell, s+1, b) \\ & + \beta\left(\frac{1}{D}\right)\left(\frac{\ell}{N} g_{k-1}(\ell-1, 0, 0) + \frac{N-\ell}{N} g_{k-1}(\ell, 0, 0) + \frac{\ell}{N}\right) \\ & + \beta\left(\frac{D-1}{D}\right)\left(\frac{\ell}{N} g_{k-1}(\ell-1, s, b) + \frac{N-\ell}{N} g_{k-1}(\ell, s, b)\right) \\ & + \gamma\left(\frac{1}{D}\right) g_{k-1}(\ell, 0, 0) + \gamma\left(\frac{D-1}{D}\right) g_{k-1}(\ell, s, b). \end{aligned}$$

m_0 Waits and $s = N$

This case is almost identical to the above. The only difference is in event b: when m_0 gets a share. Since we don't want to consider an infinite state space, we will assume that upon obtaining N private shares, m_0 will always publish subsequent mined shares. Since N is usually quite large, the probability of even obtaining this state is small.

- m_0 gets a share. Call this event "b" which happens with probability $\alpha\left(\frac{D-1}{D}\right)$. m_0 will publish this share. Hence the resulting state is $(\ell, s, b) = (\ell, N, b)$.

With identical reasoning to the above, we get the following expression for B:

$$\begin{aligned} A = \quad & \alpha\left(\frac{1}{D}\right) g_{k-1}(\ell, N, 1) + \alpha\left(\frac{D-1}{D}\right) g_{k-1}(\ell, N, b) \\ & + \beta\left(\frac{1}{D}\right)\left(\frac{\ell}{N} g_{k-1}(\ell-1, 0, 0) + \frac{N-\ell}{N} g_{k-1}(\ell, 0, 0) + \frac{\ell}{N}\right) \\ & + \beta\left(\frac{D-1}{D}\right)\left(\frac{\ell}{N} g_{k-1}(\ell-1, N, b) + \frac{N-\ell}{N} g_{k-1}(\ell, N, b)\right) \\ & + \gamma\left(\frac{1}{D}\right) g_{k-1}(\ell, 0, 0) + \gamma\left(\frac{D-1}{D}\right) g_{k-1}(\ell, N, b). \end{aligned}$$

m_0 Publishes a Share

If the current state of the system is (ℓ, s, b), then by publishing a share, there is a $\frac{\ell}{N}$ probability that a share belonging to m_0 is kicked out of the bag, resulting in the state $(\ell, s-1, b)$. There is a $\frac{N-\ell}{N}$ probability that a share belonging to m_1 is kicked out of the bag, resulting in the state $(\ell+1, s-1, b)$. Note that a share is not actually mined, therefore the recurrence does not involve g_{k-1}. Putting this together, we get the following value of C:

$$C = \frac{\ell}{N} g_k(\ell, s-1, b) + \frac{N-\ell}{N} g_k(\ell+1, s, b).$$

m_0 Publishes a Block

If the current state of the system is (ℓ, s, b), then by publishing a block, there is a $\frac{\ell}{N}$ probability that a share belonging to m_0 is kicked out of the bag, resulting in the state $(\ell, s-1, b)$ and m_0 earning $\frac{\ell}{N}$. There is a $\frac{N-\ell}{N}$ probability that a share belonging to m_1 is kicked out of the bag, resulting in the state $(\ell+1, s-1, b)$ and m_0 earning $\frac{\min(N, \ell+1)}{N}$. Putting this together, we get the following value of D:

$$D = \frac{\ell}{N}\left(\frac{\ell}{N} + g_k(\ell, 0, 0)\right) + \frac{N-\ell}{N}\left(\frac{\min(N, \ell+1)}{N} + g_k(\ell+1, 0, 0)\right).$$

Checking for Equilibrium Conditions

Observe that since the state space is finite the value obtained per 'step' k is at most 1, the following limit is well defined for all (l, s, b):

$$\phi(l, s, b) = \lim_{k \to \infty} \frac{g_k(l, s, b)}{k}.$$

This $\phi(l, s, b)$ can be thought of as the potential of state (l, s, b). Comparing ϕ values of different states, we can deduce where the miner obtains higher payoffs.To tell for which α, β, N, and D honest mining is a dominant strategy, we need to check the following conditions:

$$\phi(\ell+1, 0, 0) \geq \phi(\ell, 1, 0) \text{ when } (\ell) \in [N]. \tag{7.3}$$

$$\frac{N-\ell}{N} \cdot \phi(\ell+1, 0, 0) + \frac{\ell}{N} \cdot \phi(\ell, 0, 0) \geq \phi(\ell, 0, 1) \text{ when } (\ell) \in [N], \tag{7.4}$$

where eq. (7.3) and eq. (7.4) show that releasing shares and blocks respectively is at least as good as hoarding them.

Computational Setup

Unfortunately we cannot solve exactly for ϕ, but we can approximate it well enough given high enough k to observe if strategic mining is an option. We will use g_k as defined in eq. (7.2) with one difference: if state (l, N, b) is reached the miner is rewarded 1 and jumps to state $(N, 0, 0)$. This incentivizes hoarding blocks, since hoarding enough of them results in an instant maximum payoff. The reason for this is to strengthen the experimental results: if honest mining is viable in this setting, it is more likely that it is actually the best option.

Having computed g_k for large enough k, we can then check Equation (7.3) as it is. For Equation (7.4) we need the following small adjustment:

$$\frac{N-\ell}{N}\left(\frac{\ell+1}{N}+g_k(\ell+1,0,0)\right)+\frac{\ell}{N}\left(\frac{\ell}{N}+g_k(\ell,0,0)\right) > g_k(\ell,0,1) \text{ when } (\ell)\in[N].$$

This is because as k goes to infinity, the immediate rewards $(\ell+1)/N$ and ℓ/N vanish compared to the potential of the three states involved (as ϕ is defined by diving with k). Since we are working with finite k, we need to plug them back in.

Results

To understand the behaviour of the miners, we explicitly compute the truncated recursion in Section 7.4 for $N = 1000, D = 500$ and $k = 150$. Each of the following graphs shows the best action for m_0 given an *initial* fraction of shares in the bag, which we call F. For $F \leq 0.35$, we have:

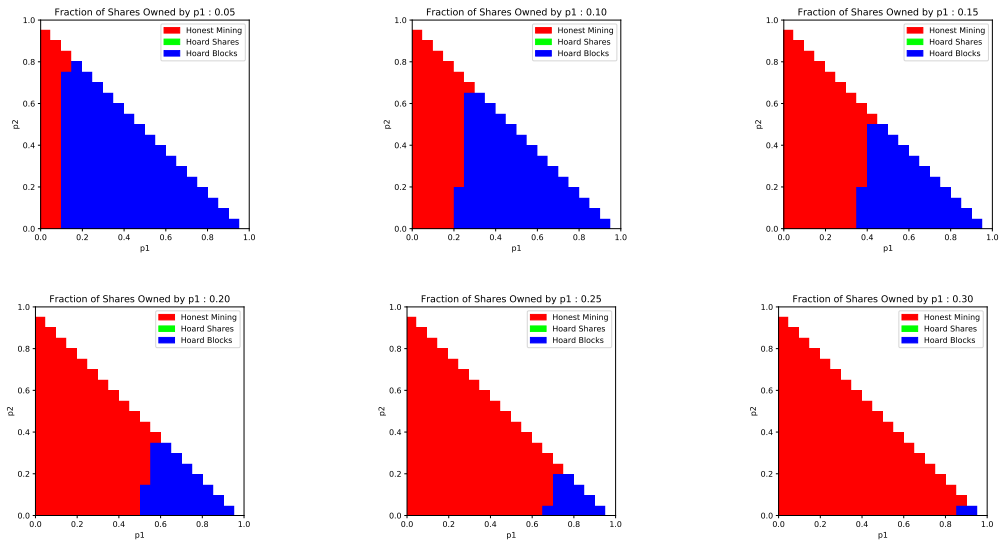

Figure 7.3: Best action for m_0 for $F \leq 0.35$.

In this regime of F values, the strategic miner begins by controlling but a few shares of the bag, and his best option is often to *hoard* a block. This is because they expect to be able to add a couple more shares to the bag before someone else manages to mine another block and invalidate their private block. In Section 7.5 we formally demonstrate that for common difficulty settings of $D = \Theta(N)$ (typically $D = N/2$), if $\alpha = \Omega(1/N)$, and a pool miner has no shares in the bag, then hoarding a block for a single turn dominates behaving honestly. This behaviour is not unique to RPPLNS though, as we also demonstrate in the same section that for similar hash rates, if a pool miner is facing an empty queue in PPLNS, they will hoard a private block with hopes of finding a share in subsequent turns.

What is most interesting though, is that if we recall our derivations of the steady state of bag shares from Section 7.3.6, strategic block hoarding only occurs at hash rates and F values such that F is in fact much less than the expected number of bag shares in the steady state. This suggests that if all miners behave honestly, a single pool miner is more likely to find himself at a state where mining honestly is a dominant strategy.

On the other hand, for initial share distributions, $0.35 \leq F \leq 0.70$, our recursions suggest that honest mining is the best option throughout. We have omitted graphs of these cases since they simply paint the simplex red entirely. Finally, when $F \geq 0.70$, as in the low initial share regime, we see that strategic behaviour arises once more, though this time in the form of hoarding shares rather than blocks. For example, it is not difficult to see that if $F = 1$, then in both PPLNS and RPPLNS hoarding a share dominates publishing it immediately, as doing so has no effect on the state of the pool.

Once again, share hoarding becomes a better strategy at initial share distributions that are far from the expected share distribution of the steady state of bag shares from Section 7.3.6. This once again suggests that states where miners act strategically are more rare than those where a miner is honest.

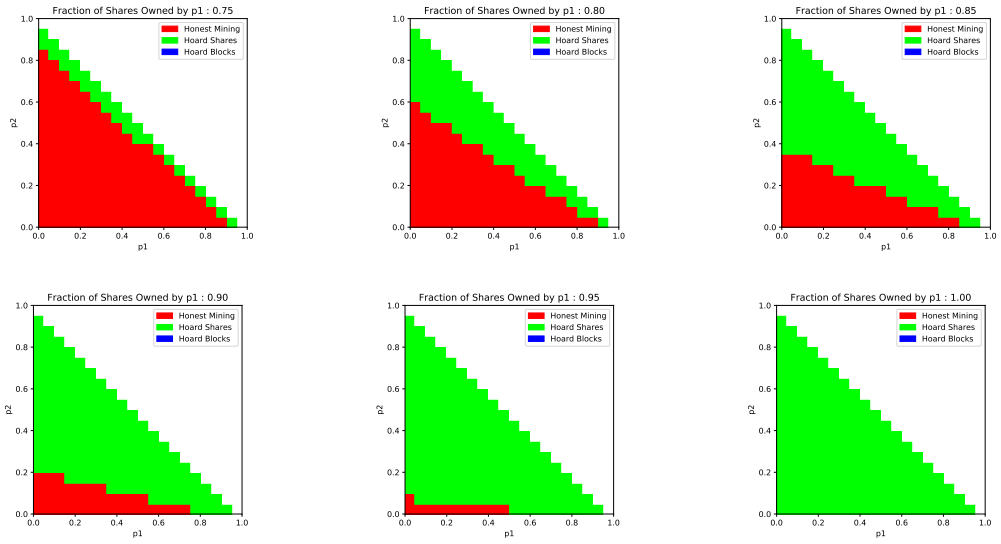

Figure 7.4: Best action for m_0 for $F \geq 0.75$.

7.5 Strategic Hoarding

As seen in our empirical results, there are parameter settings in which we see strategic pool mining in RPPLNS. In this section we theoretically justify why this is the case for block hoarding in particular.

7.5.1 PPLNS

We assume that there is a single strategic pool miner, m_0 with hash power α. This miner operates within a PPLNS mining pool with hash power $\alpha+\beta$, where the honest miners in the pool can be considered as a single honest pool miner m_1. Finally, we let m_2 represent all other honest miners in a system with collective hash power $\gamma = 1-\alpha-\beta$.

Let us suppose that m_0 has no shares in the pool queue and that he currently holds a valid block he could send to the pool operator. We compare two mining strategies over a two-element decision window. If m_0 is honest, he will immediately publish this block, wait one more turn to see who receives a share/block, and subsequently act honestly (publish a new share/block immediately if found). On the other hand, we consider a one-time strategic deviation by m_0 as follows: m_0 hoards the current block he has, and waits one more turn to see what occurs. If he receives a share, he publishes the share before his held block. In all other scenarios he acts honestly. We call the honest strategy H and the strategic deviation S.

Theorem 7.5. *If $\alpha > \frac{N+D-1}{(D-1)^2}$, then S is a strictly dominates H.*

Proof. How H and S perform depends on precisely one of 6 cases:

- m_0 finds a block in the next turn. The probability of this is $\alpha\frac{1}{D}$.
- m_0 finds a share in the next turn. The probability of this is $\alpha\frac{D-1}{D}$.
- m_1 finds a block in the next turn. The probability of this is $\beta\frac{1}{D}$.
- m_1 finds a share in the next turn. The probability of this is $\beta\frac{D-1}{D}$.
- m_2 finds a block in the next turn. The probability of this is $\gamma\frac{1}{D}$.
- m_2 finds a share in the next turn. The probability of this is $\gamma\frac{D-1}{D}$.

First we focus on the expected payoffs H receives in each of these cases:

- Here m_0 publishes two blocks back to back. This results in an immediate reward of $\frac{3}{N}$. The older block is still eligible for payment for $N-2$ turns and the latter for $N-1$ turns, hence their expected future payment is $\frac{N-2}{ND}+\frac{N-1}{ND}$. Overall we denote this expected revenue by $H_1=\frac{3}{N}+\frac{N-2}{ND}+\frac{N-1}{ND}$.
- Here m_0 publishes a block and then a share. This results in an immediate reward of $\frac{1}{N}$. As before, the older share is alive for $N-2$ more turns and the latter for $N-1$, hence their expected future payment is $\frac{N-2}{ND}+\frac{N-1}{ND}$. Overall we denote this expected revenue by $H_2=\frac{1}{N}+\frac{N-2}{ND}+\frac{N-1}{ND}$.
- Here m_0 publishes a block and m_1 subsequently publishes a block. This results in an immediate reward of $\frac{2}{N}$. The first block is alive for another $N-2$ turns, hence its expected future payment is $\frac{N-2}{ND}$. Overall we denote this expected revenue by $H_3=\frac{2}{N}+\frac{N-2}{ND}$.
- Here m_0 publishes a block and m_1 subsequently publishes a share. This results in an immediate reward of $\frac{1}{N}$. The first block is alive for another $N-2$ turns, hence its expected future payment is $\frac{N-2}{ND}$. Overall we denote this expected revenue by $H_4=\frac{1}{N}+\frac{N-2}{ND}$.
- Here m_0 publishes a block and m_2 subsequently finds a block. The latter event has no bearings on the pool, hence this is equivalent to m_0 publishing a single block, which gives an immediate reward of $\frac{1}{N}$ and since this block is alive for another $N-1$ turns, an expected future reward of $\frac{N-1}{ND}$. Overall we denote this expected revenue by $H_5=\frac{1}{N}+\frac{N-1}{ND}$.

- Here m_0 publishes a block and m_2 subsequently finds a share. The latter event has no bearings on the pool, hence this is equivalent to m_0 publishing a single block, which gives an immediate reward of $\frac{1}{N}$ and since this block is alive for another $N-1$ turns, an expected future reward of $\frac{N-1}{ND}$. Overall we denote this expected revenue by $H_6 = \frac{1}{N} + \frac{N-1}{ND}$.

Given our previous expressions, we can write the expected revenue m_0 obtains from using H by

$$R_H = H_1\left(\frac{\alpha}{D}\right)+H_2\left(\frac{\alpha(D-1)}{D}\right)+H_3\left(\frac{\beta}{D}\right)+H_4\left(\frac{\beta(D-1)}{D}\right)+H_5\left(\frac{\gamma}{D}\right)+H_6\left(\frac{\gamma(D-1)}{D}\right).$$

We proceed to compute revenues for the strategic deviation S:

- Here m_0 hoards a block and then receives a new block. This forcibly invalidates a single block hence, hence m_0 only gets an immediate reward of $\frac{1}{N}$ and subsequently this block lives another $N-1$ turns to obtain an expected lifetime revenue of $\frac{N-1}{ND}$. Overall we denote this expected revenue by $S_1 = \frac{1}{N} + \frac{N-1}{ND}$.

- Here m_0 hoards a block and then recieves a share, publishing the share first and then the block. This results in an immediate reward of $\frac{2}{N}$. Subsequently the latter share is alive for $N-2$ turns and the block for $N-1$, hence the expected future payoff of these is $\frac{N-2}{ND}+\frac{N-1}{ND}$. Overall we denote this expected revenue by $S_2 = \frac{2}{N}+\frac{N-2}{ND}+\frac{N-1}{ND}$.

- Here m_0 hoards a block and m_1 publishes a block. This invalidates the secret block and m_0 gets no revenue. We denote this by $S_3 = 0$.

- Here m_0 hoards a block and m_1 publishes a share. m_0 only gets an immediate reward of $\frac{1}{N}$ and subsequently this block lives another $N-1$ turns to obtain an expected lifetime revenue of $\frac{N-1}{ND}$. Overall we denote this expected revenue by $S_4 = \frac{1}{N} + \frac{N-1}{ND}$.

- Here m_0 hoards a block and m_2 publishes a block. This invalidates the secret block and m_0 gets no revenue. We denote this by $S_5 = 0$.

- Here m_0 hoards a block and m_1 publishes a share. m_0 only gets an immediate reward of $\frac{1}{N}$ and subsequently this block lives another $N-1$ turns to obtain an expected lifetime revenue of $\frac{N-1}{ND}$. Overall we denote this expected revenue by $S_6 = \frac{1}{N} + \frac{N-1}{ND}$.

Given our previous expressions, we can write the expected revenue m_0 obtains from using S by

$$\begin{aligned} R_S =& S_1\left(\frac{\alpha}{D}\right) + S_2\left(\frac{\alpha(D-1)}{D}\right) + S_3\left(\frac{\beta}{D}\right) \\ &+ S_4\left(\frac{\beta(D-1)}{D}\right) + S_5\left(\frac{\gamma}{D}\right) + S_6\left(\frac{\gamma(D-1)}{D}\right). \end{aligned}$$

With these expressions in hand, we are interested in parameter values such that $R_S > R_H$.

$$\begin{aligned} R_S - R_H = &- \frac{\alpha}{D}\left(\frac{2}{N} + \frac{N-2}{ND}\right) + \frac{\alpha(D-1)}{D}\left(\frac{1}{N}\right) - \frac{\beta}{D}\left(\frac{2}{N} + \frac{N-2}{ND}\right) \\ &+ \frac{\beta(D-1)}{D}\left(\frac{1}{ND}\right) - \frac{\gamma}{D}\left(\frac{1}{N} + \frac{N-1}{ND}\right). \end{aligned}$$

We wish to show when this latter expression is strictly greater than 0. First we notice that the final term $-\frac{\gamma}{D}\left(\frac{1}{N} + \frac{N-1}{ND}\right) = -\frac{\gamma}{D}\left(\frac{2}{N} + \frac{N-2}{ND}\right) + \frac{\gamma}{D}\left(\frac{1}{N} - \frac{1}{ND}\right)$. This means that we can join all terms that have $\frac{2}{N} + \frac{N-2}{ND}$ and take advantage of the fact that $\alpha + \beta + \gamma = 1$. This gives us the following expression:

$$R_S - R_H = \frac{\alpha(D-1)}{ND} + \frac{\beta(D-1)}{ND^2} + \frac{\gamma}{D}\left(\frac{1}{N} - \frac{1}{ND}\right) - \frac{1}{D}\left(\frac{2}{N} + \frac{N-2}{ND}\right) > 0.$$

We can now multiply both sides of the inequality by $\frac{ND}{D-1}$ to isolate α and get

$$\alpha + \frac{\beta}{D} + \frac{\gamma}{D-1} - \frac{\gamma}{D(D-1)} - \frac{2}{D-1} - \frac{N-2}{D(D-1)} > 0.$$

Subsequent simplification gives

$$\begin{aligned} \alpha &> \frac{2D + N - 2 + \gamma - \beta(D-1) - D\gamma}{D(D-1)}, \\ &= \frac{2D + N - 2 - (D-1)(\beta+\gamma)}{D(D-1)}, \\ &= \frac{2D + N - 2 - (D-1)(1-\alpha)}{D(D-1)}, \\ &= \frac{N + D - 1}{D(D-1)} + \frac{\alpha}{D}, \\ \alpha\left(\frac{D-1}{D}\right) &> \frac{N + D - 1}{D(D-1)}, \\ \alpha &> \frac{N + D - 1}{(D-1)^2}. \end{aligned} \tag{7.5}$$

□

In most cases $N = 2D$, which gives a lower bound of $\alpha > \frac{3D-1}{(D-1)^2}$. D is also usually quite large, hence this shows that in almost all cases miners will hoard blocks in the case they get lucky and find a block with an empty queue!

7.5.2 RPPLNS

Since RPPLNS is memoryless, we can extend the above analysis to the case where m_0 miner has k shares in the bag and holds a private block. We want to study the expected revenue from being honest for a single turn vs. waiting for the next turn in hopes of finding another share to publish before the withheld block.

Important Terms

In RPPLNS, if a specific share surives being pushed, it will be eligible for payment another $N-1$ pushes in expectation due to the memory-less property of their survival (it is a geometric random variable). If everyone is honest, each of these payment opportunities has a $\frac{1}{D}$ probability of actually paying a $\frac{1}{N}$ amount, hence shares that survive being pushed give m_0 an expected revenue of $\frac{N-1}{ND}$.

To simplify the expressions, we let $f_i^B(k,N)$ and $f_i^S(k,N)$ represent the expected utility m_0 makes when miner $i \in \{0,1\}$ pushes a block or a share into the bag respectively:

$$\begin{aligned}
f_0^B(k,N) &= \frac{k}{N}\left(\frac{k}{N} + k\left(\frac{N-1}{ND}\right)\right) + \frac{N-k}{N}\left(\frac{k+1}{N} + (k+1)\left(\frac{N-1}{ND}\right)\right), \\
f_0^S(k,N) &= \frac{k}{N}\left(k\left(\frac{N-1}{ND}\right)\right) + \frac{N-k}{N}\left((k+1)\left(\frac{N-1}{ND}\right)\right), \\
f_1^B(k,N) &= \frac{k}{N}\left(\frac{k-1}{N} + (k-1)\left(\frac{N-1}{ND}\right)\right) + \frac{N-k}{N}\left(\frac{k}{N} + k\left(\frac{N-1}{ND}\right)\right), \\
f_1^S(k,N) &= \frac{k}{N}\left((k-1)\left(\frac{N-1}{ND}\right)\right) + \frac{N-k}{N}\left(k\left(\frac{N-1}{ND}\right)\right).
\end{aligned} \tag{7.6}$$

With this in hand, we consider a two-turn scenario as with PPLNS. In one m_0 has a block in hand, and is honest with this current block and with what happens the following term. This strategy is H for honest. In the second case, m_0 hoards this block in hand with hopes of mining a share the following turn and publishing the share before the block. After this minor strategic deviation for a turn though, m_0 returns to being honest. This strategy is referred to by S. First we write the expected utility for each action for H

$$\begin{aligned}
H_1 &= \frac{k}{N}\left(\frac{k}{N} + f_0^B(k,N)\right) + \frac{N-k}{N}\left(\frac{k+1}{N} + f_0^B(k+1,N)\right), \\
H_2 &= \frac{k}{N}\left(\frac{k}{N} + f_0^S(k,N)\right) + \frac{N-k}{N}\left(\frac{k+1}{N} + f_0^S(k+1,N)\right), \\
H_3 &= \frac{k}{N}\left(\frac{k}{N} + f_1^B(k,N)\right) + \frac{N-k}{N}\left(\frac{k+1}{N} + f_1^B(k+1,N)\right), \\
H_4 &= \frac{k}{N}\left(\frac{k}{N} + f_1^S(k,N)\right) + \frac{N-k}{N}\left(\frac{k+1}{N} + f_1^S(k+1,N)\right), \\
H_5 &= f_0^B(k,N), \\
H_6 &= f_0^B(k,N).
\end{aligned} \tag{7.7}$$

Next we write the expected utility for each action for S

$$\begin{aligned}
S_1 &= f_0^B(k, N), \\
S_2 &= \frac{k}{N}\left(f_0^B(k, N)\right) + \frac{N-k}{N}\left(f_0^B(k+1, N)\right), \\
S_3 &= 0, \\
S_4 &= \frac{k}{N}\left(f_0^B(k-1, N)\right) + \frac{N-k}{N}\left(f_0^B(k, N)\right), \\
S_5 &= 0, \\
S_6 &= f_0^B(k, N).
\end{aligned} \tag{7.8}$$

We remember the probability of each state

$$\begin{aligned}
p_1 &= \frac{\alpha}{D}, \\
p_2 &= \frac{\alpha(D-1)}{D}, \\
p_3 &= \frac{\beta}{D}, \\
p_4 &= \frac{\beta(D-1)}{D}, \\
p_5 &= \frac{\gamma}{D}, \\
p_6 &= \frac{\gamma(D-1)}{D}.
\end{aligned} \tag{7.9}$$

With this in hand, we know that the following condition implies that S dominates H when m_0 has a block in hand:

$$R_S = \sum_{i=1}^{6}(S_i)p_i > \sum_{i=1}^{6}(H_i)p_i = R_H.$$

The Case where $k = 0$

Just as in PPLNS, we focus on the scenario where the bag is empty and strategic pool miners find a block. In terms of the equations above, this amounts to the case where $k = 0$.

Theorem 7.6. *If $k = 0$, and α, β are such that*

$$\alpha > \frac{1}{(D-1)^2}\left(\frac{ND}{N-1} + N - \beta(N-2)\right) = \Theta\left(\frac{N+D}{D^2}\right).$$

then S strictly dominates H.

Proof. We begin by computing the per-state surplus that S gives to H, $\Delta_i = S_i - H_i$, which gives us the following:

$$\begin{aligned}
\Delta_1 &= -\left(\frac{1}{N} + \frac{N-1}{N^2} + \frac{(N-1)^2}{N^2 D}\right), \\
\Delta_2 &= \frac{N-1}{N^2}, \\
\Delta_3 &= -\left(\frac{1}{N} + \frac{N-1}{N^2} + \frac{N-1}{N^2 D}\right), \\
\Delta_4 &= \frac{N-1}{N^2 D}, \\
\Delta_5 &= -\left(\frac{1}{N} + \frac{N-1}{ND}\right), \\
\Delta_6 &= 0.
\end{aligned} \tag{7.10}$$

Given these expressions, we are interested in the scenarios where $\sum_{i=1}^{6} p_i \Delta_i > 0$. Given the fact that $N, D \geq 0$, this is equivalent to $ND^2 \sum_{i=1}^{6} p_i \Delta_i > 0$. This leads to the following condition on α for hoarding to dominate revealing a block.

$$\alpha > \frac{1}{(D-1)^2}\left(\frac{ND}{N-1} + N - \beta(N-2)\right) = \Theta\left(\frac{N+D}{D^2}\right)$$

□

In summary, we have shown that as in PPLNS, if $\alpha = \Omega\left(\frac{N+D}{D^2}\right)$, then hoarding a block when up against an empty bag is strictly better than publishing it immediately. As in with PPLNS, it is often the case that $D = \Theta(N)$, hence this tells us that in this scenario, if $\alpha = \Omega\left(\frac{1}{N}\right)$, then miners are strategic by hoarding blocks. This is indeed what our empirical recursion results show.

7.6 Conclusion and Future Directions

In this chapter we presented RPPLNS, a novel twist on the already popular "Pay-per-last-N-shares" (PPLNS) mining pool scheme used by the majority of the Bitcoin network. By suitably randomising PPLNS, we are able to maintain its strengths (fairness, variance reduction, pool-hop-proof-ness) while proving robustness guarantees for honest mining and reducing the underlying memory constraints of the protocol. There are several possible directions for future research on RPPLNS. In the following paragraphs, we touch upon the ones we consider the most fruitful.

Interpolating between RPPLNS and PPLNS: Queue-bag protocols. At its core, PPLNS and RPPLNS are quite similar. It is not difficult to see that for a given

value of N, one can interpolate between PPLNS and RPPLNS by considering a pool protocol $\mathcal{M}_{Q,N}$ that maintains a queue of length $Q \in [N]$, and when a share is kicked out of the queue, it is put in a bag of size $N - Q$ (As visualised in Figure 7.5). Clearly $\mathcal{M}_{1,N}$ is RPPLNS and $\mathcal{M}_{N,N}$ is PPLNS. One can show that such family of queue-bag protocols share similar fairness, variance reduction and pool-hop-proof properties of PPLNS and RPPLNS. For such protocols, it would be interesting to study their incentive compatibility and see whether there is an optimal choice of Q to be made in terms of variance reduction, incentive compatibility and memory usage minimisation.

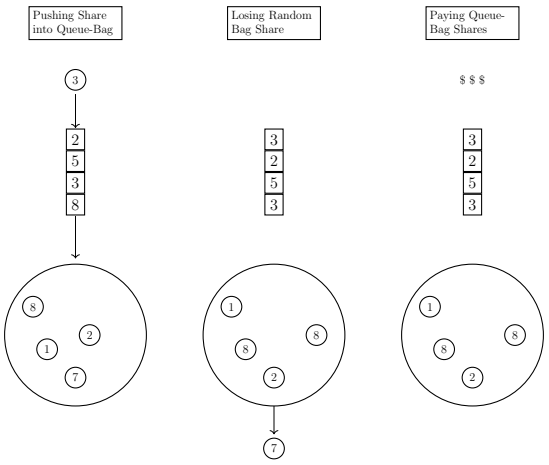

Figure 7.5: A single randomised transition for a queue-bag protocol. A miner sends share 3 to the pool operator, who pushes it into the initial queue. This causes share 8 to leave the queue and be pushed into the bag. In this case, share 7 was randomly selected to leave the bag to make room for share 8. Finally, all owners of shares in the queue and bag are paid $1/N$ for each such share.

Possible Benefits of Informational Fairness. We have already mentioned the fact that the reduced cardinality of the state space of RPPLNS vs. PPLNS can give rise to space usage gains for an implementation of the pool protocol. An interesting consequence of this fact is that since the state space of RPPLNS is smaller, it is also succinctly describable, and hence easy to communicate to all pool miners. It would be interesting to know that if a pool is operating a PPLNS mining protocol and there are two strategic agents m_1 and m_2 within the pool, where m_1 has full knowledge of the state of the queue $s \in S$ at any given moment of time and m_2 only has partial information, say some statistic, s' over s (this could be the queue considered as a bag for example, with order of elements in the queue forgotten), whether this gives m_1 any undue advantage over m_2. If this were the case, there would be a strong further justification for RPPLNS, as a way of putting all agents on an equal informational playing ground.

Stronger Incentive Guarantees. It would be great to rigorously understand the recurrence relation governing the gain of an optimal strategic pool miner in RPPLNS.

This would give us stronger results regarding incentive compatibility of pool miners and could glean insights into unforeseen strategic considerations.

Appendix A

Inequivalence of $\mathcal{Q}_U$ and $\mathcal{Q}_M$

Proposition A.1. *In the worst case, $\mathcal{Q}_U$ requires an exponential number of queries in the number of players to simulate a best response oracle with high probability.*

Proof. Suppose that we have n players. Let us define a family of games G_0^j and G_1^j for $1 \leq j \leq 2^{n-1}$ as follows:

- For all G_0^j and G_1^j, $\mathcal{A}_i = \{0, 1\}$ for each $i \in [n]$, and $\mathcal{A}_{-1} = \{a_1, ..., a_{2^{n-1}}\}$ for notational convenience.
- For all G_0^j and G_1^j and for all players $i \in [n]$ with $i > 1$, $U_i(a) = 0$ for all $a \in \mathcal{A}$.
- Suppose that $1 \leq j \leq 2^{n-1}$. In G_0^j the following hold: $U_1(0, a_k) = U_1(1, a_k) = 0$ for all $k \neq j$, $U_1(0, a_j) = 1$, and $U_1(1, a_j) = 0$.
- Suppose that $1 \leq j \leq 2^{n-1}$. In G_1^j the following hold: $U_1(0, a_k) = U_1(1, a_k) = 0$ for all $k \neq j$, $U_1(0, a_j) = 0$, and $U_1(1, a_j) = 1$.

In what follows, we focus on the specific distribution $x_{-1} = (1/2)_{i=2}^n$, which corresponds to the uniform distribution over $\mathcal{A}_{-1}$ (in fact any product distribution with full support over $\mathcal{A}_{-1}$ suffices). Clearly for any $1 \leq j \leq 2^{n-1}$, $BR_1(x_{-1})$ is $\{0\}$ for G_0^j, and $\{1\}$ for G_1^j.

Now let us suppose that $\mathcal{P}$ is any randomised protocol that makes use of $\mathcal{Q}_U$ and computes $BR_1(x_{-1})$ with a success probability greater than $2/3$ using at most α calls to $\mathcal{Q}_U$. We use Yao's minimax principle to show that in fact it must be the case that $\alpha = \Omega(2^n)$. Ultimately, since $BR(x_{-i})$ is always a singleton in our family of games, this demonstrates that even providing an adversarial best response oracle necessarily must use this many queries.

Suppose that $\mathcal{D}$ is the uniform distribution over inputs of the form G_h^j for $h \in \{0, 1\}$ and $j \in \{1, ..., 2^{n-1}\}$. It is straightforward to see that for a given input game G_h^j, any deterministic algorithm must make the query $(h, j) \in \mathcal{A}$ to $\mathcal{Q}_U$ to correctly determine

$BR_1(x_{-1})$. Therefore, under $\mathcal{D}$, if a deterministic algorithm makes at most α queries, it cannot succeed with probability larger than $\alpha/2^n$ on $\mathcal{D}$. Consequently, by Yao's minimax principle it must be the case that $\alpha = \Omega(2^n)$ as desired.

□

Bibliography

[1] Yaron Azrieli and Eran Shmaya. Lipschitz games. *Math. Oper. Res.*, 38(2):350–357, 2013.

[2] Yakov Babichenko. Best-reply dynamics in large binary-choice anonymous games. *Games and Economic Behavior*, 81:130–144, 2013.

[3] Yakov Babichenko. Query complexity of approximate Nash equilibria. *J. ACM*, 63(4):36:1–36:24, 2016.

[4] Yakov Babichenko, Siddharth Barman, and Ron Peretz. Empirical distribution of equilibrium play and its testing application. *Math. Oper. Res.*, 42(1):15–29, 2017.

[5] Yakov Babichenko and Aviad Rubinstein. Communication complexity of approximate Nash equilibria. In *Procs. of 49th STOC*, pages 878–889, 2017.

[6] Qianlan Bai, Xinyan Zhou, Xing Wang, Yuedong Xu, Xin Wang, and Qingsheng Kong. A deep dive into blockchain selfish mining. Cryptology ePrint Archive, Report 2018/1084, 2018. https://eprint.iacr.org/2018/1084.

[7] Elizabeth Baldwin, Paul W. Goldberg, Paul Klemperer, and Edwin Lock. Solving strong-substitutes product-mix auctions. *CoRR*, abs/1909.07313, 2019.

[8] Elizabeth Baldwin and Paul Klemperer. Understanding preferences: "demand types", and the existence of equilibrium with indivisibilities. LSE Research Online Documents on Economics 63198, London School of Economics and Political Science, LSE Library, October 2016.

[9] Keith Ball. An elementary introduction to modern convex geometry. In Silvio Levi, editor, *Flavors of Geometry*, chapter 1, pages 1–58. Cambridge University Press, Cambridge, 1997.

[10] Ulrich Berger. Brown's original fictitious play. *Journal of Economic Theory*, 135(1):572–578, 2007.

[11] Georgios Birmpas, Elias Koutsoupias, Philip Lazos, and Francisco J. Marmolejo Cossío. Fairness and efficiency in dag-based cryptocurrencies. *CoRR*, abs/1910.02059, 2019.

[12] Hartwig Bosse, Jaroslaw Byrka, and Evangelos Markakis. New algorithms for approximate Nash equilibria in bimatrix games. *Theoretical Computer Science*, 411(1):164–173, 2010.

[13] George W. Brown. Some notes on computation of game solutions. *RAND corporation report*, page 78, April 1949.

[14] Nader H. Bshouty, Paul W. Goldberg, Sally A. Goldman, and H. David Mathias. Exact learning of discretized geometric concepts. *SIAM J. Comput.*, 28(2):674–699, 1998.

[15] 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.

[16] Nicolas T. Courtois and Lear Bahack. On subversive miner strategies and block withholding attack in bitcoin digital currency. *arXiv preprint arXiv:1402.1718*, 2014.

[17] Constantinos Daskalakis, Paul W.. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. SIAM J. Comput., 39(1):195–259, 2009.

[18] Constantinos Daskalakis, Aranyak Mehta, and Christos Papadimitriou. A note on approximate Nash equilibria. *Theoretical Computer Science*, 410(17):1581–1588, 2009.

[19] Constantinos Daskalakis and Qinxuan Pan. A counter-example to Karlin's strong conjecture for fictitious play. In *Procs. of 55th FOCS*, pages 11–20, 2014.

[20] Ittay Eyal. The miner's dilemma. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 89–103. IEEE, 2015.

[21] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-NG: a scalable blockchain protocol. In *Procs. of 13th NSDI*, pages 45–59, 2016.

[22] Ittay Eyal and Emin Gün Sirer. Majority is not enough: bitcoin mining is vulnerable. In *Procs. of 18th International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.

[23] John Fearnley, Martin Gairing, Paul W.. Goldberg, and Rahul Savani. Learning equilibria of games via payoff queries. *J. Mach. Learn. Res.*, 16:1305–1344, 2015.

[24] John Fearnley and Rahul Savani. Finding approximate Nash equilibria of bimatrix games via payoff queries. In *Procs. of 15th ACM EC*, pages 657–674, 2014.

[25] Fabrizio Germano and Gabor Lugosi. Global Nash convergence of Foster and Young's regret testing. *Games and Economic Behavior*, 60(1):135–154, 2007.

[26] Paul W. Goldberg and Stephen Kwek. The precision of query points as a resource for learning convex polytopes with membership queries. In *Procs. of 13th COLT*, pages 225–235. Citeseer, 2000.

[27] Paul W. Goldberg and Francisco J. Marmolejo-Cossío. Learning convex partitions and computing game-theoretic equilibria from best response queries. In *International Conference on Web and Internet Economics*, pages 168–187. Springer, 2018.

[28] Paul W. Goldberg, Francisco J. Marmolejo-Cossío, and Zhiwei Steven Wu. Logarithmic query complexity for approximate Nash computation in large games. *Theory of Computing Systems*, 63(1):26–53, 2019.

[29] Paul W. Goldberg and Aaron Roth. Bounds for the query complexity of approximate equilibria. *ACM Trans. Economics and Comput.*, 4(4):24:1–24:25, 2016.

[30] Paul W. Goldberg, Rahul Savani, Troels Bjerre Sørensen, and Carmine Ventre. On the approximation performance of fictitious play in finite games. *Int. J. Game Theory*, 42(4):1059–1083, 2013.

[31] Paul W.. Goldberg and Stefano Turchetta. Query complexity of approximate equilibria in anonymous games. *Journal of Computer and System Sciences*, 90:80–98, 2017.

[32] Philip A. Haile, Ali Hortaçsu, and Grigory Kosenok. On the empirical content of quantal response equilibrium. *American Economic Review*, 98(1):180–200, 2008.

[33] 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.

[34] Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, September 2000.

[35] Sergiu Hart and Andreu Mas-Colell. Uncoupled dynamics do not lead to Nash equilibrium. *The American Economic Review*, 93(5):1830–1836, dec 2003.

[36] Sergiu Hart and Noam Nisan. The query complexity of correlated equilibria. *Games and Economic Behavior*, 2016.

[37] Michael D. Hirsch, Christos H. Papadimitriou, and Stephen A. Vavasis. Exponential lower bounds for finding brouwer fix points. *Journal of Complexity*, 5(4):379–416, 1989.

[38] Fritz John. *Extremum Problems with Inequalities as Subsidiary Conditions*, pages 197–215. Springer Basel, Basel, 2014.

[39] James S. Jordan. Three problems in learning mixed-strategy Nash equilibria. *Games and Economic Behavior*, 5(3):368–386, 1993.

[40] Shizuo Kakutani. A generalization of Brouwer's fixed point theorem. *Duke Math. J.*, 8(3):457–459, 09 1941.

[41] Ehud Kalai. Large robust games. *Econometrica*, 72(6):1631–1665, 2004.

[42] Debarun Kar, Fei Fang, Francesco M. Delle Fave, Nicole Sintov, Milind Tambe, and Arnaud Lyet. Comparing human behavior models in repeated Stackelberg security games: An extended study. *Artificial Intelligence*, 240:65–103, 2016.

[43] Michael Kearns, Mallesh M. Pai, Aaron Roth, and Jonathan Ullman. Mechanism design in large games: Incentives and privacy. *American Economic Review*, 104(5):431–35, May 2014.

[44] Aggelos Kiayias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tselekounis. Blockchain mining games. In *Procs. of 17th ACM EC*, pages 365–382, 2016.

[45] Paul Klemperer. The product-mix auction: a new auction design for differentiated goods. *Journal of the European Economic Association*, 8(2/3):526–536, 2010.

[46] Spyros C. Kontogiannis and Paul G. Spirakis. Well supported approximate equilibria in bimatrix games. *Algorithmica*, 57(4):653–667, 2010.

[47] Yujin Kwon, Dohyun Kim, Yunmok Son, Eugene Vasserman, and Yongdae Kim. Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin. In *Procs. of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 195–209, 2017.

[48] Philip Lazos, Elias Koutsoupias, Foluso Ogunlana, and Paolo Serafino. Blockchain mining games with pay forward. In *The World Wide Web Conference*, pages 917–927, 2019.

[49] Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Playing large games using simple strategies. In *Procs. of the 4th ACM EC*, pages 36–41, 2003.

[50] Hanqing Liu, Na Ruan, Rongtian Du, and Weijia Jia. On the strategy and behavior of bitcoin mining with N-attackers. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 357–368. ACM, 2018.

[51] Francisco J Marmolejo-Cossío, Eric Brigham, Benjamin Sela, and Jonathan Katz. Competing (semi-) selfish miners in bitcoin. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 89–109, 2019.

[52] Richard D McKelvey and Thomas R. Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1):6–38, 1995.

[53] Dov Monderer and Lloyd S. Shapley. Fictitious play property for games with identical interests. *Journal of economic theory*, 68(1):258–265, 1996.

[54] Dov Monderer and Lloyd S. Shapley. Potential games. *Games and economic behavior*, 14(1):124–143, 1996.

[55] John H Nachbar. "evolutionary" selection dynamics in games: Convergence and limit properties. *International journal of game theory*, 19(1):59–89, 1990.

[56] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[57] John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

[58] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: generalizing selfish mining and combining with an eclipse attack. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 305–320. IEEE, 2016.

[59] Thanh Hong Nguyen, Rong Yang, Amos Azaria, Sarit Kraus, and Milind Tambe. Analyzing the effectiveness of adversary modeling in security games. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[60] Christos H. Papadimitriou and Tim Roughgarden. Computing correlated equilibria in multi-player games. *Journal of the ACM*, 55(3), July 2008.

[61] Dean P.Foster and H. Peyton Young. Regret testing: learning to play Nash equilibrium without knowing you have an opponent. *Theoretical Economics*, 1(3):341–367, sep 2006.

[62] Rui Qin, Yong Yuan, and Fei-Yue Wang. A novel hybrid share reporting strategy for blockchain miners in PPLNS pools. *Decision Support Systems*, 118:91–101, 2019.

[63] Julia Robinson. An iterative method of solving a game. *The Annals of Mathematics*, 54(2):296–301, 1951.

[64] Meni Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

[65] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Procs. of 20th International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.

[66] Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In *International Conference on Financial Cryptography and Data Security*, pages 477–498. Springer, 2016.

[67] Lloyd Shapley. Some topics in two-person games. *Advances in game theory*, 52:1–29, 1964.

[68] Eran Shmaya. Brouwer implies Nash implies Brouwer. *The Leisure of the Theory Class, http://theoryclass. wordpress. com/2012/01/05/brouwer-implies-nash-implies-brouwer*, 2012.

[69] Dale O. Stahl II and Paul W. Wilson. Experimental evidence on players' models of other players. *Journal of economic behavior & organization*, 25(3):309–327, 1994.

[70] Roger Wattenhofer. *Blockchain Science: Distributed Ledger Technology*. Inverted Forest Publishing, 2019.

[71] James R. Wright and Kevin Leyton-Brown. Beyond equilibrium: Predicting human behavior in normal-form games. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[72] Rong Yang, Fernando Ordonez, and Milind Tambe. Computing optimal strategy against quantal response in security games. In *AAMAS*, pages 847–854, 2012.

[73] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *Procs. of 18th FOCS*, pages 222–227. IEEE, 1977.

[74] Stephen T. Yau and Letian Zhang. An upper estimate of integral points in real simplices with an application to singularity theory. *Mathematical Research Letters*, 13(6):911–921, 2006.

[75] H. Peyton Young. Learning by trial and error. *Games and economic behavior*, 65(2):626–643, 2009.

[76] Yevhen Zolotavkin, Julian García, and Carsten Rudolph. Incentive compatibility of pay per last N shares in bitcoin mining pools. In *International Conference on Decision and Game Theory for Security*, pages 21–39. Springer, 2017.