The Dissertation Committee for Nedialko Boyanov Dimitrov
certifies that this is the approved version of the following dissertation:

Coping with Dynamic Membership, Selfishness, and Incomplete Information: Applications of Probabilistic Analysis and Game Theory

Committee:

C. Greg Plaxton, Supervisor

Adam Klivans

David Morton

Peter Stone

David Zuckerman

Coping with Dynamic Membership, Selfishness, and Incomplete Information: Applications of Probabilistic Analysis and Game Theory

by

Nedialko Boyanov Dimitrov, B.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2008

To my family, friends and mentors: Thank you. – Ned.

Acknowledgments

A special thank you to Greg Plaxton, for spending many hours guiding me to ultimately produce all the work in this dissertation.

Nedialko Boyanov Dimitrov

The University of Texas at Austin
May 2008

Coping with Dynamic Membership, Selfishness, and Incomplete Information: Applications of Probabilistic Analysis and Game Theory

Publication No. _____

Nedialko Boyanov Dimitrov, Ph.D.
The University of Texas at Austin, 2008

Supervisor: C. Greg Plaxton

The emergence of large scale distributed computing networks has given increased prominence to a number of algorithmic concerns, including the need to handle dynamic membership, selfishness, and incomplete information. In this document, we outline our explorations into these algorithmic issues.

We first present our results on the analysis of a graph-based coupon collec-

tor process related to load balancing for networks with dynamic membership. In addition to extending the study of the coupon collector process, our results imply load balancing properties of certain distributed hash tables.

Second, we detail our results on worst case payoffs when playing buyer-supplier games, against many selfish, collaborating opponents. We study optimization over the set of core vectors. We show both positive and negative results on optimizing over the cores of such games. Furthermore, we introduce and study the concept of focus point price, which answers the question: If we are constrained to play in equilibrium, how much can we lose by playing the wrong equilibrium?

Finally, we present our analysis of a revenue management problem with incomplete information, the online weighted transversal matroid matching problem. In specific, we present an algorithm that delivers expected revenue within a constant of optimal in the online setting. Our results use a novel algorithm to generalize several results known for special cases of transversal matroids.

Contents

Chapter 1

Three Algorithmic Concerns

1.1 Introduction

The emergence of large scale distributed computing networks has given prominence to a number of algorithmic concerns. First, algorithms should handle dynamic membership. In a large scale system, it is undesirable to have preallocated, central, coordinating nodes for controlling membership. Such nodes limit the scalability of the network and are bottle necks of communication. Second, algorithms should handle selfish behavior on behalf of the computational nodes. For example, if the network is spread across machines owned by several entities, each entity may have its selfish interests override the interests of the community. Third, the algorithms should work even when there is a lack of total information. For example, when running an algorithm based on input from many participants, the algorithm may not have access to the full input before a decision is required. Instead, the algorithm would need to make on the spot, near-optimal decisions as input becomes available.

In this document, we present some methods of dealing with the algorithmic issues of distributed computing networks. First, using probabilistic analysis, we consider a random process with close connections to load balancing in distributed networks. We show some desirable properties of the random process that directly lead to a load balancing scheme in distributed networks with dynamic membership. Second, using game theory, we analyze a class of games called buyer-supplier games. We consider the case where players can selfishly collude in an unrestricted fashion and derive algorithms for finding worst case payoffs in the selfish environment. Third,

we present an algorithm for finding near-optimal matchings in weighted transversal matroids. The algorithm works without full knowledge of the input and finds applications in search engine auctions, scheduling, and other domains.

The rest of the document is structured as follows. In the rest of this chapter, we give some context and discussion of the results presented in the document. In Section 1.2, we briefly describe our results on load balancing in distributed networks. The full results are then presented in Chapter 2. In Section 1.3, we review some game theoretic preliminaries and discuss our work on buyer-supplier games. The full results are then presented in Chapter 3. In Section 1.4, we detail some of the connections between probabilistic analysis and game theory that lead to our work on the online matroid problem. We present our full results on the online matroid problem in Chapter 4. Finally, in Chapter 5, we provide some concluding remarks and discussion on future directions.

1.2 Dynamic Membership

In this section, we discuss our results on analyzing graph-based variants of the classic coupon collector process. To provide context, in Section 1.2.1, we present some history of the applications and variants of the coupon collector process. Then, in Section 1.2.2, we discuss our results on several graph-based variants and related load balancing applications.

1.2.1 Coupon Collector Processes

One of the most commonly discussed stochastic processes in computer science is the so-called coupon collector process [26, 40]. In the coupon collector process, there are n distinct coupons in a bag. The process proceeds in rounds, where in each round we select a uniformly random coupon from the bag and replace it back into the bag. The coupon collector process has been studied extensively. For example, we know that $\Theta(n \log n)$ rounds are required, with high probability, to see each coupon at least once. We also know that after n rounds, we would have seen each coupon no more than $O(\log n / \log \log n)$ times with high probability [25]. The coupon process can also be described as as placing balls uniformly at random into bins, where each bin represents a coupon and placing a ball into a bin represents selecting that coupon.

One application of the coupon collector process is load balancing in distributed systems. For example, imagine a distributed system with n servers. Furthermore, imagine that jobs arrive into the system and each job is assigned to a server. We call the number of jobs assigned to a server the server's *load*. The coupon collector results described in the previous paragraph immediately yield the result that if n jobs are assigned uniformly at random to the n servers, then the maximum load in the system is $O(\log n / \log \log n)$ with high probability. The coupon collector process also finds applications in hashing, Markov chain mixing, Monte Carlo methods and other areas [25, 37, 48].

Study of coupon collector processes goes as far back as de Moivre's work of 1712 [26]. Since then, numerous properties and variants of coupon collector processes have been considered. For example, Erdös and Rényi bound the number of rounds required to draw each of the n coupons at least m times [17]. Their result is generalized by Kaplan, who bounds the number of rounds required for l of the n coupons to be seen at least m times [29]. More recently, Azar et al., using the balls and bins interpretation of the process, and consider a variant where in each round we choose two bins uniformly at random and place the ball in the least loaded bin [7]. They show that in this variant, the maximally loaded bin after n rounds has $\log \log n / \log 2 + O(1)$ balls with high probability, an exponential decrease in the maximum load of the original process. Mitzenmacher extensively studies this multi-choice process variant and its applications to load balancing in his thesis [39].

1.2.2 A Graph-Based Coupon Collector Process

Similar to Mitzenmacher, we also study several variants of the coupon collector process. The variants we study take place on a graph and find applications in load balancing for distributed hash tables.

A distributed hash table (DHT), is a special type of distributed network with a primary focus on data storage. In a DHT, every machine is in charge of a fraction of the storage namespace; this fraction of the namespace is the machine's load. A typical DHT connects n machines so that each machine has $\Theta(\log n)$ neighbors. One method for the $(n+1)$st machine to join the DHT consists of the following three steps:

- Select a uniformly random machine from the n machines in the DHT.

- Query the selected machine's neighbors for their loads.
- Join the network by splitting the load of the most heavily loaded neighbor.

Adler et al. have shown how to use similarities between the described join procedure and a graph-based coupon collector process to prove load balance guarantees in the distributed hash table [1].

Consider the following graph-based coupon collector process. Initially, all the nodes of the graph are uncovered. The process then proceeds in rounds. In each round, we first select a uniformly random node and cover one of the selected node's uncovered neighbors. Of course, exactly how we choose which uncovered neighbor to cover leads to different variants of the graph-based coupon collector process. If we imagine that a DHT is formed by the nodes of the graph, one can relate covering a node in the process to splitting the load of the node in the DHT. Adler et al. use this relationship to show that $O(n)$ cover time, where n is the number of nodes in the graph, in such a graph-based coupon collector process lead to load balance guarantees in DHTs.

To be more specific, the cover time of a graph-based coupon collector process is a random variable indicating the round on which the last uncovered node of the graph is covered. Using the approach of Adler et al., an $O(n)$ cover time be used to show that, under the described DHT join procedure, if the DHT initially has a single machine with a load of 1, then after $n-1$ machines join the DHT, the load of every machine is $O(\frac{1}{n})$ with high probability. Thus, up to a constant factor, no machine is loaded with more than its fair share of the total DHT load. For the load-balance guarantees to hold, the $O(n)$ cover time result must hold both in expectation and with high probability.

Adler et al. have shown that picking a uniformly random uncovered neighbor when the process takes place on the hypercube leads to an $O(n)$ cover time. The proofs of $O(n)$ cover time of Adler et al. strongly depend on the structure of the hypercube, and thus the resulting load balance results apply only to hypercubic DHTs.

In Chapter 2, we analyze several graph-based coupon collector processes on arbitrarily structured graphs. For several natural variants, we show that if all nodes of the graph have degree $\Theta(\log n)$, regardless of the specific graph structure, then the cover time of the process is $\Theta(n)$ with high probability. We show that the

same asymptotic bound holds for the expected cover time. See Theorem 2.4.1 and Lemma 2.5.2 for a precise statement of results. Using the connection between cover time and load balance due to Ader et al., our cover time results directly imply load balance guarantees to arbitrarily structured DHTs, as long as the DHT nodes have $\Theta(\log n)$ neighbors, as is the case with most common DHTs [35].

To show our cover time results, we use what is known as a delay sequence argument as our argument scaffold [34]. The earliest instances of the delay sequence argument in the literature appear in the works of Aleliunas and Upfal on parallel communication [3, 58]. The delay sequence argument is used to prove that a random process completes in a certain time bound, say $T(n)$, and works as follows. First, we identify a set of problem specific combinatorial objects called delay sequences. Second, we define the notion of an i-active delay sequence, which connects the combinatorial object to a particular round in the process. Third, we argue that if the process completes in round j, a j-active delay sequence is created. Finally, we argue that there is no $\omega(T(n))$-active delay sequence with high probability. These steps let us conclude that the process completes in time $O(T(n))$ with high probability.

In addition to showing fast cover time results for several variants of the graph coupon collector process, in Chapter 2 we also show that an arbitrary method of choosing which uncovered neighbor to cover does not result in $O(n)$ cover time. Specifically, we show that under a natural process variant there exists a graph such that each node in the graph has degree $\Theta(\log n)$ but the graph-based coupon collector process has $\Omega(n\sqrt{\log n/\log\log n})$ cover time with high probability. In other words, we show that the method in which we choose which uncovered neighbor to cover does matter. See Theorem 2.6.1 for a precise statement of results.

1.3 Selfishness

In this section, we present our results on optimization over the core of buyer-supplier games. But, before we can introduce the results in detail, in Sections 1.3.1 and 1.3.2, we present some of the required game theoretic preliminaries. In Section 1.3.1, we informally introduce the definition of a game. In Section 1.3.2, we informally introduce the notion of a solution concept for a game by discussing several example solution concepts. Then, we present our results on optimizing over the core of buyer-supplier games in Sections 1.3.3 and 1.3.4.

1.3.1 Game Theoretic Preliminaries

A game is a strategic interaction between a group of players. Every game has a set of possible outcomes. The particular outcome realized by playing the game depends on the actions selected by the game's players. Each player has a utility, also known as a payoff, associated with each outcome. Each player's goal is to take actions that maximize their payoff in the game's final, realized outcome.

There are two basic ways of specifying games. The first is called the extensive form, or game tree. Each level of the tree designates a point in time. The game begins at the root level and continues to the leaves of the tree, where payoffs are determined. Each node in the tree represents a decision available to the players. The game described by the tree is sequential in the sense that the strategic options available at level i in the tree are determined by the decisions made on levels 1 through $i - 1$.

The second basic way of specifying a game is called the normal form. Contrary to extensive form games, which have an intuitive, built-in, sequential nature, normal form games are single-shot, simultaneous decision games. In other words, in a normal form game, each player makes a single decision and the combined decisions of all the players specify the game's realized outcome.

It is possible to express extensive form games as normal form games. To transform an extensive form game to a normal form game, we have the single decision made by each player in the normal form game specify a strategy for all nodes in the extensive form game tree. Once the simultaneous decisions are made in the normal form game, there is a unique, fixed path from the root of the game tree to the game's outcome; thus, the payoffs are determined.

We restrict our attention to normal form games. Since extensive form games can be expressed as normal form games, we do not eliminate any games by restricting our attention to normal form games. More explicitly, a normal form game is defined as follows. First, the game has a set $\mathcal{P}$ of players. Each player in $\mathcal{P}$ has an associated set of actions, called a strategy set, as well as a utility function which determines the player's payoff given the actions chosen by all players in the game. Once the players, the strategy sets, and the utility functions are specified, the normal form game is fully specified. Figure 1.1 depicts a typical normal form game.

Row \ Col	C	D
C	5 5	10 −10
D	−10 10	−5 −5

Figure 1.1: A typical normal form game. The players are {Row,Col}. The strategy set for both Row and Col happens to be the same, namely {C,D}. The utility functions listed in the matrix specify the payoff to each player, given the vector of strategies chosen by both players. For example, the payoff to the Row player when Row chooses C and Column chooses D is −10.

1.3.2 Introduction to Solution Concepts

In game theory, a solution concept predicts the outcome of a game; the predictions made by the solution concept are called equilibria. Different models of game play lead to different solution concepts. For example, the famous Nash equilibrium solution concept was developed within the framework alluded to in the previous section: players make strategic decisions simultaneously, without any inter-player communication [56, p. 240]. Intuitively, in a Nash equilibrium, each player is selfishly selecting the best response, personal payoff maximizing strategy. A set of best response strategies can be complementary such that no player wishes to unilaterally change strategy. To illustrate this, in Figure 1.1 fix the Strategy D for the Row player and the Strategy D for the Col player. Given this fixed set of strategies, it is to neither player's advantage to unilaterally change strategy. If we find such a set of complementary best response strategies, we have found a Nash equilibrium. Every normal form game has a Nash equilibrium in mixed strategies, where a mixed strategy is a probability distribution over the player's strategy set.

A stronger solution concept than Nash equilibrium comes from the definition of a strongly dominated strategy. We say that Strategy 1 strongly dominates Strategy 2 for some player if, regardless of the actions of the remaining players, Strategy 1 always gives a higher payoff than Strategy 2. For example, in Figure 1.1, the Strategy D strongly dominates Strategy C for the Row player. Regardless of what Col plays, Row always receives a higher payoff from playing D than C. Thus, C is not a reasonable strategy for Row and we can eliminate C from the possible

choices for Row. If iterative elimination of strongly dominated strategies leads to a single outcome, we have found a dominant strategy equilibrium. All dominant strategy equilibria are Nash equilibria, but the converse is not true. Because of the associated strong requirements, dominant strategy equilibria do not always exist.

As a third and final example of a solution concept, we present the core, which was developed with the notion that players may collude by communicating before playing the game, and reallocating payoffs at the end of the game. Intuitively, if no collusion is allowed, only some strategies are rational, since an individual player never accepts a payoff that the player can unilaterally and selfishly improve. Allowing some sets of players to collude further restricts the set of rational strategies. A group of players does not accept a total payoff that is less than the group can achieve through collaboration. The core is the set of payoff vectors resulting from strategies that are rational when we allow every subset of the players to collude. Again, because of the strong rationality requirements, the core of a game may be empty.

For a more explicit example of the core, consider a game involving three players, $\{A, B, C\}$. Each subset of the players, if allowed to collude, can unilaterally guarantee some payoff, regardless of the remaining player's actions. Suppose that $\{A\}$, $\{B\}$, $\{C\}$, $\{AB\}$, $\{AC\}$, $\{BC\}$, $\{ABC\}$ can unilaterally guarantee payoffs of 3, 2, 2, 10, 10, 10, 37, respectively. The core is then the set of payoff vectors where each subset of players receives at least as much payoff as the subset can guarantee, and we give out no more payoff than can be earned in the game. In our example, the payoff vector $(10, 10, 17)$, in alphabetical order of player name, is in the core since every subset of the players receives at least as much as it can guarantee and we give out a total of 37, which can be earned in the game. Similarly, the payoff vectors $(12, 15, 10)$ and $(3, 27, 7)$ are also in the core. However, the payoff vector $(10, 10, 16)$ is not in the core, since the subset of players $\{ABC\}$ does not receive as much as it can guarantee. The payoff vector $(3, 28, 7)$ is also not in the core, because, even if everyone colludes, the players cannot achieve a total payoff of 38.

1.3.3 Buyer-Supplier Games

The connections between computer science and game theory can be categorized in two ways.

1. The transformation of optimization problems studied in computer science into games. For example, Nisan and Ronen study scheduling under a truthful mechanism setting [42]. Also, Archer and Tardos devise truthful mechanisms for several combinatorial problems where the players' secret information is a single positive real number [6].

2. Algorithmic investigations into game theoretic concepts. For example, the work of Daskalakis et al. classifies the computational complexity of Nash equilibrium [13]. Also, Deng and Papadimitriou study the computability of the Arrow-Debreu pricing equilibrium [15].

In Chapter 3, we present our results on optimization over the core of a large class of games called buyer-supplier games. As such, our work fits into both categories. A buyer-supplier game can be derived from most combinatorial minimization problems, thus the work fits into the first category. Given a game theoretic characterization of the core, the study of efficient optimization over the core fits into the second category.

Buyer-supplier games, a subset of assignment games, have been studied for decades. Shapley and Shubik originally characterized the core of buyer-supplier games [55]. Since the work a Shapley and Shubik, there has been a series of works both in the economics and computer science literature on buyer-supplier games [10, 18, 22].

We introduce buyer-supplier games with a specific example. The buyer-supplier minimum spanning tree (MST) game is a buyer-supplier game derived from the MST minimization problem. The set of players for the MST buyer-supplier game consists of a set of suppliers, one for each edge, and a single buyer player. Each supplier owns their corresponding edge, and the buyer's goal is to purchase a spanning tree from the suppliers. Consequently, a strategy for a supplier is a single positive real number representing a bid submitted by the supplier to the buyer. A strategy for the buyer is a subset of the suppliers, representing the set of suppliers chosen by the buyer for a purchase. To model a real situation accurately, the utility functions are defined as follows. Each supplier has an internal cost associated with supplying goods to the buyer. The buyer has a fixed maximum amount of money, M, they are willing to spend on purchasing services from the suppliers. The payoff for the buyer is 0 if the edges chosen do not span the graph, and is otherwise equal to

the remainder of the M dollars, after payments to the chosen suppliers are complete. The payoff for a supplier is 0 if the supplier is not chosen by the buyer, and equal to the supplier's bid otherwise.

The MST buyer-supplier game lends itself to a natural interpretation. A company owns factories on every node of a graph. The company wishes to connect the factories by purchasing edges in the graph. Each edge is owned by a unique supplier player. Each supplier has an internal cost associated with the company's usage of the edge. The company has a maximum amount of money it is willing to spend on purchasing edges. The buyer-supplier game paradigm yields similarly natural games when applied to other minimization problems. For example, Steiner tree, shortest path, minimum set cover, minimum cut, facility location, single- and multi-commodity flow can all be used to instantiate a buyer-supplier game. A general method of transforming a combinatorial minimization problem to a buyer-supplier game is presented in Chapter 3.

1.3.4 Optimization Over the Core of Buyer-Supplier Games

We informally introduced the core solution concept in Section 1.3.2. Unlike Nash equilibrium, which is usually defined by a vector of strategies, the core is usually defined by a vector of payoffs. Furthermore, the core can be defined under two distinct settings: the transferable utility (TU) setting, and the non-transferable utility (NTU) setting. In the TU setting, we expand each player's strategy set to include the transfer of any amount of their own utility to other players of the game. In the NTU setting, such utility transfers are disallowed.

Shapley and Shubik developed the definition of the core and showed that for buyer-supplier games, the core in the TU setting can be characterized by an exponentially sized set of linear inequalities on the payoff vector [55]. In Chapter 3, we show the surprising result that the same characterization holds in the NTU setting. The proof is based on the intuitive observation that transfers of utility can be simulated in the NTU setting by an adjustment to the supplier bids.

In our investigations, we introduce the concept of *focus point price*. Solution concepts often yield multiple predictions, or equilibria. For example, there may be multiple vectors in the core. In actual game play, however, only one of the equilibria can be chosen by the players. Experiments show that conditions outside the game,

such as societal pressures or undue attention to a specific player, focus the players' attention on the point of a single equilibrium. A classic example is given by a boardroom meeting, where a volunteer is needed to perform an undesirable task. In real situations, everyone stays perfectly still in the hope of that the group's focus will not fall on them.

The focus point price measures the value of a good focus point. In other words, the focus point price is the difference between the best and worst equilibrium outcome, where the terms best and worst are based on some desirable criterion. Stated succinctly, the focus point price answers the question: If we are constrained to play in equilibrium, how much can we lose by playing the wrong equilibrium?

The focus point price concept motivates our examination of optimization over the core of buyer-supplier games. We concentrate on the focus point price from the buyer's perspective. In other words, we try to answer the question: What is the difference between the best and worst possible core outcome for the buyer? For the rest of the document, when we say the *buyer's focus point price*, we mean this specific instance of the focus point price concept. In Chapter 3, we give two main computational results, with applications to general buyer supplier games.

First, we give a positive computational result by exhibiting a separation oracle over the exponential set of equations defining the core. Given a separation oracle, the ellipsoid algorithm can be used to solve linear optimizations over the exponential set of equations. The ellipsoid algorithm only calls the separation oracle a polynomial number of times. Thus, if the separation oracle runs in polynomial time, we can optimize linear functions over the exponential set of equations in polynomial time. For a combinatorial minimization problem and the corresponding buyer-supplier game, we are able to show that if it is possible to solve the minimization problem in polynomial time, then it is possible to optimize linear functions over the core of the game in polynomial time. See Theorem 3.4.1 for the precise statement of results.

Second, we give a complementary result, which can be used to show hardness of optimization over the core. For a combinatorial minimization problem and the corresponding buyer-supplier game, by using several polynomial time reductions, we can show that solving the minimization problem is polynomial time equivalent to optimization over the core of the game. See Theorem 3.5.8 for the precise statement of results. In the process of showing Theorem 3.5.8, we also derive strong negative results for computing the buyer's focus point price. We show that if the underlying

minimization problem is not solvable in polynomial time, then it is not possible to approximate the buyer's focus point price to within *any* multiplicative factor in polynomial time.

Finally, to complement our general results on optimization over the core of buyer-supplier games, we also exhibit a simple extension of Kruskal's algorithm that can be used to compute the buyer's focus point price in the buyer-supplier MST game.

1.4 Incomplete Information

In this section, we present our results on incomplete information settings. To provide context, before we discuss our results, we explore some of the intimate ties between probabilistic analysis and game theory that result from modeling incomplete information. In Sections 1.4.1, we give an example of how incentives can lead players not to reveal their private information, leading to incomplete information. In 1.4.2, we discuss some of the main approaches to dealing with incomplete information in games, which include modeling a lack of knowledge as a probability distribution over the possibilities. In Section 1.4.3, we consider the connections between random processes and games to further reinforce the connections between game theory and probabilistic analysis. Finally, in Section 1.4.4, we discuss our work on the online matroid problem, a problem in a random process setting with incomplete information.

1.4.1 Private Information Leads to Incomplete Information

The application of any of the solution concepts in Section 1.3.2, and indeed a player's decision making process as they try to maximize their payoff, depends on knowing the utilities of all the players in the game. However, in real applications, it is typical for each player to know only their own utility function. Each player depends on outside information to get a sense of the other players' utilities. Because of this dependence on outside information, it may be advantageous for players not to reveal their true evaluation of the game's outcomes.

To illustrate the benefits of not revealing private information, consider the "Chicken" game, displayed in Figure 1.2a. In this game, two cars drive directly at each other. Each car has the choice of driving forward or swerving. If they

Car1 \ Car2	Swerve	Drive
Swerve	-10, -10	10, -30
Drive	-30, 10	-100, -100

(a) The Chicken Game

Car1 \ Car2	Swerve	Drive
Swerve	-5, -10	10, -30
Drive	-5, 10	10, -100

(b) Tough Man Chicken Game

Figure 1.2: The normal form game in 1.2a is the "Chicken" game. In this game two cars square off and drive directly at each other. Each driver has the choice to continue driving forward or swerve. The normal form game in 1.2b is the "Tough Man Chicken" game. In this game Car2 has convinced Car1 that Car2 only cares about its honor – driving is always preferable to swerving.

both drive forward, they crash and suffer terrible losses. If one swerves but the other drives, the swerver suffers humiliation and other car declares victory. Finally, if both cars swerve, both suffer mild humiliation. Applying the Nash equilibrium solution concept, we find two equilibria at {Swerve, Drive} and {Drive, Swerve}.

Suppose that Car2 manages to convince Car1 that Car2 only cares about its honor. In other words, Car1 believes that regardless of Car1's actions, Car2 always prefers to avoid humiliation and drive forward. This "Tough Man Chicken" game is illustrated in Figure 1.2b. In this game there is only a single Nash equilibrium, {Swerve, Drive}. In fact, {Swerve, Drive} also satisfies the stronger notion of a dominant strategy equilibrium. Thus, Car2 ensures victory in the game by convincing Car1 of its toughness.

Since players often benefit from preserving the secrecy of their true evaluations of the game's outcomes, to develop a truly applicable game theory, we must find ways to deal with this "problem of private information."

1.4.2 Addressing Incomplete Information

In this section, we detail how handling incomplete and private information leads to the introduction of probabilistic analysis in games. In particular we detail two main approaches to dealing with private information.

One approach to dealing with private information is through the design of

truthful mechanisms. We call a game truthful if it is to every player's advantage to reveal all private information. A truthful mechanism is a truthful game in which the equilibrium outcomes have some desired property. For the purposes of this discussion, we are interested in the truthfulness property, since it removes the problem of private information.

A simple example of a truthful mechanism is the Vickrey auction, which is also called a second price sealed bid auction. In this game the players are bidding to purchase an item. Each player may bid a non-negative amount for the item. The players simultaneously submit bids to the auctioneer, and the auctioneer assigns the item to the player with the highest bid. However, the winning player does not pay their bid, but only the second highest bid submitted to the auctioneer. In the Vickrey auction, the private information of each player is the maximum amount of money the player is willing to pay for the item on sale.

Intuitively, the Vickrey auction is truthful because the auction structure guarantees that each player has no regret from revealing their private valuation of the item. If a player bids more than their private valuation, the player runs the risk of winning and paying more than the valuation. On the other hand, if the player bids less than the private valuation, they do not gain any utility since there is no decrease in the payment required upon winning. Upon winning, the player always pays the second highest bid. Thus, when a player bids less than their private valuation, the player just runs the risk of losing the auction at no gain in utility. So, it is in each player's interest to bid exactly their private evaluation of the item on sale.

There has been a great amount of work in theoretical computer science on truthful mechanism design, starting with the work by Nisan and Ronen [42]. Feigenbaum, Papadimitriou, and Shenker have used truthful mechanisms in the context of multicast transmissions [19]. Tardos and Archer have created a general framework for designing truthful mechanisms where the private information of each player is a single positive real number [6].

There is an intuitive problem with truthfulness which makes the method not applicable in certain situations. We wish to have a player's private information so that we can make optimal decisions when playing against that player. However, a rational player does not reveal private information if it is disadvantageous to do so. We give two specific examples where this intuitive problem is manifested. First, if

the players are involved in just two consecutive, identical executions of a truthful mechanism, in general, it is no longer the case that revealing all private information is the optimal strategy. If a player reveals private information in the first execution, then the player may suffer a loss of utility in the second execution. The brittleness of truthfulness under repetition holds even for the most common truthful mechanism, the Vickrey auctions introduced at the beginning of this section [60]. A second example of the non applicability of truthfulness is related to the difficulty of enforcing a desirable property known as budget balance. To maintain truthfulness, many mechanisms make excessively large utility payments to the players. The payments made by the mechanism are often larger than the total wealth present at the beginning of the game, implying that the mechanism is not budget balanced. The constraints imposed by budget balance often irreconcilably conflict with truthfulness and the mechanism's desired property. For example, in general, it is impossible to design truthful, budget balanced mechanisms that maximize the net utility to all players [49].

A second approach to dealing with the problem of private information is the method of types developed by Harsanyi [27]. Informally, the method of types removes the problem of private information by introducing a mathematically meaningful notion of "beliefs" into the game. In Harsanyi's model, every player has beliefs, represented by probability distributions, about the private information of the other players. With this model, a player can make optimal decisions by pretending to play a random game with complete information, where the game is drawn from a distribution determined by the player's beliefs about the private values of the other players. The key to the application of this method is the ability to deal with a wide range of games of complete information.

While mathematically elegant, Harsanyi's method of types does have several drawbacks. The first, most glaring drawback is that a player may be entirely unsure of their own beliefs. But, to properly analyze a game using the method of types, the belief distributions are required. More pessimistic, worst case approaches can be taken, but even those require at least a range for the unknown data [2].

The second drawback to the method of types is a more technical one. In the formal statement of the method of types, each player has an associated set of types, where each type specifies a utility function and a belief probability distribution over the other players' types. The belief probability distributions remove the problem of

private information when it comes to unknown utility functions. However, to apply the method of types, while each player may not know the types (utilities and beliefs) of the other players, all players must know each other's *possible* types.

A third drawback of the method of types stems from computational considerations. To compute an equilibrium for a game using the method of types, we compute an equilibrium strategy for each player *type* instead of simply for each player. For example, suppose that we have a game between two players. Say Player 1 has two types $\{t_0^1, t_1^1\}$ and Player 2 has two types $\{t_0^2, t_1^2\}$. Suppose that we are helping Player 1 compute an equilibrium strategy. Since we are working with Player 1, we know the type of Player 1. Let us say that the type of Player 1 is t_0^1. To compute an equilibrium strategy for t_0^1, since we are unsure of the type of Player 2, we must compute equilibrium strategies for each of t_0^2 and t_1^2. But, the equilibrium strategies of the types t_0^2 and t_1^2 depend on an equilibrium strategy of the unrealized type t_1^1. Thus, to compute an equilibrium strategy for t_0^1, we compute one equilibrium strategy for each type in the game instead of one equilibrium strategy for each player. A lack of information results in an explosion in the number of types and an increase in the computational burden required for finding equilibria.

1.4.3 Random Processes in Games

Harsanyi's method of types is one instance where we can think of the set of players as playing a random game. The concept of playing random games is powerful and has found wide applicability in game theory. Similarly to the method of types, other models using a mixture of game theory and random processes are typically more realistic and applicable than the standard game theoretic models. For example, it is common to model complex games as a Markov chain, where each node of the chain contains a game. The result of playing the game at a particular node, along with some randomness, determines a transition to another node in the Markov chain. Games evolving based on a random process are called stochastic games.

One common application of stochastic games comes in modeling evolution. A common evolution model involves a fixed game and a player pool. At each step of the random process, we randomly pair players in the player pool and each pair plays the game. If a player does poorly over several rounds of random pairings, the player "dies" and is removed from the pool. On the other hand, players who do well in the

random pairings get to "reproduce," creating clones of themselves in the pool. The idea behind the evolution model is that over time, successful players dominate the population in the pool.

A second common application of stochastic games comes in modeling markets. A market model usually involves a set of players playing a market game, as well as a set of bystanders each of whom is presented with the choice of entering the market game. In addition to the current strategies of all players, the payoff each player receives from playing the market game is also dependent on the player's past strategy, the number of players playing the game, as well as some randomness inherent in the game. A key notion in such market models is that of a "shock." Every so often, as determined by random events, the market game experiences a shock which may drastically change the payoffs given to the players.

A third example of the combination of random processes and games comes from Shapley's study of stochastic games [54]. Shapley shows that two player, zero sum stochastic games have a value. In other words, under every complementary best-response equilibrium, the expected sum of payment exchanges over the course of the stochastic game is the same. Shapley analyzes the expected payments in the stochastic game, similarly to our analysis of the expected cover time in the graph-based coupon collector process. Shapley continues by giving expressions for optimal strategies in the stochastic game, strategies achieving the game's value. A natural extension of his work would be an analysis of the expected deviation from the game's value if the players follow optimal strategies. Such an extension would be analogous to our high probability deviation results for the expected cover time.

1.4.4 The Online Matroid Problem

In Chapter 4, we examine a revenue management problem under a random process with incomplete information. Suppose we own a store with n items for sale. We know that m customers will enter the store throughout the day. Whenever a customer enters, the customer specifies a subset of the items and tells us that he is willing to pay a certain dollar amount for any one of those items. We must immediately make a decision on which item to sell to the customer if any. After the transaction, the customer leaves the store never to be seen again. A natural question to ask is, even though arbitrary customers may arrive, if we know that a uniformly random

permutation of customers arrives throughout the day, can we achieve revenue that is within a constant factor of optimal without a priori knowing the customers' desired subsets of items or dollar amounts?

This particular revenue management problem finds applications in online advertising auctions and scheduling. For example, the problem can be reinterpreted as selling keywords on a search engine with the customer's dollar amounts representing bids for those keywords. Another interpretation may be selling server time slots, where customers represent jobs asking to be scheduled in certain slots for certain dollar amounts.

In general, the problem we have described can be abstracted as the online weighted transversal matroid matching (OWTMM) problem. See Chapter 4 for the formal definitions. The OWTMM problem is a generalization of the well studied secretary problem, which arose as a folklore problem in the 1950's [20, 21]. In the secretary problem, there is only a single item in the store, and we must choose the customer to whom we sell the item. Usually, the secretary problem is a reinterpretation of having a single position for a secretary and interviewing a random permutation of candidates.

Motivated by the applications in auctions, mechanism design, and revenue management, Babaioff et al. introduce the online matroid problem, a generalization of the OWTMM problem et al. [8]. In the online matroid problem, given the total number of matroid elements, the goal is to find an independent set with weight within a constant of the max-weight independent set when a random permutation of the elements revealed to us, one element at a time [43]. We call an algorithm solving the online matroid problem a *competitive* algorithm. As well as studying several instances of the online matroid problem, Babaioff et al. demonstrate the connections between the online matroid problem and the design of truthful mechanisms for online auctions. The work of Babaioff et al. extends the work Kleinberg on such truthful mechanisms [32].

The instances of the online matroid problem for which competitive algorithms are known include the secretary problem [36], uniform matroids [32], bounded left-degree transversal matroids, graphic matroids, and truncated matroids [8]. For general matroids, the best known algorithm gives an independent set within a factor of $O(\log r)$ of optimal, where r is the rank of the matroid [8].

In Chapter 4, we present a competitive algorithm for the OWTMM problem.

Our work generalizes all of the previously known competitive algorithm results for the online matroid problem. For example, as has been mentioned, the secretary problem is a special case of OWTMM, where there is only one item in the store. The online matroid problem on uniform matroids is a special case of OWTMM where every customer bids on every item. Of course, the online matroid problem on bounded left-degree transversal matroids is a special case of OWTMM where each customer can only bid on a small, fixed number of items. The graphic matroid results of Babaoiff et al. depend on a reduction to bounded left-degree transversal matroids. The results on truncated matroids give a reduction from an algorithm for matroid M to an algorithm for the truncation of matroid M. Along with our results on transversal matroids, this reduction gives competitive algorithms on the truncations of transversal matroids.

All of the competitive algorithms in the literature are ultimately based on generalization of a sample-and-price algorithm for the secretary problem that works as follows. We sample the first half of the customers, not selling the item to anyone, but recording the customer bids. We then sell the item to the first non-sampled customer, if any, who bids higher than the maximum bid of the sampled customers. This algorithm sells the item to the customer with the highest bid with probability at least $1/4$, since, with probability at least $1/4$, the second highest bidding customer is sampled and the highest bidding customer is not.

It is difficult to extend this algorithm to the full transversal matroid setting. Indeed, that is one reason why Babaioff et al. consider bounded left-degree transversal matroids. We discuss some of the difficulties in Chapter 4. We discuss our novel sample-and-price technique for solving the OWTMM problem. One of the main difficulties in proving the effectiveness of the technique are the probabilistic dependencies that arise in the analysis. To address these concerns, the main technical section of our proof is based on a series of counting arguments that directly imply the required probabilistic results.

Chapter 2

Dynamic Membership

2.1 Introduction

One of the most commonly discussed stochastic processes in computer science is the so-called coupon collector process [40]. In that process, there are n distinct coupons and we proceed in rounds, collecting one uniformly random coupon (with replacement) in each round. Are $O(n)$ rounds sufficient to collect all of the coupons? Put differently, is picking coupons with replacement as efficient, to within a constant factor, as picking them without replacement? No, it is a well-known fact that with high probability the number of rounds required to collect all of the coupons is $\Theta(n \log n)$.

This shortcoming has motivated Adler et al. [1] and Alon [4] to study a similar graph-based covering process. The nodes of the graph nodes represent the coupons and covering a node represents collecting a coupon. In each round, a uniformly random node w is selected. If an uncovered neighbor of w exists, choose one such uncovered neighbor and cover it. We refer to this process as process CC.

Process CC can use a variety of different *covering methods* to decide which uncovered neighbor to cover. If our ultimate goal is to minimize cover time, certainly the most powerful covering method available is an offline method with knowledge of the entire sequence of node selections and with infinite computing power. We refer to this powerful cover time minimizing version of process CC as process MIN. To achieve our $O(n)$ goal, it is natural to consider $\log n$-regular graphs since the work of Alon implies process MIN has an expected cover time of $\Omega(n + \frac{n \log n}{d})$ rounds on

d-regular graphs [4].

2.1.1 Logarithmic-Degree Graphs

Another natural version of process CC — in which the covering method chooses a uniformly random uncovered neighbor, if any — was studied by Adler et al. [1] and by Alon [4]. We refer to this version of process CC as process UNI. Alon shows that for logarithmic-degree Ramanujan expander graphs, process UNI completes in $O(n)$ time, matching the lower bound for process MIN.

Adler et al. show that for the hypercube, which has a weak expansion property but is not an expander, process UNI takes $O(n)$ time, also matching the lower bound for process MIN [1]. They also show that for arbitrary logarithmic-degree graphs, process UNI completes in $O(n \log \log n)$ time. Furthermore, Adler et al. present an application of process UNI to load balancing in a hypercubic distributed hash table (DHT).

A process that is intuitively similar to process UNI is one where we initially assign a rank to each node using a uniformly random permutation of the nodes, and the covering method covers the minimum-rank uncovered neighbor, if any. We refer to this permutation-based version of process CC as process P-RANK. In this chapter, we show that process P-RANK completes in $O(n)$ time on arbitrary logarithmic-degree graphs.

In fact, we analyze a more general and local version of process CC in which each node initially chooses a uniformly random rank in a suitable range, and the covering method covers the minimum-rank uncovered neighbor of the selected node. (We assume that the nodes are numbered from 1 to n, and that ties in rank are broken in favor of the lower-numbered node.) We refer to this random rank version of process CC as process R-RANK. We show that the more general and local process R-RANK completes in $O(n)$ time on arbitrary logarithmic-degree graphs.

2.1.2 Results for General Graphs

Alon shows that process MIN on any d-regular graph has expected cover time at least $n - \frac{n}{d} + \frac{n}{d} \ln(\frac{n}{d})$ [4]. Alon also shows that process UNI completes in time $n + (1 + o(1))\frac{n \ln n}{d}$ for random nearly d-regular graphs. Alon further shows that on any (n, d, λ)-expander graph the expected cover time of process UNI is at most

$n + n(\frac{\lambda}{d})^2(\ln n + 1)$. In particular, this implies that on Ramanujan graphs process UNI completes in $(1 + o(1))n$ time, matching the lower bound for process MIN.

If our goal is to maximize cover time, certainly the most powerful covering method available is an offline adversary with knowledge of the entire sequence of node selections and with infinite computing power. We refer to this powerful cover time maximizing version of process CC as process MAX. Alon notes that the upper bounds for expanders hold even if after every round an adversary "is allowed to shift the uncovered nodes to any place he wishes, keeping their number." In particular, this shows that on Ramanujan graphs, the cover time for process MAX matches the cover time for process MIN, up to constant factors. In effect, the covering method does not matter for this class of graphs.

Another previously studied variant of process CC favors covering the selected node. In this variant, we check — immediately after selecting a uniformly random node — if the selected node is uncovered. If it is, we cover it and move to the next selection. Only otherwise do we consider the neighbors of the selected node. We refer to the selection-biased variants of processes process UNI, process P-RANK, and process R-RANK as process UNI$'$, process P-RANK$'$, and process R-RANK$'$, respecively.

Adler et al. show that for all d-regular graphs, processes UNI and UNI$'$ finish in $O(n+n(\log n)(\log d)/d)$ time[1]. They also show that for random d-regular graphs only $O(n + \frac{n \log n}{d})$ steps are needed. Furthermore, they exhibit an application of process UNI$'$ to load balancing in DHTs.

All of the results matching Alon's lower bound for process MIN presented prior to this work have used some expansion properties of the underlying graph. In contrast, our proof techniques do not require the underlying graph to have any particular structure. Thus, we show the following general result: for directed graphs, with self-loops but no parallel edges, where each node has in-degree at least δ_{in} and at most Δ_{in}, and out-degree at most Δ_{out}, both process R-RANK and process R-RANK$'$ cover all nodes in $O(n \max(\Delta_{\text{in}}\Delta_{\text{out}}/\delta_{\text{in}}^2, (\log n)/\delta_{\text{in}}))$ rounds with high probability. This result matches Alon's lower bound for $\delta_{\text{in}} = \Delta_{\text{in}} = \Delta_{\text{out}} = \Theta(d)$, and is thus optimal under these conditions.

Furthermore, Alon's results for Ramanujan graphs raise the question whether there is any separation between the cover times for process MAX and process MIN. In other words, are there any graphs for which the choice of covering method mat-

ters? We define a weakly adversarial process, process A-RANK, that is similar to process P-RANK. In process A-RANK, instead of picking a uniformly random permutation, an adversary is initally allowed to fix the permutation used to assign ranks to the nodes. We then proceed as in process P-RANK. In addition, we define the selection-biased variant of process A-RANK as process A-RANK$'$. We establish that there exists a logarithmic-degree graph on which process A-RANK and process A-RANK$'$ each take $\omega(n)$ rounds to complete. This implies that in general there is separation between the cover times of process MIN and process MAX. In other words, the covering method does matter.

2.1.3 Proof Outline

The proof of our theorem is inspired by the delay sequence argument used by Ranade for the analysis of a certain packet routing problem on the butterfly [47] (see also [34, Section 3.4.4]). In a delay sequence argument, we identify certain combinatorial structures that exist whenever the random process lasts for a long time. Then, we show that the probability any of these structures exist is small. This in turn implies an upper bound on the running time of the random process.

There are significant differences between our proof and that of Ranade. For example, in our problem, the connection between the running time and the length of a delay sequence is not clear-cut, while in the butterfly routing problem analyzed by Ranade, the length of the delay sequence is equal to the running time. But let us begin by giving the notion of a delay sequence in our problem.

Consider the node that was covered last, say w_1. Why wasn't w_1 covered earlier? It was not covered earlier because at the last opportunity to cover w_1 — that is, the last selection in w_1's neighborhood — we covered some other node, w_2, instead. In such a case we consider w_1 to be delayed by w_2. Similarly, w_2 may be delayed by some node w_3, et cetera, until finally we reach a node w_k that is not delayed, i.e., w_k is covered at the first opportunity. The sequence of nodes $w_1, \ldots, w_k$ corresponds to our notion of a delay sequence.

In analyzing process R-RANK, we find it useful to first analyze a much simpler process, process SELECT, in which we repeatedly select a uniformly random node, never covering anything. After establishing several lemmas for the simpler process, we proceed to analyzing process R-RANK. This is the bulk of the proof,

and includes a technical lemma to work around the difficulties in linking cover time to delay sequence length. Finally, we reduce process R-RANK$'$ to process R-RANK to show that the same bounds hold.

The rest of this chapter is structured as follows. In Section 2.2, we present a number of useful definitions and lemmas related to standard probability distributions. In Section 2.3, we analyze the simple process, process SELECT. In Section 2.4, we analyze process R-RANK. In Section 2.5, we analyze process R-RANK$'$. In Section 2.6, we show the existence of a $\log(n)$-regular graph on which process A-RANK and process A-RANK$'$ each take $\omega(n)$ rounds to complete, establishing that the neighbor selection method does matter. We conclude the paper with Section 2.7, where we provide different yet equivalent views of the processes discussed here as well as discuss several remaining open problems.

2.2 Preliminaries

We use the term ℓ-sequence to refer to a sequence of length ℓ. For any ℓ-sequence σ of elements of a given type, and any element x of the same type, we let $\sigma : x$ denote the $(\ell + 1)$-sequence obtained by appending element x to σ.

For any nonnegative integer n and probability p, we use the notation $X \sim \mathrm{Bin}\,(n, p)$ to denote that the random variable X has a binomial distribution with n trials and success probability p. Similarly, we write $X \sim \mathrm{Geo}\,(p)$ to indicate that the random variable X has a geometric distribution with success probability p, and we write $X \sim \mathrm{NegBin}\,(r, p)$ to indicate that the random variable X has a negative binomial distribution with r successes and success probability p.

Lemma 2.2.1. *Let p denote an arbitrary probability, let ℓ denote an arbitrary nonnegative integer, and let $X \sim \mathrm{NegBin}\,(\ell, p)$. For any integer j such that $1 \leq j \leq \ell$, let p_j denote an arbitrary probability such that $p_j \geq p$, let $Y_j \sim \mathrm{Geo}\,(p_j)$, and let $Y = \sum_{1 \leq j \leq \ell} Y_j$. Then for any nonnegative integer i, $\Pr(X \geq i) \geq \Pr(Y \geq i)$.*

Proof. Note that if $p_j = p$ for all j, then the random variables X and Y have the same distribution. Furthermore, increasing any of the p_j's can only decrease Y. □

Lemma 2.2.2. *For any nonnegative integers r and n, and any probability p, we have $\Pr(X < r) = \Pr(Y > n)$, where $X \sim \mathrm{Bin}\,(n, p)$ and $Y \sim \mathrm{NegBin}\,(r, p)$.*

Proof. The random variables X and Y can be seen as different views of the same experiment where we successively flip coins with probability of success p. With Y, we ask "How many flips are required for r successes?" With X, we ask "How many successes are in the first n flips?" In this experiment, the event of seeing less than r successes in the first n flips ($X < r$) corresponds to the event that we have to wait more than n flips for the first r successes ($Y > n$). This gives the result. □

Lemma 2.2.3. *For any integer $r \geq 2$, we have $\Pr(X \geq 2E[X]) = \Pr(X \geq 2r/p) \leq \exp(-r/8)$, where $X \sim \text{NegBin}(r, p)$.*

Proof. Let $j = \left\lfloor \frac{2r}{p} \right\rfloor - 1$ and let $Y \sim \text{Bin}(j, p)$. By Lemma 2.2.2, we know that $\Pr(X \geq \frac{2r}{p}) \leq \Pr(X \geq \left\lfloor \frac{2r}{p} \right\rfloor) = \Pr(X > \left\lfloor \frac{2r}{p} \right\rfloor - 1) = \Pr(Y < r) = \Pr(Y \leq r - 1)$.

$$\begin{aligned} \Pr\left(Y \leq \frac{jp}{2}\right) &= \Pr\left(Y \leq r - (\eta+1)\frac{p}{2}\right) \\ &= \Pr(Y \leq r - 1) \end{aligned}$$

where $\frac{2r}{p} = \left\lfloor \frac{2r}{p} \right\rfloor + \eta$ and the last equality holds because $0 < (\eta+1)\frac{p}{2} < 1$.

Recall the Chernoff bounds in the form $\Pr(Y \leq (1-\lambda)jp) \leq \exp(-\lambda^2 jp/2)$ for $0 < \lambda < 1$ (see [5, 28]).

We apply this bound with $\lambda = \frac{1}{2}$ to get

$$\begin{aligned} \Pr(Y \leq r - 1) &= \Pr(Y \leq jp/2) \\ &\leq \exp(-jp/8) \\ &\leq \exp\left(\frac{-2r + (\eta+1)p}{8}\right) \\ &\leq \exp(-r/8) \end{aligned}$$

where η is as previously defined and the last inequality holds because $r \geq 2$. □

Lemma 2.2.4. *Let p be an arbitrary probability and let X be the sum of n independent Bernoulli variables $X_1, \ldots, X_n$, where X_j has success probability $p_j \geq p$. Then $\Pr(X \leq np/2) \leq \exp(-np/12)$.*

Proof. The result follows from Chernoff bounds (see, e.g., [5, 28]). □

Lemma 2.2.5. *Suppose we repeatedly throw balls independently and uniformly at random into n bins, and let the random variable X denote the number of throws required for every bin to receive at least n balls. Then X is $O(n^2)$ with high probability, that is, with failure probability that is an arbitrary inverse polynomial in n.*

Proof. The result follows from Lemma 2.2.4. □

Lemma 2.2.6. *Let j balls be thrown independently and uniformly at random into n bins. Let X denote the number of bins with at least one ball at the end of the experiment. Then,* $\Pr(X \leq \min(n/4, j/4)) \leq \exp(-j/2)$.

Proof. Let $[n] = \{1, 2, \ldots, n\}$. Suppose $\min\left(\frac{n}{4}, \frac{j}{4}\right) = k$. Let $S \subseteq [n]$ be a particular subset of size k. Then,

$$\Pr(\text{all balls land in } S) \leq \left(\frac{k}{n}\right)^j$$

Thus,

$$\begin{aligned}
\Pr(X \leq k) &= \Pr\left(\bigcup_{S \text{ s.t. } |S|=k} \text{all balls land in } S\right) \\
&\leq \binom{n}{k}\left(\frac{k}{n}\right)^j \\
&\leq \left(\frac{en}{k}\right)^k \left(\frac{k}{n}\right)^j \\
&= \left(\frac{en}{k}\right)^k \left(\frac{k}{n}\right)^{\frac{j}{2}} \left(\frac{k}{n}\right)^{\frac{j}{2}}
\end{aligned}$$

Now, since $\frac{j}{2} \geq 2k$ and since $k \leq \frac{n}{4}$ implies $\frac{ek}{n} \leq \frac{e}{4} < 1$

$$\begin{aligned}
\Pr(X \leq k) &\leq \left(\frac{ek}{n}\right)^k \left(\frac{k}{n}\right)^{\frac{j}{2}} \\
&\leq \left(\frac{ek}{n}\right)^k \left(\frac{1}{4}\right)^{\frac{j}{2}} \\
&\leq \exp\left(-\frac{j}{2}\right)
\end{aligned}$$

□

2.3 Process SELECT

Throughout the remainder of the chapter, we fix an arbitrary directed graph $G = (V, E)$ where $|V| = n > 0$. We say that an event holds "with high probability" if the probability that it fails to occur is upper bounded by an arbitrary inverse polynomial in n. We let δ_{in}, Δ_{in}, and Δ_{out} denote the minimum in-degree, maximum in-degree, and maximum out-degree of any node, respectively. For ease of exposition, we assume throughout the chapter that $\delta_{\text{in}} > 0$. The edge set E is allowed to contain loops but not parallel edges. For any node v, we define $\Gamma_{\text{in}}(v)$ as $\{w \mid (w, v) \in E\}$. For any sequence of edges $\sigma = (u_1, v_1), \ldots, (u_\ell, v_\ell)$, we define the two sequences of nodes $\text{src}(\sigma) = u_1, \ldots, u_\ell$ and $\text{dst}(\sigma) = v_1, \ldots, v_\ell$.

In this section, we analyze a simple stochastic process, process SELECT, defined as follows. Initially, we fix a positive integer r and independently assign each node in V a uniformly random integer rank between 1 and r. Process SELECT then proceeds in an infinite number of rounds, indexed from 1. In each round, one node is selected uniformly at random, with replacement. The following definitions are central to our analysis of this process.

A node sequence is said to be *rank-sorted* if the associated sequence of node ranks is nondecreasing.

For any node sequence σ, we inductively define $\text{duration}(\sigma)$, a nonnegative integer, and a node sequence $\text{select}(\sigma)$ as follows. If σ is empty, then $\text{duration}(\sigma)$ is 0 and $\text{select}(\sigma)$ is empty. Otherwise, σ is of the form $\tau : v$ for some shorter node sequence τ and node v. Let i denote the the least i such that $i > \text{duration}(\tau)$ and the node selected in round i belongs to $\Gamma_{\text{in}}(v)$. Let u denote the node selected in round i. Then we define $\text{duration}(\sigma)$ as i, and $\text{select}(\sigma)$ as $\text{select}(\tau) : u$.

Lemma 2.3.1. *For any ℓ-sequence of distinct nodes σ, $\Pr(\sigma$ is rank-sorted$) = \binom{\ell+r-1}{\ell} r^{-\ell}$.*

Proof. There are $\binom{\ell+r-1}{\ell}$ ways that ranks can be assigned to the ℓ distinct nodes so that the resulting ℓ-sequence is rank-sorted. The result follows since each such assignment occurs with probability $r^{-\ell}$. □

Lemma 2.3.2. *For any ℓ-sequence of nodes $\sigma = v_1, \ldots, v_\ell$ and any nonnegative integer i, we have*
$\Pr(\text{duration}(\sigma) = i) \leq \Pr(X \geq i)$, *where* $X \sim \text{NegBin}\left(\ell, \frac{\delta_{\text{in}}}{n}\right)$.

Proof. We proceed by proving that

$$\Pr(\text{duration}(\sigma) = i) \quad = \quad \Pr\left(\sum_{k=1}^{\ell} Y_k = i\right)$$

where $Y_k \sim \text{Geo}\left(\frac{d_k}{n}\right)$ and d_k denotes the in-degree of v_k. The desired bound then follows by Lemma 2.2.1.

We prove the foregoing claim by induction on ℓ. If $\ell = 0$, the claim holds since $\text{duration}(\sigma) = \sum_{k=1}^{\ell} Y_k = 0$.

For $\ell > 0$, we let τ denote the node sequence $v_1, \ldots, v_{\ell-1}$ and assume inductively that

$$\Pr(\text{duration}(\tau) = i) \quad = \quad \Pr\left(\sum_{k=1}^{\ell-1} Y_k = i\right).$$

Thus,

$$\begin{aligned}
&\Pr(\text{duration}(\sigma) = i) \\
&\quad = \sum_{j=0}^{i-1} \Pr(\text{duration}(\tau) = j) \cdot \\
&\qquad\qquad \Pr(\text{duration}(\sigma) - \text{duration}(\tau) = i - j \mid \text{duration}(\tau) = j) \\
&\quad = \sum_{j=0}^{i-1} \Pr(\text{duration}(\tau) = j) \cdot \Pr(\text{duration}(\sigma) - \text{duration}(\tau) = i - j) \\
&\quad = \sum_{j=0}^{i-1} \Pr(\text{duration}(\tau) = j) \cdot \Pr(Y_\ell = i - j) \\
&\quad = \sum_{j=0}^{i-1} \Pr\left(\sum_{k=1}^{\ell-1} Y_k = j\right) \cdot \Pr(Y_\ell = i - j) \\
&\quad = \Pr\left(\sum_{k=1}^{\ell-1} Y_k = i\right).
\end{aligned}$$

The second equality holds because each selection is independent of previous selections. The third equality holds because the waiting time to obtain a selection in $\Gamma_{\text{in}}(v_\ell)$ is distributed as Y_ℓ. □

Lemma 2.3.3. *For any ℓ-sequence of edges σ,* $\Pr(\text{select}(\text{dst}(\sigma)) = \text{src}(\sigma)) \leq \delta_{\text{in}}^{-\ell}$.

Proof. We proceed by induction on ℓ. For $\ell = 0$, $\Pr(\text{select}(\text{dst}(\sigma)) = \text{src}(\sigma)) = 1 = \delta_{\text{in}}^{0}$ since we have assumed that $\delta_{\text{in}} > 0$.

For $\ell > 0$, σ can be written in the form $\tau : (u, v)$, where we inductively assume that the claim of the lemma holds for τ. Let A denote the event that the first node selected in $\Gamma_{\text{in}}(v)$ after round $\text{duration}(\text{dst}(\tau))$ is u. We have

$$\begin{aligned}
&\Pr(\text{select}(\text{dst}(\sigma)) = \text{src}(\sigma)) \\
&\quad = \Pr(\text{select}(\text{dst}(\tau)) = \text{src}(\tau)) \cdot \Pr(A \mid \text{select}(\text{dst}(\tau)) = \text{src}(\tau)) \\
&\quad = \Pr(\text{select}(\text{dst}(\tau)) = \text{src}(\tau)) \cdot \Pr(A) \\
&\quad \leq \delta_{\text{in}}^{-\ell}.
\end{aligned}$$

The second step follows from the independence of the events A and $\text{select}(\text{dst}(\tau)) = \text{src}(\tau)$. (These two events are independent since each selection is independent of previous selections.) The third step follows from the induction hypothesis and the observation that $\Pr(A)$ is equal $1/\Gamma_{\text{in}}(v)$, which is at most $1/\delta_{\text{in}}$. □

Lemma 2.3.4. *For any ℓ-sequence of edges σ and nonnegative integer i, the three events specified as $A =$ "$\text{dst}(\sigma)$ is rank-sorted", $B =$ "$\text{duration}(\text{dst}(\sigma)) = i$", and $C =$ "$\text{select}(\text{dst}(\sigma)) = \text{src}(\sigma)$" are mutually independent.*

Proof. Note that event A depends only on the rank assignments, while events B and C depend only on the selections. Thus event A is independent of events B and C. Below we argue that events B and C are independent.

Let $\sigma = (u_1, v_1), \ldots, (u_\ell, v_\ell)$ and let σ_j denote the length-j prefix of σ, $0 \leq j \leq \ell$. Define a selection to be *j-special*, $1 \leq j \leq \ell$, if it is the first selection after round $\text{duration}(\sigma_{j-1})$ in $\Gamma_{\text{in}}(v_j)$. A selection is *special* if it is j-special for some j. Note that event B depends only on the timing of the special events; in particular, B occurs if and only if the ℓ-special selection occurs in round i. Suppose we run process SELECT, but at each step, instead of revealing the selected node, we reveal only whether the selection is special. This information is sufficient to determine the unique i for which B occurs, but does not bias the distribution of $\text{select}(\text{dst}(\sigma))$. Since event C only depends on $\text{select}(\text{dst}(\sigma))$, it is independent of B. □

Lemma 2.3.5. *Let σ be an ℓ-sequence of edges so that the nodes of* $\mathrm{dst}(\sigma)$ *are distinct, let* $X \sim \mathrm{NegBin}\left(\ell, \frac{\delta_{\mathrm{in}}}{n}\right)$*, let i be a nonnegative integer, and let events A, B, and C be defined as in the statement of Lemma 2.3.4. Then* $\Pr(A \cap B \cap C) \leq \binom{\ell+r-1}{\ell} \cdot \Pr(X \geq i) \cdot (r\delta_{\mathrm{in}})^{-\ell}$.

Proof. By Lemma 2.3.1, $\Pr(A) \leq \binom{\ell+r-1}{\ell} r^{-\ell}$. By Lemma 2.3.2, $\Pr(B) \leq \Pr(X \geq i)$. By Lemma 2.3.3, $\Pr(C) \leq \delta_{\mathrm{in}}^{-\ell}$. The claim then follows by Lemma 2.3.4. □

2.4 Process R-RANK

In the section we analyze an augmented version of process SELECT, referred to as Process R-RANK, in which we maintain a notion of a "covered subset" of the nodes. Initially, all of the nodes are uncovered. Process R-RANK then proceeds in rounds in exactly the same manner as process SELECT, except that in any given round, if one or more outgoing neighbors of the selected node are uncovered, we cover the uncovered outgoing neighbor with minimum rank. (As indicated in Section 2.1, ties are broken according to some arbitrary numbering of the nodes.)

Note that process R-RANK simply augments process SELECT by also covering nodes; rank assignment and selections are performed in exactly the same manner in the two processes. Thus all of the definitions and lemmas presented in Section 2.3 are applicable to process R-RANK. The following additional definitions are useful for our analysis of process R-RANK.

The *cover time* of process R-RANK is defined as the number of rounds required to cover all of the nodes.

We inductively define the notion of a *linked* sequence of edges. For ℓ equal to 0 or 1, any ℓ-sequence of edges is linked. For $\ell > 1$, an ℓ-sequence of edges of the form $\sigma : (u, v) : (u', v')$ is linked if the $(\ell - 1)$-sequence $\sigma : (u, v)$ is linked and (u, v') belongs to E.

For any node v, we define $\mathrm{parent}(v)$ as follows. Let i denote the round in which node v is covered. If i is the first round in which some node in $\Gamma_{\mathrm{in}}(v)$ is selected, then $\mathrm{parent}(v)$ is defined to be nil. Otherwise, $\mathrm{parent}(v)$ is the node covered in the first round prior to round i in which the selected node belongs to $\Gamma_{\mathrm{in}}(v)$.

We inductively define the notion of a *chronological* sequence of nodes as follows. Any ℓ-sequence of nodes with $\ell \leq 1$ is chronological. An ℓ-sequence of

nodes of the form $\sigma : v : v'$ is chronological if $\sigma : v$ is chronological and node v is covered before node v'.

We inductively define the notion of an *active* node sequence as follows. The empty node sequence is active. A singleton node sequence consisting of the node v is active if $\text{parent}(v) = \text{nil}$. An ℓ-sequence of nodes of the form $\sigma : v : v'$ is active if $\sigma : v$ is active and $\text{parent}(v') = v$.

We call an ℓ-sequence of edges σ *active* if $\text{dst}(\sigma)$ is active and $\text{select}(\text{dst}(\sigma)) = \text{src}(\sigma)$.

We call an ℓ-sequence of edges σ *i-active* if it is active and either $\ell = i = 0$ or $\ell > 0$, σ is of the form $\sigma : (u, v)$, and v is the node covered in round i.

Lemma 2.4.1. *For any nonnegative integer ℓ, there are at most $n\Delta_{\text{out}}^{\ell}\Delta_{\text{in}}^{\ell-1}$ linked ℓ-sequences of edges.*

Proof. We proceed by induction on ℓ, treating $\ell = 0$ and $\ell = 1$ as the base cases. For $\ell = 0$, the empty sequence is the only linked 0-sequence, and the claim holds since $n/\Delta_{\text{in}} \geq 1$. (Note that Δ_{in} is at most n since we do not allow parallel edges.) For $\ell = 1$, the number of linked 1-sequences is at most $|E| \leq n\Delta_{\text{out}}$.

Now let ℓ be greater than 1 and inductively assume that the number of linked $(\ell-1)$-sequences of edges is at most $n\Delta_{\text{out}}^{\ell-1}\Delta_{\text{in}}^{\ell-2}$. Recall that any linked ℓ-sequence of edges is of the form $\sigma : (u, v) : (u', v')$ where the $(\ell-1)$-sequence of edges $\sigma : (u, v)$ is linked and (u, v') belongs to E. Observe that for any linked $(\ell-1)$-sequence of edges $\sigma : (u, v)$, there are at most Δ_{out} nodes v' such that (u, v') belongs to E, and for each such choice of v', there are at most Δ_{in} nodes u' such that (u', v') belongs to E. Thus the number of linked ℓ-sequences is at most $\Delta_{\text{out}}\Delta_{\text{in}}$ times the number of linked $(\ell-1)$-sequences, and the desired bound follows from the induction hypothesis. □

Lemma 2.4.2. *Suppose we run two instances of process R-RANK in parallel using the same random ranks and the same sequence of random selections, but in the second instance, we allow an arbitrary subset of the covered nodes to be uncovered after each round. Then the cover time of the first instance is at most the cover time of the second instance.*

Proof. By a straightforward induction on the number of rounds, at all times, the set of covered nodes in the first instance contains the set of covered nodes in the second instance. The claim of the lemma follows. □

Lemma 2.4.3. *For any rank assignment, the expected cover time of process R-RANK is $O(n^2)$.*

Proof. It follows from Lemma 2.2.5 that the cover time is $O(n^2)$ with high probability since in that time each vertex is selected at least n times, implying that all of its neighbors are covered.

We can then consider a modified version of process R-RANK in which the infinite sequence of rounds is partitioned into epochs of $O(n^2)$ rounds, and where at the end of each epoch, if the nodes are not all covered, all nodes are uncovered before proceeding to the next epoch. Since each epoch covers all the nodes with high probability, the expected cover time of this modified version of process R-RANK is $O(n^2)$. By Lemma 2.4.2, for any rank assignment, the expected cover time of process R-RANK is $O(n^2)$. □

Lemma 2.4.4. *Assume that v is the node covered in round i and let u be the node selected in round i. Then there is an i-active edge sequence σ terminating in edge (u, v) and such that* $\text{duration}(\text{dst}(\sigma)) = i$.

Proof. Observe that u belongs to $\Gamma_{\text{in}}(v)$. Furthermore, if $\text{parent}(v) = \text{nil}$, then the singleton node sequence v is active with $\text{duration}(v) = i$. Thus the singleton edge sequence $\sigma = (u, v)$ is i-active with $\text{duration}(\text{dst}(\sigma)) = i$.

We prove the claim by induction on i. For $i = 1$, we have $\text{parent}(v) = \text{nil}$ and so the claim follows by the observations of the previous paragraph.

For $i > 1$, if $\text{parent}(v) = \text{nil}$, the claim once again follows from the foregoing observations. Otherwise, $\text{parent}(v) = v'$ where v' is the node covered in round j with $j < i$. Let u' denote the node selected in round j. Since $j < i$, we can inductively assume that there is a j-active edge sequence, call it τ, terminating in edge (u', v') and such that $\text{duration}(\text{dst}(\tau)) = j$. Since τ is active, the node sequence $\text{dst}(\tau)$ is active and $\text{select}(\text{dst}(\tau)) = \text{src}(\tau)$. Let $\sigma = \tau : (u, v)$. Thus $\text{src}(\sigma) = \text{src}(\tau) : u$ and $\text{dst}(\sigma) = \text{dst}(\tau) : v$. Since $\text{parent}(v) = v'$ and $\text{dst}(\tau)$ is an active node sequence terminating in node v', $\text{dst}(\sigma)$ is active. Since $\text{duration}(\text{dst}(\tau)) = j$, $\text{select}(\text{dst}(\tau)) = \text{src}(\tau)$, u was selected in round i, and i is the least integer greater that j such that the node selected in round i belongs to $\Gamma_{\text{in}}(v)$, we have $\text{duration}(\text{dst}(\sigma)) = i$ and $\text{select}(\text{dst}(\sigma)) = \text{src}(\sigma)$. Since $\text{dst}(\sigma)$ is active and $\text{select}(\text{dst}(\sigma)) = \text{src}(\sigma)$, σ is active. Since σ is active and v is the node covered in round i, σ is i-active. Thus the edge sequence σ satisfies all of the requirements of the lemma. □

Lemma 2.4.5. *Any active node sequence is rank-sorted, chronological, and consists of distinct nodes.*

Proof. Note that any chronological node sequence consists of distinct nodes. Thus, in what follows, it is sufficient to prove that any active node sequence is rank-sorted and chronological.

We proceed by induction on the length of the sequence. For the base case, note that any node sequence of length 0 or 1 is rank-sorted and chronological. For the induction step, consider an active node sequence σ of the form $\tau : v : v'$. Since σ is active, $\tau : v$ is active and $\text{parent}(v') = v$. Since $\tau : v$ is active, the induction hypothesis implies that it is also rank-sorted and chronological. Since $\text{parent}(v') = v$, $\text{rank}(v) \leq \text{rank}(v')$ and v is covered before v'. Hence σ is rank-sorted and chronological. □

Lemma 2.4.6. *For any nonempty active edge sequence σ, if the last edge in σ is (u, v), then v is the node covered in round* $\text{duration}(\text{dst}(\sigma))$ *and node u is selected in the same round.*

Proof. We prove the claim by induction on the length of the active edge sequence σ.

If σ consists of a single edge (u, v), then by the definition of an active edge sequence, the singleton node sequence $\text{dst}(\sigma)$ is active and $\text{select}(\text{dst}(\sigma)) = \text{src}(\sigma)$. Since $\text{dst}(\sigma)$ is active, $\text{parent}(v) = \text{nil}$, that is, v is the node covered in the first round in which a node in $\Gamma_{\text{in}}(v)$ is selected, which is round $\text{duration}(\text{dst}(\sigma))$. Since $\text{select}(\text{dst}(\sigma)) = \text{src}(\sigma)$, node u is selected in the same round.

Now assume that σ is an active edge sequence of the form $\tau : (u, v)$, where τ is of the form $\tau' : (u', v')$. Since σ is active, the node sequence $\text{dst}(\sigma)$ is active and $\text{select}(\text{dst}(\sigma)) = \text{src}(\sigma)$. It follows that $\text{dst}(\tau)$ is active and $\text{select}(\text{dst}(\tau)) = \text{src}(\tau)$, that is, τ is also active. Since τ is active and shorter than σ, we can inductively assume that v' is the node covered in round $\text{duration}(\text{dst}(\tau))$ and node u' is selected in the same round. Since $\text{dst}(\sigma)$ is active, $\text{parent}(v) = v'$, that is, v is the node covered in the first round after round $\text{duration}(\text{dst}(\tau))$ in which a node in $\Gamma_{\text{in}}(v)$ is selected. Applying the definition of $\text{duration}(\text{dst}(\sigma))$, we conclude that v is the node covered in round $\text{duration}(\text{dst}(\sigma))$. Since $\text{select}(\text{dst}(\sigma)) = \text{src}(\sigma)$, node u is selected in the same round. □

Lemma 2.4.7. *If σ is an active sequence of edges, then σ is linked.*

Proof. We proceed by induction on the length of σ. If the length of σ is 0 or 1, then σ is linked by definition.

Now assume that σ is an edge sequence of the form $\tau : (u, v)$, where τ is of the form $\tau' : (u', v')$ and σ is active. Since σ is active, $\mathrm{dst}(\sigma)$ is active. Since $\mathrm{dst}(\sigma)$ is active, $\mathrm{dst}(\tau)$ is also active. Since $\mathrm{dst}(\tau)$ is active and τ is shorter than σ, we can inductively assume that τ is linked. Therefore, in order to establish that σ is linked, it is sufficient to prove that (u', v) is an edge. Since $\mathrm{dst}(\sigma)$ is active, $\mathrm{parent}(v) = v'$. Hence, letting i denote the round in which node v is covered, we find that v' is the node covered in the first round prior to round i in which the selected node belongs to $\Gamma_{\mathrm{in}}(v)$. By Lemma 2.4.6, v' is covered in a round in which node u' is selected. Thus u' belongs to $\Gamma_{\mathrm{in}}(v)$, that is, (u', v) is an edge, as required. □

Lemma 2.4.8. *If an edge sequence σ is i-active, then* $\mathrm{duration}(\mathrm{dst}(\sigma)) = i$.

Proof. If σ is empty, then the claim holds since $i = 0$ and $\mathrm{duration}(\mathrm{dst}(\sigma)) = 0$. Otherwise, σ is of the form $\tau : (u, v)$, and by the definition of an i-active edge sequence, v is the node covered in round i. By Lemma 2.4.6, v is the node covered in round $\mathrm{duration}(\mathrm{dst}(\sigma))$, so $\mathrm{duration}(\mathrm{dst}(\sigma)) = i$. □

Lemma 2.4.9. *For any ℓ-sequence of edges σ, and any nonnegative integer i, the probability that σ is i-active is at most $\binom{\ell+r-1}{\ell} \cdot \Pr(X \geq i) \cdot (r\delta_{\mathrm{in}})^{-\ell}$, where $X \sim \mathrm{NegBin}\left(\ell, \frac{\delta_{\mathrm{in}}}{n}\right)$.*

Proof. If the nodes in $\mathrm{dst}(\sigma)$ are not all distinct, then $\Pr(\sigma \text{ is } i\text{-active}) = 0$ by Lemma 2.4.5 and the claimed inequality holds since the right-hand side is nonnegative.

Now assume that $\mathrm{dst}(\sigma)$ consists of distinct nodes, and let events A, B, and C be as defined in the statement of Lemma 2.3.4. Below we prove that if σ is i-active, then events A, B, and C all occur. The claimed inequality then follows by Lemma 2.3.5.

Assume that σ is i-active. Thus event B occurs by Lemma 2.4.8. Furthermore, σ is active, so $\mathrm{dst}(\sigma)$ is active and event C occurs by the definition of an active edge sequence. Since $\mathrm{dst}(\sigma)$ is active, event A occurs by Lemma 2.4.5. □

Lemma 2.4.10. *For any nonnegative integers i and ℓ, the probability that some ℓ-sequence of edges is i-active is at most*

$$n\Delta_{\text{out}}^{\ell}\Delta_{\text{in}}^{\ell-1}\binom{\ell+r-1}{\ell}\frac{\Pr(X \geq i)}{(r\delta_{\text{in}})^{\ell}}$$

where $X \sim \text{NegBin}\left(\ell, \frac{\delta_{\text{in}}}{n}\right)$.

Proof. By Lemma 2.4.7, if an edge sequence σ is not linked, then $\Pr(\sigma \text{ is } i\text{-active}) = 0$. A union bound then implies that the probability some ℓ-sequence of edges is i-active is at most the number of linked ℓ-sequences of edges multiplied by the maximum probability that any particular ℓ-sequence is i-active. The desired inequality then follows by Lemmas 2.4.1 and 2.4.9. □

Lemma 2.4.11. *For nonnegative integers i, ℓ, and r satisfying the properties $i \geq 64n\max(\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}^2, (\ln n)/\delta_{\text{in}})$ and $r \geq \min(\lceil 2e^2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}\rceil, \ell)$, we have*

$$\Delta_{\text{out}}^{\ell}\Delta_{\text{in}}^{\ell-1}\binom{\ell+r-1}{\ell}\frac{\Pr(X \geq i)}{(r\delta_{\text{in}})^{\ell}} \quad \leq \quad \exp(-i\delta_{\text{in}}/(32n))$$

where $X \sim \text{NegBin}\left(\ell, \frac{\delta_{\text{in}}}{n}\right)$.

Proof. First, we show that the LHS of the claimed inequality is a nonincreasing function of r.

It is sufficient to prove that the expression $\binom{\ell+r-1}{\ell}r^{-\ell}$ is a nonincreasing function of r. Fix ℓ and let $f(r)$ denote the preceding expression. Note that

$$\begin{aligned}\frac{f(r+1)}{f(r)} &= \frac{r+\ell}{r}\left(\frac{r}{r+1}\right)^{\ell} \\ &= \left(1+\frac{\ell}{r}\right)\left(1+\frac{1}{r}\right)^{-\ell} \\ &\leq 1,\end{aligned}$$

where the last inequality holds since the binomial theorem implies $(1+\frac{1}{r})^{\ell} \geq 1+\frac{\ell}{r}$.

Since we have established that the LHS of the claimed inequality is a nonincreasing function of r, we can assume that $r = \min(\lceil 2e^2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}\rceil, \ell)$.

Let us rewrite the LHS of the claimed inequality as $\lambda \cdot \Pr(X \geq i)$, where

$$\begin{aligned}
\lambda &= \Delta_{\text{out}}^{\ell} \Delta_{\text{in}}^{\ell-1} \binom{\ell + r - 1}{\ell} (r\delta_{\text{in}})^{-\ell} \\
&\leq \Delta_{\text{out}}^{\ell} \Delta_{\text{in}}^{\ell} \left(\frac{e(\ell + r - 1)}{\ell r \delta_{\text{in}}} \right)^{\ell} \\
&\leq \left(\frac{e \Delta_{\text{out}} \Delta_{\text{in}} (\ell + r)}{\ell r \delta_{\text{in}}} \right)^{\ell}. \qquad (2.1)
\end{aligned}$$

We begin by establishing two useful upper bounds on λ, namely, Equations (2.2) and (2.4) below.

If $r = \lceil 2e^2 \Delta_{\text{out}} \Delta_{\text{in}} / \delta_{\text{in}} \rceil$, then since since $r = \min(\lceil 2e^2 \Delta_{\text{out}} \Delta_{\text{in}} / \delta_{\text{in}} \rceil, \ell)$, we have $r \leq \ell$. Substituting the value of r into Equation (2.1), we find that

$$\begin{aligned}
\lambda &\leq \left(\frac{e(\ell + r)}{2e^2 \ell} \right)^{\ell} \\
&\leq \left(\frac{2e\ell}{2e^2 \ell} \right)^{\ell} \\
&\leq e^{-\ell}. \qquad (2.2)
\end{aligned}$$

If $r = \ell$, then Equation (2.1) implies

$$\lambda \leq \left(\frac{2e \Delta_{\text{out}} \Delta_{\text{in}}}{\ell \delta_{\text{in}}} \right)^{\ell}. \qquad (2.3)$$

Let $h(\ell)$ denote the natural logarithm of the RHS of Equation (2.3), that is, $h(\ell) = \ell \ln(2e\Delta_{\text{out}}\Delta_{\text{in}}/(\ell\delta_{\text{in}}))$. Using elementary calculus, it is straightforward to prove that the derivative of $h(\ell)$ with respect to ℓ is positive for $\ell < 2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}$, is 0 when $\ell = 2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}$, and is negative for $\ell > 2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}$. It follows that $h(\ell) \leq h(2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}) = 2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}$. Since ln is monotonic, the RHS of Equation (2.3) is also maximized when $\ell = 2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}$. Combining this result with Equation (2.2), we find that for any r

$$\lambda \leq \exp(2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}). \qquad (2.4)$$

(Note that $\exp(2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}) \geq 1$ and Equation (2.2) implies $\lambda \leq 1$ when $r = \lceil 2e^2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}} \rceil$.)

We are now ready to proceed with the proof of the lemma. We consider the two cases $\ell > \lceil i\delta_{\text{in}}/(2n)\rceil$ and $\ell \leq \lceil i\delta_{\text{in}}/(2n)\rceil$ separately.

If $\ell > \lceil i\delta_{\text{in}}/(2n)\rceil$, then $\ell > 2ec\max(\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}, \ln n)$ where $c = 16/e > e$. Thus $\ell > \lceil 2e^2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}\rceil$ and so $r = \lceil 2e^2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}\rceil$. It follows from Equation (2.2) that $\lambda \leq e^{-\ell} \leq \exp(-i\delta_{\text{in}}/(2n)) \leq \exp(-i\delta_{\text{in}}/(64n))$, and hence the claim holds since $\Pr(X \geq i) \leq 1$.

Now assume that $\ell \leq \lceil i\delta_{\text{in}}/(2n)\rceil$. Let $Y \sim \text{NegBin}\left(\left\lfloor \frac{i\delta_{\text{in}}}{2n}\right\rfloor, \frac{\delta_{\text{in}}}{n}\right)$ and $Z \sim \text{NegBin}\left(\left\lfloor \frac{i\delta_{\text{in}}}{2n}\right\rfloor - \ell, \frac{\delta_{\text{in}}}{n}\right)$. By the definition of the negative binomial distribution, $\Pr(Y \geq i) = \Pr(X + Z \geq i)$. And, since Z is nonnegative, $\Pr(X + Z \geq i) \geq \Pr(X \geq i)$. Thus

$$\Pr(X \geq i) \leq \Pr(Y \geq i). \tag{2.5}$$

Since $E[Y] \leq \frac{i}{2}$ and $\lfloor i\delta_{\text{in}}/(2n)\rfloor \geq \lfloor 32\max(\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}, \ln n)\rfloor > 2$, Lemma 2.2.3 implies $\Pr(Y \geq i) \leq \Pr(Y \geq 2E[Y]) \leq \exp\left(\frac{-i\delta_{\text{in}}}{16n} + \frac{1}{8}\right)$. The claim follows since

$$\begin{aligned}
\lambda \cdot \Pr(X \geq i) &\leq \exp\left(\frac{2\Delta_{\text{out}}\Delta_{\text{in}}}{\delta_{\text{in}}}\right) \cdot \Pr(Y \geq i) \\
&\leq \exp\left(\frac{-i\delta_{\text{in}}}{16n} + \frac{1}{8} + \frac{2\Delta_{\text{out}}\Delta_{\text{in}}}{\delta_{\text{in}}}\right) \\
&\leq \exp\left(\frac{-i\delta_{\text{in}}}{32n} + \frac{1}{8}\right) \\
&\leq \exp\left(\frac{-i\delta_{\text{in}}}{64n}\right).
\end{aligned}$$

(The first step follows from Equations (2.4) and (2.5). For the third step and fourth steps, note that the assumption $i \geq 64n\max(\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}^2, (\ln n)/\delta_{\text{in}})$ implies $i\delta_{\text{in}}/(32n) \geq 2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}$ and $i\delta_{\text{in}}/(64n) \geq 1/8$, respectively.) □

Lemma 2.4.12. *If $r \geq \min(\lceil 2e^2\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}\rceil, n)$, then every active edge sequence is, with high probability, $O(n\max(\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}^2, (\log n)/\delta_{\text{in}}))$-active.*

Proof. Let c denote an arbitrary positive real greater than or equal to 1, and let i denote the positive integer $\lceil 64cn\max(\Delta_{\text{out}}\Delta_{\text{in}}/\delta_{\text{in}}^2, (\ln n)/\delta_{\text{in}}))\rceil$.

For any nonnegative integer j, let p_j denotes the probability that there is a j-active edge sequence. Any j-active edge sequence σ is active, so the associated node

sequence $\mathrm{dst}(\sigma)$ is active. It follows from Lemma 2.4.5 that any j-active sequence has length at most n. In other words, $\ell \leq n$ for any j-active ℓ-sequence of edges. Furthermore, if $j > 0$ then the length of a j-active sequence is nonzero. Since any j-active ℓ-sequence of edges satisfies $\ell \leq n$, the condition $r = \min(\lceil 2e^2 \Delta_{\text{out}} \Delta_{\text{in}} / \delta_{\text{in}} \rceil, n)$ allows us to apply Lemmas 2.4.10 and 2.4.11. Applying these two lemmas, together with a union bound, we obtain $p_j \leq n^2 \exp(-j\delta_{\text{in}}/(64n))$ for $j > i$.

Let p denote the probability that there is a j-active edge sequence for some $j \geq i$. By a union bound, $p \leq \sum_{j \geq i} p_j$. Using the upper bound on p_j derived in the preceding paragraph, we find that p is upper bounded by an infinite geometric sum with initial term $n^2 \exp(-i\delta_{\text{in}}/(64n))$ and ratio $\exp(-\delta_{\text{in}}/(64n))$. Thus

$$\begin{aligned} p &= O((n^3/\delta_{\text{in}}) \exp(-i\delta_{\text{in}}/(64n))) \\ &= O(n^3 \exp(-c \max(\Delta_{\text{out}} \Delta_{\text{in}}/\delta_{\text{in}}, \log n))) \\ &= O(n^{3-c}). \end{aligned}$$

By setting c to a sufficiently large positive constant, we can drive p below any desired inverse polynomial threshold. The claim of the lemma follows. □

Lemma 2.4.13. *If $r \geq \min(\lceil 2e^2 \Delta_{\text{out}} \Delta_{\text{in}} / \delta_{\text{in}} \rceil, n)$, then the cover time of process R-RANK is, with high probability, $O(n \max(\Delta_{\text{out}} \Delta_{\text{in}} / \delta_{\text{in}}^2, (\log n)/\delta_{\text{in}}))$. The same asymptotic bound holds for the expected cover time.*

Proof. The high probability claim is immediate from Lemmas 2.4.4 and 2.4.12. The bound on the expected cover time then follows by Lemma 2.4.3. □

Theorem 2.4.1. *If both Δ_{in} and Δ_{out} are $O(\delta_{\text{in}})$, then there is an r in $O(\delta_{\text{in}})$ such that the cover time of process R-RANK is $O(n + \frac{n \log n}{\delta_{\text{in}}})$ with high probability. The same asymptotic bound holds for the expected cover time.*

Proof. Immediate from Lemma 2.4.13. □

The result of Theorem 2.4.1 matches the lower bound proved by Alon for process MIN and is thus optimal [4].

Note that as r tends to infinity, the behavior of process R-RANK converges to that of process P-RANK. Thus, the bounds of Theorem 2.4.1 also hold for process P-RANK.

2.5 Process R-RANK′

In this section we analyze a biased version of process R-RANK, which we call process R-RANK′. Process R-RANK′ is similar to process R-RANK, except that immediately after a selection, if the selected node is uncovered we cover it and move to the next selection. Otherwise, we proceed as in process R-RANK.

In our analysis, we find it helpful to consider another process, which we call process H. Process H runs in two phases. For the first phase, consisting of the first $cn \max(1, (\log n)/\delta_{\text{in}})$ rounds, we run process SELECT. At the end of phase 1, we remove from the graph all edges which did not have at least one end-point selected during phase 1. After the edge removal, we proceed to phase 2 where we begin to cover vertices as in process R-RANK.

Lemma 2.5.1. *If process H and process R-RANK′ use the same random rank assignment, infinite series of selections, and tie-breaking node order, the cover time of process R-RANK′ is at most the cover time of process H.*

Proof. We prove the stronger claim that if process H and process R-RANK′ use the same random rank assignment, infinite series of selections, and tie-breaking node order, then at the end of any round i, all nodes covered in process H are also covered in process R-RANK′.

Call a round i low if $i \leq cn \max(1, (\log n)/\delta_{\text{in}})$, and high otherwise. We call a node *marked* if it was selected in some low round.

We proceed by induction on i. For the base case, we consider any low round i. In these rounds, process H covers no nodes, so there is nothing to prove.

Now, assume i is high. Let u be the node selected in round i in both process R-RANK′ and process H. If no node is covered in process H, the claim follows from the induction hypothesis. Now assume node v is covered in process H in round i. If v is covered in process R-RANK′ in some round prior to round i, there is nothing to prove. Thus, assume that v is not covered in process R-RANK′ prior to round i. We now complete the induction step by arguing that v must also be covered in process R-RANK′ in round i.

If v is marked, then v is covered in process R-RANK′ in a low round since it was selected in a low round. But, v is not covered in process R-RANK′ prior to round i, so v is unmarked. Since process H selects u and covers v in round i, (u, v)

is not removed by process H at the end of phase 1. Thus, u and v cannot both be unmarked, so u is marked.

It follows that u is not equal to v and u is already covered in process R-RANK$'$ as it was selected in a low round. Since u is marked, it has the same set of outgoing neighbors in both processes, i.e., no edge (u, w) is thrown away in process H at the end of the first phase.

Let S (resp., T) be the uncovered outgoing neighbors of u in process R-RANK$'$ (resp., process H) at the beginning of round i. By the induction hypothesis, S is contained in T. Since both processes use the same random ranks and tie-breaking node order, the neighbor selection procedure gives well defined order of the nighbors of u. Since $S \subseteq T$ and v is the minimum order node in T and belongs to S, v is the minimum order node in S. Thus v also is covered in round i in process R-RANK$'$. □

Lemma 2.5.2. *If $r \geq \min(\lceil 2e^2 \Delta_{\text{out}} \Delta_{\text{in}} / \delta_{\text{in}} \rceil, n)$, then the cover time of process R-RANK$'$ is, with high probability, $O(n \max(\Delta_{\text{out}} \Delta_{\text{in}} / \delta_{\text{in}}^2, (\log n)/\delta_{\text{in}}))$. The same asymptotic bound holds for the expected cover time.*

Proof. We run a copy of process R-RANK$'$ in parallel with a copy of process H, using the same random ranks, selections, and tie-breaking node order.

We call phase 1 of process H successful if at least $\delta_{\text{in}}/4$ of every node's in-neighbors are selected. If phase 1 is unsuccessful, we over estimate the cover time of process R-RANK$'$ by the $O(n \log n)$ cover time of coupon collector. If phase 1 is successful, by Lemma 2.5.1 we may overestimate the cover time of process R-RANK$'$ with the cover time bound of process H. To find the cover time bound of process H, we add the number of rounds during phase 1, to the cover time bound of process R-RANK during phase 2. We apply Lemma 2.4.13 to phase 2 of process H where the graph has in-degree at least $\delta_{\text{in}}/4$, to get a cover time bound of $O(\max(\Delta_{\text{out}} \Delta_{\text{in}} / \delta_{\text{in}}^2, (\log n)/\delta_{\text{in}}))$ for process H. Since the bound on the cover time of process H is both with high probability and in expectation, if phase 1 is successful with high probability, the same bound holds for process R-RANK$'$.

All that remains to be shown is the required result is that phase 1 is successful with high probability.

Consider a specific node w. The probability of selecting a node in $\Gamma_{\text{in}}(w)$ on any selection is a Bernoulli random variable with success probability at least δ_{in}/n.

The number of selections in $\Gamma_{\text{in}}(w)$ during phase 1 is the sum of $cn \max(1, (\log n)\delta_{\text{in}})$ such independent Bernoulli random variables. Thus, by Lemma 2.2.4, the probability of getting less than $(c/2)\max(\delta_{\text{in}}, \log n)$ selections in $\Gamma_{\text{in}}(w)$ during phase 1 is at most $\exp((c/12)\max(\delta_{\text{in}}, \log n))$, which is an arbitrary inverse polynomial by choosing a large enough constant c.

Given that $(c/2)\max(\delta_{\text{in}}, (\log n))$ selections during phase 1 select a vertex in $\Gamma_{\text{in}}(w)$, we apply Lemma 2.2.6. To do so, let the variables in the lemma be $n = |\Gamma_{\text{in}}(w)| \geq \delta_{\text{in}}$, and $j = (c/2)\max(\delta_{\text{in}}, (\log n))$ which is also at least δ_{in} if we set $c \geq 2$. Thus, Lemma 2.2.6 tell us that the probability less than $\frac{\delta_{\text{in}}}{4}$ distinct nodes of $\Gamma_{\text{in}}(w)$ are selected during phase 1 of process H is at most $\exp(\frac{c}{2}\max(\delta_{\text{in}}, \log n))$, which is an arbitrary inverse polynomial by selecting a large enough constant c. Taking the union bound over all nodes in the graph shows that phase 1 is successful with high probability. □

2.6 Lower Bounds

In this section, we show lower bound results on the cover times of the processes process A-RANK and process A-RANK′ defined in Section 2.1. These results establish that the method of picking which uncovered neighbor to cover does make a difference to the resulting expected cover time. While the full proofs of the two theorems are rather lengthy, the main ideas are straightforward. We summarize these main ideas in the two proof outlines that follow. The main technical tools employed in the full proofs are Chernoff bounds and Azuma's inequality (see, e.g., [40, 5]). Note that our lower bounds hold even if we restrict attention to the special class of directed graphs where edge (u, v) is present if and only if edge (v, u) is present; below we refer to such graphs as undirected.

Theorem 2.6.1. *For all n, there is an n-node undirected graph G in which each node has degree $\Theta(\log n)$, and an assignment of ranks 1 through n to the nodes of G, such that process A-RANK has cover time $\Omega(n\sqrt{(\log n)/\log\log n}) = \omega(n)$.*

Proof sketch: Fix n and construct G as follows. First, partition the n nodes into ℓ *levels*, numbered from 0 to $\ell - 1$, so that the following conditions hold: level 0 contains $n/2$ nodes; successive levels have a geometrically decreasing number of nodes with ratio a $= \sqrt{(\log n)/\log\log n}$; level $\ell - 1$ is the only level with fewer than

$\sqrt{n}$ nodes. Thus $\ell = \Theta((\log n)/\log\log n)$. Assign ranks 1 through n to the nodes in such a way that nodes on lower-numbered levels have lower ranks. For each node u at level i, select $\Theta(\log n)$ nodes at random from each of levels i and $i-1$ (with replacement), and add an edge from u to each selected node. (If node u is at level 0, then only select nodes from level 0.) We let s_i denote the number of nodes on level i and define τ_i to be ni/ca, where c is a sufficiently large constant. We call a level *crowded* if more than half of the nodes on that level are covered.

We inductively show that, with high probability, level i is not crowded until τ_i. For the base case, $i = 0$, the claim is trivially true since τ_0 is 0. For induction, we have two subclaims. First, using the inductive hypothesis we can show that, with high probability, only a small constant fraction of level i is covered between rounds 0 and τ_{i-1}. Second, we can show that, with high probability, only a small constant fraction of level i is covered between rounds τ_{i-1} and τ_i. This completes the proof of the inductive claim. The theorem results from setting $i = \ell$ in the inductive claim. Specifically, with high probability, level ℓ is not crowded on round $\tau_\ell = \Omega(n\sqrt{(\log n)/\log\log n})$, which gives the theorem statement. All that remains to be shown are the two subclaims.

First, we show that, with high probability, only a small constant fraction of level i is covered between rounds 0 and τ_{i-1}. We define a *bad set* as a set of nodes from levels i and $i-1$ with no uncovered neighbor on level $i-1$ on round τ_{i-1}. Inductively assuming that level $i-1$ is not crowded until τ_{i-1}, we can use the probabilistic method to show that with high probability no bad set of size greater than s_{i+1} exists. We over estimate the covers on level i from round 0 to τ_{i-1}, by assuming all selections throughout the specified rounds on level $i+1$ or on a bad set of size s_{i+1} result in covers on level i. Thus, with high probability, the rate of coverage on level i between rounds 0 to τ_{i-1} is $2s_{i+1}/n$. The expected number of nodes covered is then $2\tau_{i-1}s_{i+1}/n = o(s_i)$. Using Chernoff bounds, we can show that, with high probability, only a small constant fraction of level i is covered by round τ_{i-1}.

Second, we show that, with high probability, only a small constant fraction of level i is covered between rounds τ_{i-1} and τ_i. We over estimate the covers on level i from round τ_{i-1} to τ_i by assuming that all selections on levels $i-1$, i, and $i+1$ result in a cover on level i. The rate of coverage is upper bounded by $2s_{i-1}/n$. From the definitions of s_{i-1}, τ_{i-1} and τ_i, we find the expected number of nodes

covered on level i from round τ_{i-1} to τ_i is a small constant fraction of s_i. Using Chernoff bounds, and combining the results from this and the previous paragraph, we can show that with high probability, level i does not become crowded until τ_i, completing the inductive claim.

Theorem 2.6.2. *For all n, there is an n-node undirected graph G in which each node has degree $\Theta(\log n)$, and an assignment of ranks 1 through n to the nodes of G, such that process A-RANK′ has cover time $\Omega(n \log \log n) = \omega(n)$.*

Proof sketch: The proof of this theorem proceeds in much the same was as the proof for Theorem 2.6.1. Again, partition the n nodes into ℓ levels, numbered from 0 to $\ell - 1$ with level 0 containing about $n/2$ nodes. However, this time, let the ratio a of the number of nodes between successive levels be $(\log n)^{1/4}$. We restrict the number of levels ℓ to $\Theta((\log n)^{3/8} \log \log n)$. Again, assign ranks 1 through n to the nodes in such a way that nodes on lower-numbered levels have lower ranks. For each node u at level i, select $\Theta(\log n)$ nodes at random from each of levels i and $i - 1$ (with replacement), and add an edge from u to each selected node. (If node u is at level 0, then only select nodes from level 0.) Again, we let s_i denote the number of nodes on level i.

This time, however, we define τ_i to be a more conservative $\frac{ni}{c\mathrm{a}^{3/2}}$, where c is a sufficiently large constant. Furthermore, we change the meaning of *crowded* to denote that more than a $1 - 1/\sqrt{\mathrm{a}}$ fraction of the nodes on that level are covered. The motivation for these changes is that node covers resulting from the bias towards the selected node quickly cover a significant fraction of the nodes in each level.

Again, we inductively show that, with high probability, level i is not crowded until τ_i. For the base case, $i = 0$, the claim is trivially true since τ_0 is 0. For induction, we have two subclaims. First, using the inductive hypothesis we can show that, with high probability, only a $1 - 2/\sqrt{\mathrm{a}}$ fraction of level i is covered between rounds 0 and τ_{i-1}. Second, we can show that, with high probability, only a further $1/\sqrt{\mathrm{a}}$ fraction of level i is covered between rounds τ_{i-1} and τ_i. This completes the proof of the inductive claim. The theorem results from setting $i = \ell$ in the inductive claim. Specifically, with high probability, level ℓ is not crowded on round $\tau_\ell = \Omega(n \log \log n)$, which gives the theorem statement. All that remains to be shown are the two subclaims.

First, we show that, with high probability, only a $1 - 2/\sqrt{\mathrm{a}}$ fraction of level

i is covered between rounds 0 and τ_{i-1}. We define a *bad set* as a set of nodes from levels i and $i-1$ with no uncovered neighbor on level $i-1$ on round τ_{i-1}. Inductively assuming that level $i-1$ is not crowded until τ_{i-1}, we can use the probabilistic method to show that with high probability no bad set of size greater than s_{i+1} exists. To show the desired result, we analyse an over-estimate of the covers on level i from round 0 to τ_{i-1} in to two parts. First, we assume that all selections throughout the specified rounds on level $i+1$ or a bad set of size s_{i+1} result in covers on level i. We show that with high probability, covers of this type cover no more than a $1/\sqrt{\mathrm{a}}$ fraction of level i. Second, using Azuma's inequality, we show that the covers due to bias towards the selected node on level i cover no more than a $1-3/\sqrt{\mathrm{a}}$ fraction of the nodes on level i with high probability.

In the final remaining claim, we show that, with high probability, only an $1/\sqrt{\mathrm{a}}$ fraction of level i is covered between rounds τ_{i-1} and τ_i. We over estimate the covers on level i from round τ_{i-1} to τ_i by assuming that all selections on levels $i-1$, i, and $i+1$ result in a cover on level i. The rate of coverage is upper bounded by $2s_{i-1}/n$. From the definitions of s_{i-1}, τ_{i-1} and τ_i, we find the expected number of nodes covered on level i from round τ_{i-1} to τ_i is a $1/\sqrt{\mathrm{a}}$ fraction of s_i. Using Chernoff bounds, and combining the results from this and the previous paragraph, we can show that with high probability, level i does not become crowded until τ_i, completing the inductive claim.

2.7 Concluding Remarks

For completeness, the reader should notice that assigning r a greater value in the proof of Lemma 2.4.11 does not alter the result. It is this fact that makes the locally assigned random ranks more general than the random permutation discussed in the introduction. If we let $r = 2^n$, the random ranks will fix a random permutation with high probability.

We also note that process UNI is equivalent to a process where each node selects a uniformly random permutation of the vertices. Then, when a node is selected, we pick the min-rank neighbor based on the selected node's ranks – as opposed to the global ranks in process P-RANK. This once again highlights the similarities between process P-RANK and process UNI.

Furthermore, we note that process CC on a directed graph can be viewed

as a process on a family of sets. Let there be s, not necessarily distinct, sets from a universe of n elements. In each round, we select a set uniformly at random and cover an uncovered element from that set. When $s = n$ and each of the n sets is the set of out-neighbors of a particular node from the directed graph, the two processes are equivalent. Our proof techniques, can be used to derive results in this set based process.

Chapter 3

Selfishness

3.1 Introduction

In this chapter, we study the core of a large set of games, a subset of assignment games, which we term buyer-supplier games [10, 55] [56, Chapter 6]. We are primarily concerned with efficient computations over the set of vectors belonging to the core of buyer-supplier games. Before diving into an overview of buyer-supplier games, we present some connections between our work and the existing literature.

3.1.1 Related Work

Though suggested by Edgeworth as early as 1881 [16], the notion of the core was formalized by Gillies and Shapley [23, 53], extending von Neumann and Morgenstern's work on coalitional game theory [59]. Recently, Goemans and Skutella studied the core of a cost sharing facility location game [24]. In their paper, Goemans and Skutella are primarily interested in using core vectors as a cost sharing indicator, to decide how much each customer should pay for opening the facility used by the customer. Goemans and Skutella show that, in general, the core of the cost sharing facility location game they study is empty. In contrast, for the buyer-supplier games we study, the core is always nonempty. Additionally, in our work we do not view vectors in the core as an indication of cost shares but rather as rational outcomes of negotiation amongst the players in the buyer-supplier game. Pál and Tardos extend the work of Goemans and Skutella by developing a mechanism for the cost sharing facility location game which uses the concept of an approximate core [44].

There has been great interest in comparing the game's best outcome to the best equilibrium outcome, where the term best is based on some objective function. For example, one may wish to compare the outcome maximizing the net utility for all players in the game against the best possible Nash equilibrium, with respect to net utility. Papadimitriou termed one such comparative measure as the price of anarchy [45]. Roughgarden and Tardos have studied the price of anarchy in the context of routing [50, 51, 52].

In this chapter, we introduce a quantity with a similar motivation to that of the price of anarchy. Solution concepts often yield multiple predictions, or equilibria. In actual game play, however, only one of the equilibria can be chosen by the game's players. Experiments show that conditions outside the game, such as societal pressures or undue attention to a specific player, focus the players' attention on the point of a single equilibrium, which then becomes the outcome of the game. This is a common notion in game theory called the focus point. A player may receive different payoffs in different equilibria. How much is the player willing to pay for a good focus point? We define the *focus point price* with respect to a given player as the difference between the maximum and minimum equilibrium payoffs to the player. Stated succinctly, focus point price answers the question: If we are constrained to play in equilibrium, how much can we lose by playing the wrong equilibrium?

Recently, Garg et al. studied transferable utility games they call coalitional games on graphs [22]. Coalitional games on graphs are a proper subset of buyer-supplier games, which can be derived by setting the buyer's internal cost, Bcost, to zero (see Section 3.1.3 and Lemma 3.3.21). For some buyer-supplier games, for example the buyer-supplier facility location game, it does not appear that the game can be described with Bcost fixed to zero.

Garg et al. study the concepts of "frugality" and "agents are substitutes." They show that suppliers are substitutes if and only if the core of the game forms a lattice. In buyer-supplier games, suppliers are not always substitutes. We show, in Lemma 3.4.4, that if suppliers are substitutes, we can optimize over the core by solving a polynomially sized linear program. Garg et al. and, more recently, Karlin et al. study and characterize the frugality certain auction mechanisms; the focus point price concept introduced in this chapter is quite different from frugality [31].

A third difference between Garg et al. and this work comes from the fact that, similarly to the economics literature, Garg et al. are mainly concerned with

the characterization of the core: When does the core form a lattice? How do core vectors relate to auctions? We, on the other hand, are mainly concerned with characterizing optimization over the core. Our main results are in the flavor of Deng and Papadimitriou, in that we are interested with the complexity of computing using game theoretic characterizations [15].

Faigle and Kern study optimization over the core for submodular cost partition games [18]. Faigle and Kern exhibit a generic greedy-type algorithm for optimization of any linear function over the core of partition games whose value function is both submodular and *weakly increasing*, a property they define.

The greedy framework of Faigle and Kern captures certain buyer-supplier games, such as the buyer-supplier minimum spanning tree game. However, even some buyer-supplier games derived from problems that admit greedy solutions, such as the buyer-supplier shortest path game, are not amenable to the approach of Faigle and Kern. In this chapter, we do not restrict ourselves to greedy algorithms. By making use of the ellipsoid method, we are able to give polynomial time algorithms for optimization over the core of any buyer-supplier game for which the underlying minimization problem is solvable in polynomial time.

To provide the reader with a simple, concrete example of optimization over the core of a buyer-supplier game, towards the end of this chapter, we focus our attention on the buyer-supplier minimum spanning tree game. We give a simple greedy algorithm for this problem, which is a minor extension of Kruskal's minimum spanning tree algorithm. A greedy algorithm is provided by the work of Faigle and Kern, but their exposition involves a good deal of machinery. Our exposition is completely elementary.

Several methods, apart from buyer-supplier games, are known for transforming a combinatorial optimization problem into a game. The cores of these transformations have also been extensively studied. For example, Deng et al. show results on core non-emptyness, distinguishability of core vectors, and finding core vectors for one such transformation [14]. Caprara et al. continue the work of Deng et al. by considering a certain optimization over the set of core vectors for this alternate transformation [12].

3.1.2 Main Contributions

There has been increased interest from the theoretical computer science community in game theory. While problem-specific solutions may give us insight, to leverage the full power of decades of study in both research areas, we must find generic computational solutions to game theoretic problems. Indeed, others have already realized this need [6, 46]. In this chapter, we continue this line of work by deriving generic results for computing with core solutions in a large class of games.

The core of buyer-supplier games in the transferable utility setting is characterized by Shapley and Shubik [55]. As a minor contribution, we extend their result by showing that the core in the non-transferable utility setting is the same as the core with transferable utilities. Our primary contributions are as follows:

1. While previous work in the economics literature has concentrated on characterizing the core of buyer-supplier games and relating core vectors to auctions, our main interest is in optimizing over the set of core vectors [10]. We provide a generally applicable algorithm, based on the ellipsoid method, for optimizing over the core. If the original minimization problem is solvable in polynomial time, we show that it is possible to optimize linear functions of core vectors in polynomial time.

2. We fully characterize optimization over the core of buyer-supplier games by using a polynomial time reduction to show that if the original minimization problem is not solvable in polynomial time, it is impossible, in polynomial time, to test if an arbitrary vector is in the core of the buyer-supplier game.

3. We introduce the concept of focus point price. Our positive computational results give a polynomial time algorithm for computing the buyer's focus point price in buyer-supplier games when the underlying minimization problem is solvable in polynomial time. When the underlying minimization problem is not solvable in polynomial time, we show that it is impossible to approximate the buyer's focus point price to within *any* multiplicative factor.

3.1.3 Overview of Buyer-Supplier Games

The definition of a buyer-supplier game, given in Section 3.2.1, is self-contained and does not require an argument. However, it is also possible to transform a combina-

torial minimization problem into a buyer-supplier game. Consider a combinatorial minimization problem of the following form. We have some finite set of elements $\mathcal{C}$. We designate some subsets of $\mathcal{C}$ as feasible. To capture feasibility, we use a predicate $P : 2^{\mathcal{C}} \to \{0, 1\}$, where the predicate is one on all feasible subsets of $\mathcal{C}$. With each feasible set $\mathcal{A} \subseteq \mathcal{C}$, we associate a non-negative cost $f(\mathcal{A})$. The combinatorial minimization problem can then be captured by the function MinProb : $2^{\mathcal{C}} \to \Re_+$ defined by

$$\text{MinProb}(\mathcal{B}) = \min_{\substack{\mathcal{A} \subseteq \mathcal{B} \\ P(\mathcal{A}) = 1}} f(\mathcal{A})$$

where $\Re_+$ denotes the non-negative real numbers.

To transform the above minimization problem into a buyer-supplier game, we associate a player with each element of $\mathcal{C}$; we call such players suppliers. We also add another player whom we call the buyer. In the game, the buyer wishes to purchase a feasible subset of $\mathcal{C}$. The suppliers, on the other hand, are offering their membership to the buyer's set at a price.

To fully specify the game's model of a realistic interaction, we let M designate the maximum investment the buyer is willing to spend on a feasible set. We decompose f such that $f(\mathcal{A}) = \text{Bcost}(\mathcal{A}) + \sum_{a \in \mathcal{A}} \tau(a)$, where $\tau(a)$ is an internal cost for supplier a to be present in the buyer's set and $\text{Bcost}(\mathcal{A})$ is an internal cost to the buyer for purchasing this specific feasible set. In general, many such decompositions are possible, and they produce different games. However, when specifically applying the core solution concept, Lemma 3.3.21 shows that all such decompositions are equivalent. Though it is not necessary, to remove special cases in our statements, it is convenient to let $\text{Bcost}(\mathcal{A}) = M$ when $\mathcal{A} = \emptyset$ or $\mathcal{A}$ is not feasible.

Now that we have determined the internal costs for the buyer and the suppliers, we can specify the game. The buyer-supplier game is specified by the tuple $(\mathcal{C}, \tau, \text{Bcost})$. The strategy set for the buyer is the power set of $\mathcal{C}$. By playing $\mathcal{A} \subseteq \mathcal{C}$, the buyer chooses to purchase the membership of the suppliers in $\mathcal{A}$. The strategy set for every supplier $a \in \mathcal{C}$ is the non-negative real numbers, indicating a bid or payment required from the buyer for the supplier's membership.

For any supplier $a \in \mathcal{C}$, we let $\beta(a)$ denote the associated bid. Let $\mathcal{A}$ be the set of suppliers chosen by the buyer. The payoff for the buyer is $M - \text{Bcost}(\mathcal{A}) -$

$\sum_{a \in \mathcal{A}} \beta(a)$. The payoff for a supplier not in $\mathcal{A}$ is 0. The payoff for a supplier a in $\mathcal{A}$ is $\beta(a) - \tau(a)$.

Since we are applying the solution concept of the core, one may think of the game play as follows. All the players in the game sit down around a negotiating table. All the players talk amongst themselves until they reach an agreement which cannot be unilaterally and selfishly improved upon by any subset of the players. Once such an agreement is reached, game play is concluded. Since no subset of the players can unilaterally and selfishly improve upon the agreement, rationality binds the players to follow the agreement.

The fully formal definition of a buyer-supplier game is given in Section 3.2.1. The transformation process described above can be used to create buyer-supplier games from most combinatorial minimization problems. For example, minimum spanning tree, Steiner tree, shortest path, minimum set cover, minimum cut, single- and multi-commodity flow can all be used to instantiate a buyer-supplier game.

As a concrete example and interpretation of a buyer-supplier game, consider the buyer-supplier minimum spanning tree game. In this game, a company owns factories on every node of a graph. The company wishes to connect the factories by purchasing edges in the graph. Each edge is owned by a unique supplier player. Each supplier has an internal cost associated with the company's usage of the edge. The company has a maximum amount of money it is willing to spend on purchasing edges. Depending on the transportation conditions of a particular edge, the company may have some internal cost associated with choosing that particular edge. The buyer-supplier game paradigm yields similarly natural games when applied to other minimization problems.

In this chapter we will be concerned with efficient computation over the set of core vectors. For the rest of the chapter, when we say polynomial time, we mean time polynomial in the size of the parameter $\mathcal{C}$, which is also polynomial in the number of players of the buyer-supplier game.

3.1.4 Organization of the Chapter

In Section 3.2 we define buyer-supplier games and the core of a game. In Section 3.3 we characterize the core of buyer-supplier games. In Section 3.4 we give positive computational results, namely the generic algorithm for optimizing over the set

of core vectors. In Section 3.5 we give negative computational results by showing polynomial time equivalence between several related problems. In Section 3.6 we give the problem-specific combinatorial algorithm for the buyer-supplier game arising from the minimum spanning tree problem.

3.2 Definitions

In this section, we formally define buyer-supplier games and give the game theoretic definitions required for our analysis.

3.2.1 Buyer-Supplier Games

Let $\mathcal{C}$ be a finite set and M be a non-negative real number. Let τ be a function from $\mathcal{C}$ to $\Re_+$. Let Bcost be a function from $2^{\mathcal{C}}$ to $\Re_+$ such that $\text{Bcost}(\emptyset) = M$. The simplifying condition that $\text{Bcost}(\emptyset) = M$ is not required. We explain the condition's purpose later in this section. For $\mathcal{A} \subseteq \mathcal{C}$, let $\text{Eval}(\tau, \text{Bcost}, \mathcal{A})$ denote $\text{Bcost}(\mathcal{A}) + \sum_{a \in \mathcal{A}} \tau(a)$. For $\mathcal{A} \subseteq \mathcal{C}$, let $\text{MinEval}(\tau, \text{Bcost}, \mathcal{A})$ denote $\min_{\mathcal{B} \subseteq \mathcal{A}} \text{Eval}(\tau, \text{Bcost}, \mathcal{B})$. We will omit the parameters τ and Bcost from the functions $\text{Eval}(\tau, \text{Bcost}, \mathcal{A})$ and $\text{MinEval}(\tau, \text{Bcost}, \mathcal{A})$ when the value is clear.

Given a tuple $(\mathcal{C}, \tau, \text{Bcost})$, we proceed to define a buyer-supplier game. Associate a player with each element of $\mathcal{C}$. Call the players in $\mathcal{C}$ suppliers. Let there also be another player, μ, whom we call the buyer. Let $\mathcal{P} = \mathcal{C} \cup \{\mu\}$ be the set of players for the buyer-supplier game.

The strategy for supplier a is a tuple $(\beta(a), p_a)$ with $\beta(a) \in \Re_+$ and $p_a : \mathcal{P} \to \Re_+$. The first element, $\beta(a)$, represents supplier a's bid to the buyer, requiring the buyer to pay $\beta(a)$ for using the supplier's services. The second element, p_a, represents the non-negative side payments supplier a chooses to make to the game's players. By $p_a(b)$ we denote the side payment a makes to player b.

The strategy for the buyer, μ, is a tuple $(\mathcal{A}, p_\mu)$ where $\mathcal{A} \subseteq \mathcal{C}$ and $p_\mu : \mathcal{P} \to \Re_+$. The first element, $\mathcal{A}$, represents the suppliers chosen by the buyer for a purchase. Similarly to a supplier, the second element, p_μ, represents the non-negative side payments the buyer chooses to make to the game's players.

For each player $a \in \mathcal{P}$ we denote the player's strategy set by $\mathcal{S}_a$. For a set of players $\mathcal{A} \subseteq \mathcal{P}$, we denote the set of strategies $\bigotimes_{a \in \mathcal{A}} \mathcal{S}_a$ by $\mathcal{S}_\mathcal{A}$. We call elements of $\mathcal{S}_\mathcal{A}$ strategy vectors. We index strategy vectors from $\mathcal{S}_\mathcal{A}$ by the elements of $\mathcal{A}$.

We now define the utility function for each player. Suppose strategy $s \in \mathcal{S}_\mathcal{P}$ is played. Specifically, suppose that $(\mathcal{A}, p_\mu) \in \mathcal{S}_\mu$ and $(\beta(a), p_a) \in \mathcal{S}_a$ for each $a \in \mathcal{C}$ are played. The utility function for buyer is $u_\mu(s) = M - [\text{Bcost}(\mathcal{A}) + \sum_{a\in\mathcal{A}} \beta(a)] + [\sum_{b\in\mathcal{P}} p_b(\mu) - \sum_{b\in\mathcal{P}} p_\mu(b)]$. The utility for a supplier a in $\mathcal{A}$ is $u_a(s) = \beta(a) - \tau(a) + [\sum_{b\in\mathcal{P}} p_b(a) - \sum_{b\in\mathcal{P}} p_a(b)]$. The utility for a supplier a not in $\mathcal{A}$ is $u_a(s) = [\sum_{b\in\mathcal{P}} p_b(a) - \sum_{b\in\mathcal{P}} p_a(b)]$.

Interpreting, the buyer begins with a total of M utility and chooses to make a purchase from each supplier in $\mathcal{A}$. The buyer gives $\beta(a)$ to each supplier $a \in \mathcal{A}$ and loses an extra $\text{Bcost}(\mathcal{A})$ from the initial M utility. Each supplier a in $\mathcal{A}$ receives the bid payment from the buyer and loses $\tau(a)$ because the supplier must perform services for the buyer. The distribution of sidepayments completes the utility functions. The requirement that $\text{Bcost}(\emptyset) = M$ lets the strategy $\emptyset$ stand as a "don't play" strategy for the buyer. To remove the requirement, we could introduce a specific "don't play" strategy to the buyer's strategy set, however this creates a special case in most of our proofs.

Let the sidepayment game we have defined be denoted SP. Let NOSP denote the same game with the additional requirement that all sidepayments be fixed to zero. In other words, in NOSP we restrict the strategy set for each $a \in \mathcal{P}$ so that p_a is identically zero.

3.2.2 Game Theoretic Definitions

All of the definitions in this section closely follow those of Shubik [56, Chapter 6].

We call a vector in $\Re^{|\mathcal{P}|}$, indexed by $a \in \mathcal{P}$, a *payoff vector*. We say a payoff vector π is *realized* by a strategy vector $s \in \mathcal{S}_\mathcal{P}$ if $\pi_a = u_a(s)$ for all $a \in \mathcal{P}$.

Let π be a payoff vector and s be a strategy vector in $\mathcal{S}_\mathcal{A}$ for $\mathcal{A} \subseteq \mathcal{P}$. Let t be any strategy vector in $\mathcal{S}_\mathcal{P}$ such that the restriction of t to the coordinates in $\mathcal{A}$ is equal to s. If for all t and for all $a \in \mathcal{A}$ we have $\pi_a \leq u_a(t)$, we say that the players in $\mathcal{A}$ can *guarantee* themselves payoffs of at least π by playing s.

We use Shubik's alpha theory to define our characteristic sets [56, pp. 134-136]. Thus for a set of players $\mathcal{A} \subseteq \mathcal{P}$, we define the characteristic set, $V(\mathcal{A})$, to be the set of all payoff vectors π such that there is a strategy vector $s \in \mathcal{S}_\mathcal{A}$, possibly dependent on π, with which the players in $\mathcal{A}$ can guarantee themselves payoffs of at least π. In the transferable utility setting, SP, the characteristic sets can be

replaced with a characteristic function. Given the definitions of the utility functions in Section 3.2.1, the characteristic function $\tilde{V}(\mathcal{A})$ for a set of players $\mathcal{A}$ is equal to $M - \text{MinEval}(\tau, \text{Bcost}, \mathcal{A} - \{\mu\})$.

We say that a set $\mathcal{A} \subseteq \mathcal{P}$ of *players are substitutes* if $\tilde{V}(\mathcal{P}) - \tilde{V}(\mathcal{P} - \mathcal{B}) \geq \sum_{a \in \mathcal{B}} \tilde{V}(\mathcal{P}) - \tilde{V}(\mathcal{P} - \{a\})$ for all $\mathcal{B} \subseteq \mathcal{A}$.

We say that a payoff vector π dominates a payoff vector ν through a set $\mathcal{A} \subseteq \mathcal{P}$ if $\pi_a > \nu_a$ for all $a \in \mathcal{A}$. In other words, π dominates ν through $\mathcal{A}$ when each player in $\mathcal{A}$ does better in π than in ν.

For a set of players $\mathcal{A} \subseteq \mathcal{P}$, we define $D(\mathcal{A})$ as the set of all payoff vectors which are dominated through $\mathcal{A}$ by a payoff vector in $V(\mathcal{A})$. Interpreting, the players in $\mathcal{A}$ would never settle for a payoff vector $\pi \in D(\mathcal{A})$ since they can guarantee themselves higher payoffs than those offered in π.

The *core* of a game consists of all $\pi \in V(\mathcal{P})$ such that $\pi \notin D(\mathcal{A})$ for all $\mathcal{A} \subseteq \mathcal{P}$.

3.3 A Characterization of the Core

The characterization of the core of buyer-supplier games in the transferable utility setting was done by Shapley and Shubik [55]. In this section, we show the surprising result that the same characterization holds in the non-transferable utility setting. In general, it is not the case that the core of the transferable utility and non-transferable utility versions of a game are the same. For example, the buyer may be able to use bribes to alter the bidding strategies of some suppliers, and thus reduce the bids of other suppliers. The following condition characterizes the core of buyer-supplier games. A payoff vector π is in the core of a buyer-supplier game defined by $(\mathcal{C}, \tau, \text{Bcost})$ if and only if it satisfies

$$\begin{aligned}
\pi_a &\geq 0 && \forall a \in \mathcal{P}, \\
\sum_{a \in \mathcal{A}} \pi_a &\leq \text{MinEval}(\tau, \text{Bcost}, \mathcal{C} - \mathcal{A}) - \text{MinEval}(\tau, \text{Bcost}, \mathcal{C}) && \forall \mathcal{A} \subseteq \mathcal{C}, \\
\pi_\mu &= M - \text{MinEval}(\tau, \text{Bcost}, \mathcal{C}) - \sum_{a \in \mathcal{C}} \pi_a.
\end{aligned}$$

We prove the result from first principles. In Section 3.3.1, we give some preliminary lemmas for the games NOSP and SP. In Section 3.3.2, we show that the

core of SP is the same as the core of NOSP. In Section 3.3.3, we give a characterization of the core of NOSP.

3.3.1 Preliminary Lemmas on the Core of NOSP and SP

This section contains some preliminary lemmas that are useful in characterizing the core of NOSP and SP.

For some of our proofs it is convenient to think of SP as a two stage distribution of wealth, where the strategy $s \in \mathcal{S}_{\mathcal{P}}$ determines the utility transfers. Initially, the buyer has M utility and all suppliers have zero utility. In the first stage, the buyer gives $\beta(b)$ to each supplier $b \in \mathcal{A}$ and loses an extra $\text{Bcost}(\mathcal{A})$ from the initial M utility. Also in the first stage, each supplier $b \in \mathcal{A}$ loses $\tau(b)$ of utility. In the second stage, side payments are distributed.

Lemma 3.3.1. *Let $s \in \mathcal{S}_{\mathcal{A}\cup\mu}$ be such that $s_\mu = (\mathcal{A}, p_\mu)$. If s guarantees the players in $\mathcal{A} \cup \mu$ payoffs of at least $\pi \in \Re^{|\mathcal{A}\cup\mu|}$, then there is a $t \in \mathcal{S}_{\mathcal{A}\cup\mu}$ such that*

- *All side payments from players in $\mathcal{A} \cup \mu$ to players in $\mathcal{A} \cup \mu$ are fixed to zero in t*
- $t_\mu = (\mathcal{A}, 0)$
- *t also guarantees payoffs of at least π.*

Proof. We show how to sequentially remove the specified side payments while maintaining the payoff guarantee.

Let a and b be suppliers in $\mathcal{A}$. Let $s_a = (\beta(a), p_a)$ and $s_b = (\beta(b), p_b)$.

First, consider a supplier to supplier payment. Suppose that $p_a(b) = \lambda$, that is, supplier a pays λ to supplier b. Because both a and b are in $\mathcal{A}$, we can achieve the same utility transfer as the side payment by setting the side payment to zero and changing $\beta(a)$ to $\beta(a) - \lambda$ and $\beta(b)$ to $\beta(b) + \lambda$. Thus, we can zero out the side payment from a to b.

Now, consider a supplier to buyer payment. Suppose that $p_a(\mu) = \lambda$. In other words supplier a pays λ to the buyer. We can achieve the same utility transfer as the side payment by setting the side payment to zero and changing $\beta(a)$ to $\beta(a) - \lambda$. Thus, we can zero out the side payment from a to μ.

A similar change works for a payment from the buyer to a supplier. □

Lemma 3.3.2. *Let strategy vector $s \in \mathcal{S}_{\mathcal{P}}$ realize payoff vector π. If $s_\mu = (\mathcal{A}, p_\mu)$ and there exists $a \in \mathcal{C} - \mathcal{A}$ such that $\pi_a > 0$, then $\pi \in D(\mathcal{A} \cup \mu)$ in both* SP *and* NOSP*.*

Proof. Since all side payments are zero in NOSP, it is impossible for π_a to be greater than zero. Thus, the statement is trivial for NOSP.

Consider the two stage distribution of wealth interpretation of SP. Since there exists $a \in \mathcal{C} - \mathcal{A}$ such that $\pi_a > 0$, in s there is a net flow of side payments from $\mathcal{A} \cup \mu$ to $\mathcal{C} - \mathcal{A}$ in stage two of SP. Instead of following strategy s, the players in $\mathcal{A} \cup \mu$ can set to zero all side payments going from $\mathcal{A} \cup \mu$ to $\mathcal{C} - \mathcal{A}$. With this action, at least π_a more utility stays in $\mathcal{A} \cup \mu$ at the end of stage two. The players in $\mathcal{A} \cup \mu$ can use side payments amongst themselves so that each player gets an extra $\pi_a/(|\mathcal{A}|+1)$ utility at the end of stage two than what the player received from following strategy s. Moreover, since the players in $\mathcal{C} - \mathcal{A}$ only have control over the non-negative side payments flowing from $\mathcal{C} - \mathcal{A}$ to $\mathcal{A} \cup \mu$, we have shown that the players in $\mathcal{A} \cup \mu$ can guarantee themselves payoffs greater than the payoffs that they received from following s. Thus, $\pi \in D(\mathcal{A} \cup \mu)$ in SP. □

Lemma 3.3.3. *If π is in the core of* SP *or* NOSP*, then $\pi_a \geq 0$ for all $a \in \mathcal{P}$.*

Proof. We prove the statement by contradiction. Suppose that $\pi_a < 0$ for some player a.

If a is a supplier, a can guarantee at least 0 utility with strategy $(\tau(a), 0) \in \mathcal{S}_a$. If a is the buyer, a can guarantee 0 utility with strategy $(\emptyset, 0)$. Thus, $\pi \in D(\{a\})$ in both SP and NOSP. Thus, π is not in the core of either game. □

Lemma 3.3.4. *Let π be a payoff vector, and let s be a strategy vector in $\mathcal{S}_{\mathcal{P}}$. If the players in $\mathcal{P}$ can guarantee themselves payoffs of at least π by playing s, but s does not realize π, then π is not in the core of either* SP *or* NOSP*.*

Proof. Let $s_\mu = (\mathcal{A}, p_\mu)$.

We use a proof by contradiction. Assume π is in the core of either SP or NOSP. By Lemma 3.3.3, we know that $\pi_a \geq 0$ for all $a \in \mathcal{P}$. By Lemma 3.3.2, we know that $\pi_a = 0$ for all $a \in \mathcal{C} - \mathcal{A}$.

First, we derive a contradiction with the assumption that π is in the core of SP.

Consider the two stage distribution of wealth interpretation of SP. Since s guarantees payoffs of at least π but s does not realize π, we know that $\pi_a \leq u_a(s)$ for all $a \in \mathcal{P}$ and there is some $a \in \mathcal{P}$ such that $\pi_a < u_a(s)$. Thus by following s, the total wealth held by $\mathcal{A} \cup \mu$ in SP at the end of stage one is greater than $\sum_{a \in \mathcal{P}} \pi_a$. In turn, we have $\sum_{a \in \mathcal{P}} \pi_a \geq \sum_{a \in \mathcal{A} \cup \mu} \pi_a$. Let λ be the difference between the total wealth held by $\mathcal{A} \cup \mu$ at the end of stage one and $\sum_{a \in \mathcal{A} \cup \mu} \pi_a$.

Instead of following s, the players in $\mathcal{A} \cup \mu$ can set to zero all side payments from $\mathcal{A} \cup \mu$ to $\mathcal{C} - \mathcal{A}$. The players in $\mathcal{A} \cup \mu$ can use side payments amongst themselves so that each player gets an extra $\lambda/(|\mathcal{A}| + 1)$ utility at the end of stage two than what the player received in π. Moreover, since the players in $\mathcal{C} - \mathcal{A}$ only have control over the non-negative side payments flowing from $\mathcal{C} - \mathcal{A}$ to $\mathcal{A} \cup \mu$, we have shown that the players in $\mathcal{A} \cup \mu$ can guarantee themselves payoffs greater than the payoffs that they received in π. Thus, we have constructed a new strategy $t \in \mathcal{S}_{\mathcal{A} \cup \mu}$ in SP for the players in $\mathcal{A} \cup \mu$ which guarantees payoffs greater than π for each player in $\mathcal{A} \cup \mu$. Thus, $\pi \in D(\mathcal{A} \cup \mu)$ in SP. This contradicts the assumption that π is in the core of SP. Thus, π must be in the core of NOSP.

We now derive a contradiction with the assumption that π is in the core of NOSP. By Lemma 3.3.1 and the fact that t guarantees payoffs greater than π for each player in $\mathcal{A} \cup \mu$, we also have $\pi \in D(\mathcal{A} \cup \mu)$ in NOSP. This contradicts the assumption that π is in the core of NOSP. □

3.3.2 Core Equivalence between SP and NOSP

In this section, we prove that the core of NOSP is the same as the core of SP. All the lemmas in this section are used solely to prove the main result of the section, Theorem 3.3.11.

Lemma 3.3.5. *Let π be a payoff vector. If $\pi \in D(\mathcal{A})$ in* NOSP*, then $\pi \in D(\mathcal{A})$ in* SP*.*

Proof. The players in $\mathcal{A}$ can follow exactly the same strategy in SP as they would in NOSP to guarantee payoffs greater than the payoffs in π; they simply fix all their side payments to zero. □

Lemma 3.3.6. *Let π be a payoff vector. If $\pi \in V(\mathcal{P})$ in* SP *and π is in the core of* SP*, then $\pi \in V(\mathcal{P})$ in* NOSP*.*

Proof. By Lemma 3.3.4, π is realized by some strategy vector $s \in \mathcal{S}_{\mathcal{P}}$ in SP. Let $s_\mu = (\mathcal{A}, p_\mu)$.

Consider the two stage distribution of wealth interpretation of SP. By Lemma 3.3.2, we have $\pi_a = 0$ for all $a \in \mathcal{C} - \mathcal{A}$. Thus, at the end of stage two of SP, there is no utility in $\mathcal{C} - \mathcal{A}$. Thus, the players can achieve the same utility distribution by setting to zero all side payments except side payments from $\mathcal{A} \cup \mu$ to $\mathcal{A} \cup \mu$. Thus π is realized by a strategy vector with all zero side payments except the side payments from $\mathcal{A} \cup \mu$ to $\mathcal{A} \cup \mu$.

By Lemma 3.3.1, there is a strategy vector with all side payments fixed to zero which guarantees payoffs of at least π for all the players in $\mathcal{P}$. Thus $\pi \in V(\mathcal{P})$. □

Lemma 3.3.7. *If payoff vector π is in the core of* SP*, then π is in the core of* NOSP*.*

Proof. The statement follows from Lemma 3.3.6 and the contrapositive of Lemma 3.3.5. □

Lemma 3.3.8. *Let π be a payoff vector. If $\pi \in V(\mathcal{P})$ in* NOSP*, then $\pi \in V(\mathcal{P})$ in* SP*.*

Proof. By the definition of $V(\mathcal{P})$, the players in $\mathcal{P}$ can guarantee themselves payoffs of at least π by playing some strategy $s \in \mathcal{S}_{\mathcal{P}}$ in NOSP. The players in $\mathcal{P}$ can follow exactly the same strategy in SP to guarantee payoffs of at least π. □

Lemma 3.3.9. *If payoff vector π is in the core of* NOSP*, then for all $\mathcal{A} \subseteq \mathcal{P}$ we have $\pi \notin D(\mathcal{A})$ in* SP*.*

Proof. We use a proof by contradiction. Suppose that $\pi \in D(\mathcal{A})$ in SP for some $\mathcal{A} \subseteq \mathcal{P}$. Thus, there is a strategy vector $s \in \mathcal{S}_{\mathcal{A}}$ in SP which guarantees each player in $\mathcal{A}$ a greater payoff than the payoff given in π.

Since π is in the core of NOSP, by Lemma 3.3.3 we know $\pi_a \geq 0$ for all $a \in \mathcal{A}$.

We split the proof into two cases. In the first case, suppose that $\mu \notin \mathcal{A}$. It is impossible for s to guarantee a payoff greater than 0 for any player in $\mathcal{A}$ since the buyer can always play $\emptyset$. Thus, we get a contradiction with the assumption that $\pi \in D(\mathcal{A})$ in SP.

For the second case, suppose $\mu \in \mathcal{A}$. Let $s_\mu = (\mathcal{B}, p_\mu)$. There can not be some supplier in $\mathcal{B}$ but not in $\mathcal{A}$ since that supplier can always play the strategy $(\lambda, 0)$ where $\lambda > M$ to give the buyer a negative payoff. Since $\pi_\mu \geq 0$, the existence of a supplier in $\mathcal{B} - \mathcal{A}$ contradicts the assumption that $\pi \in D(\mathcal{A})$ in SP.

Thus, we have $\mathcal{B} \subseteq \mathcal{A} - \{\mu\}$.

If there is some supplier a in $\mathcal{A}$ but not in $\mathcal{B}$, since $\pi_a \geq 0$, we know that s must guarantee a payoff greater than 0 for a. Let ν be the payoff vector realized when the players in $\mathcal{A}$ follow s and each of the players in $\mathcal{P} - \mathcal{A}$ follow the strategy $(0, 0)$. Thus, we have $\nu_a > 0$ and $\nu_b > \pi_b$ for all $b \in \mathcal{A}$. By Lemma 3.3.2, we have $\nu \in D(\mathcal{B} \cup \mu)$ in SP. Since $\nu \in D(\mathcal{B} \cup \mu)$ in SP, $\nu_b > \pi_b$ for all $b \in \mathcal{A}$, and $\mathcal{B} \subseteq \mathcal{A} - \{\mu\}$, we have $\pi \in D(\mathcal{B} \cup \mu)$ in SP.

Thus, we have $s_\mu = (\mathcal{B}, p_\mu)$ and $\pi \in D(\mathcal{B} \cup \mu)$ in SP. By Lemma 3.3.1, we also have $\pi \in D(\mathcal{B} \cup \mu)$ in NOSP, which contradicts the fact that π is in the core of NOSP. □

Lemma 3.3.10. *If payoff vector π is in the core of* NOSP*, then π is in the core of* SP*.*

Proof. The statement follows from Lemmas 3.3.8 and 3.3.9. □

Theorem 3.3.11. *The core of* NOSP *is equal to the core of* SP*.*

Proof. Follows from Lemmas 3.3.7 and 3.3.10. □

3.3.3 The Core of NOSP

In this section, we show a characterization of the core of NOSP with Theorem 3.3.20. By Theorem 3.3.11, the same characterization is true of the core of SP. Throughout the section, we refer to Equations (3.1), (3.2), and (3.3), whose definition comes from Theorem 3.3.20. The lemmas in this section are solely used to prove the main result of the section, Theorem 3.3.20. At the end of the section, as a corollary to the theorem, we show Lemma 3.3.21, stating that the core does not change depending on the decomposition chosen in the transformation from a combinatorial minimization problem to a buyer-supplier game.

Lemma 3.3.12. *If a payoff vector π is realized by some strategy vector $s \in \mathcal{S}_\mathcal{P}$ where $s_\mu = (\mathcal{A}, 0)$, then $\pi_\mu = M - \text{Eval}(\mathcal{A}) - \sum_{a \in \mathcal{C}} \pi_a$.*

Proof. For any $a \in \mathcal{C}$ let $s_a = (\beta(a), p_a)$. Since π is realized by s and all side payments are zero, we have $\pi_a = \beta(a) - \tau(a)$ for all $a \in \mathcal{A}$. We also have that $\pi_\mu = M - \left[\text{Bcost}(\mathcal{A}) + \sum_{a\in\mathcal{A}} \beta(a)\right]$. Substituting for $\beta(a)$, we have $\pi_\mu = M - \left[\text{Bcost}(\mathcal{A}) + \sum_{a\in\mathcal{A}} \tau(a) + \sum_{a\in\mathcal{A}} \pi_a\right]$. By the definition of Eval, we have $\pi_\mu = M - \text{Eval}(\mathcal{A}) - \sum_{a\in\mathcal{A}} \pi_a$.

Since all side payments are fixed to zero, we have $\pi_a = 0$ for all $a \in \mathcal{C} - \mathcal{A}$. Thus we can write $\pi_\mu = M - \text{Eval}(\mathcal{A}) - \sum_{a\in\mathcal{C}} \pi_a$. □

Lemma 3.3.13. *If a payoff vector π is in the core of* NOSP*, then $\pi_\mu = M - \text{MinEval}(\mathcal{C}) - \sum_{a\in\mathcal{C}} \pi_a$.*

Proof. Let $\mathcal{F} \subseteq \mathcal{C}$ be such that $\text{Eval}(\mathcal{F}) = \text{MinEval}(\mathcal{C})$. We first show that any ν realized by a strategy vector s with $s_\mu = (\mathcal{A}, 0)$ and $\text{Eval}(\mathcal{A}) \neq \text{Eval}(\mathcal{F})$ is not in the core.

Let $\lambda = (\text{Eval}(\mathcal{A}) - \text{Eval}(\mathcal{F}))/(|\mathcal{F}| + 1)$. Since $\text{Eval}(\mathcal{A}) \neq \text{Eval}(\mathcal{F})$ and by the definition of $\mathcal{F}$, we have $\lambda > 0$. Construct a strategy vector $t \in \mathcal{S}_{\mathcal{F}\cup\mu}$ where

$$t_\mu = (\mathcal{F}, 0)$$
$$t_a = (\nu_a + \tau(a) + \lambda, 0) \qquad \text{for all } a \in \mathcal{F}.$$

Since side payments are fixed to zero, the suppliers in $\mathcal{C} - \mathcal{F}$ have no strategies which can affect the payoffs of the players in $\mathcal{F} \cup \mu$ given that the players in $\mathcal{F} \cup \mu$ follow t. Let $u \in \mathcal{S}_\mathcal{P}$ be any strategy vector with projection onto $\mathcal{F} \cup \mu$ equal to t.

Straight forward calculations with the game's utility functions show that $u_a(u) - \nu_a = u_a(u) - u_a(s) = \lambda$ for each supplier $a \in \mathcal{F}$.

Consider

$$\begin{aligned}
u_\mu(u) - \nu_\mu &= u_\mu(u) - u_\mu(s) \\
&= [M - \text{Bcost}(\mathcal{F}) - \sum_{a\in\mathcal{F}}(\nu_a + \tau(a) + \lambda)] \\
&\quad - [M - \text{Bcost}(\mathcal{A}) - \sum_{a\in\mathcal{A}}(\nu_a + \tau(a))] \\
&= [-\text{Eval}(\mathcal{F}) - \sum_{a\in\mathcal{F}}(\nu_a + \lambda)] - [-\text{Eval}(\mathcal{A}) - \sum_{a\in\mathcal{A}}\nu_a] \\
&= \text{Eval}(\mathcal{A}) - \text{Eval}(\mathcal{F}) - \sum_{a\in\mathcal{F}}\lambda + [\sum_{a\in\mathcal{A}}\nu_a - \sum_{a\in\mathcal{F}}\nu_a].
\end{aligned}$$

By the utility functions of NOSP and the definition of $\mathcal{A}$ and ν, we have $\nu_a = 0$ for all $a \in \mathcal{C} - \mathcal{A}$. Thus, the bracketed quantity in the above expression is at least zero. Thus, we have

$$u_\mu(u) - \nu_\mu \geq \text{Eval}(\mathcal{A}) - \text{Eval}(\mathcal{F}) - \sum_{a\in\mathcal{F}}\lambda = \lambda$$

where the equality comes from the definition of λ.

Thus, we have $\nu \in D(\mathcal{F} \cup \mu)$.

We have shown that any vector in the core is realized by a strategy vector s with $s_\mu = (\mathcal{A}, 0)$ where $\text{Eval}(\mathcal{A}) = \text{Eval}(\mathcal{F})$. The lemma statement follows from Lemma 3.3.12 and the definition of $\mathcal{F}$. □

Lemma 3.3.14. *If payoff vector π is in the core of* NOSP*, then*

$$\sum_{a\in\mathcal{A}}\pi_a \leq \text{MinEval}(\mathcal{C} - \mathcal{A}) - \text{MinEval}(\mathcal{C})$$

for all $\mathcal{A} \subseteq \mathcal{C}$.

Proof. We use a proof by contradiction. Assume π is in the core and $\sum_{a\in\mathcal{A}}\pi_a > \text{MinEval}(\mathcal{C} - \mathcal{A}) - \text{MinEval}(\mathcal{C})$ for some $\mathcal{A} \subseteq \mathcal{C}$. We show that $\pi \in D(\mathcal{F} \cup \mu)$ where $\mathcal{F} \subseteq \mathcal{C} - \mathcal{A}$ is such that $\text{Eval}(\mathcal{F}) = \text{MinEval}(\mathcal{C} - \mathcal{A})$.

Since π is in the core, by Lemma 3.3.4 it is realized by some strategy vector $s \in \mathcal{S}_\mathcal{P}$

Let $\lambda = (\sum_{a\in\mathcal{A}} \pi_a - \text{Eval}(\mathcal{F}) + \text{MinEval}(\mathcal{C}))/(|\mathcal{F}| + 1)$. Since $\sum_{a\in\mathcal{A}} \pi_a > \text{MinEval}(\mathcal{C} - \mathcal{A}) - \text{MinEval}(\mathcal{C})$, we have $\lambda > 0$. Construct a strategy vector $t \in \mathcal{S}_{\mathcal{F}\cup\mu}$ where

$$\begin{aligned} t_\mu &= (\mathcal{F}, 0) \\ t_a &= (\pi_a + \tau(a) + \lambda, 0) \qquad\qquad \text{for all } a \in \mathcal{F}. \end{aligned}$$

Since side payments are fixed to zero, the suppliers in $\mathcal{C} - \mathcal{F}$ have no strategies which can affect the payoffs of the players in $\mathcal{F} \cup \mu$ given that the players in $\mathcal{F} \cup \mu$ follow t. Let $u \in \mathcal{S}_\mathcal{P}$ be any strategy vector with projection onto $\mathcal{F} \cup \mu$ equal to t.

Straight forward calculations with the game's utility functions show that $u_a(u) - \pi_a = u_a(u) - u_a(s) = \lambda$ for each supplier $a \in \mathcal{F}$.

Let $s_\mu = (\mathcal{B}, 0)$ and consider

$$\begin{aligned} u_\mu(u) - \pi_\mu &= u_\mu(u) - u_\mu(s) \\ &= [M - \text{Bcost}(\mathcal{F}) - \sum_{a\in\mathcal{F}} (\pi_a + \tau(a) + \lambda)] \\ &\quad - [M - \text{Bcost}(\mathcal{B}) - \sum_{a\in\mathcal{B}} (\pi_a + \tau(a))] \\ &= [-\text{Eval}(\mathcal{F}) - \sum_{a\in\mathcal{F}} (\pi_a + \lambda)] - [\text{Eval}(\mathcal{B}) - \sum_{a\in\mathcal{B}} \pi_a] \\ &= \text{Eval}(\mathcal{B}) - \text{Eval}(\mathcal{F}) - \sum_{a\in\mathcal{F}} \lambda + [\sum_{a\in\mathcal{B}} \pi_a - \sum_{a\in\mathcal{F}} \pi_a] \end{aligned}$$

By the utility functions of NOSP and the definitions of $\mathcal{B}$ and π, we have $\pi_a = 0$ for all $a \in \mathcal{C} - \mathcal{B}$. Thus, $\sum_{a\in\mathcal{B}} \pi_a = \sum_{a\in\mathcal{C}} \pi_a$. Thus, we can write

$$\begin{aligned} u_\mu(u) - u_\mu(s) &= \text{Eval}(\mathcal{B}) - \text{Eval}(\mathcal{F}) - \sum_{a\in\mathcal{F}} \lambda + [\sum_{a\in\mathcal{C}} \pi_a - \sum_{a\in\mathcal{F}} \pi_a] \\ &= \text{Eval}(\mathcal{B}) - \text{Eval}(\mathcal{F}) - \sum_{a\in\mathcal{F}} \lambda + \sum_{a\in\mathcal{A}} \pi_a + [\sum_{a\in\mathcal{C}-\mathcal{A}} \pi_a - \sum_{a\in\mathcal{F}} \pi_a] \end{aligned}$$

Since $\mathcal{F} \subseteq \mathcal{C} - \mathcal{A}$, we know that the bracketed quantity in the above expression is at least zero. Thus, we have

$$u_\mu(u) - u_\mu(s) \geq \text{Eval}(\mathcal{B}) - \text{Eval}(\mathcal{F}) + \sum_{a\in\mathcal{A}} \pi_a - \sum_{a\in\mathcal{F}} \lambda = \lambda$$

where the equality comes from the definition on λ.

Thus, we have $\pi \in D(\mathcal{F} \cup \mu)$, which contradicts the fact that π is in the core of NOSP. □

Lemma 3.3.15. *Payoff vectors in the core of* NOSP *satisfy Equations (3.1), (3.2), and (3.3).*

Proof. The statement follows from Lemmas 3.3.3, 3.3.14, and 3.3.13. □

Lemma 3.3.16. *If payoff vector π satisfies Equations (3.1), (3.2), and (3.3) then $\pi \notin D(\mathcal{A})$ for $\mathcal{A} \subseteq \mathcal{P}$ such that $\mu \notin \mathcal{A}$.*

Proof. We use a proof by contradiction. Suppose $\pi \in D(\mathcal{A})$. In other words, the players in $\mathcal{A}$ can guarantee payoffs greater than the payoffs given in π. But, we know that $\pi_a \geq 0$ for all $a \in \mathcal{A}$ and the players in $\mathcal{A}$ can only guarantee 0 payoffs because the buyer can always play $(\emptyset, 0)$. Thus, we have a contradiction with $\pi \in D(\mathcal{A})$. □

Lemma 3.3.17. *If payoff vector π satisfies Equations (3.1), (3.2), and (3.3) then $\pi \notin D(\mathcal{A})$ for $\mathcal{A} \subseteq \mathcal{P}$ such that $\mu \in \mathcal{A}$.*

Proof. We use a proof by contradiction. Suppose $\pi \in D(\mathcal{A})$ for $\mathcal{A} \subseteq \mathcal{P}$ such that $\mu \in \mathcal{A}$. Thus, the players in $\mathcal{A}$ can follow a strategy $s \in \mathcal{S}_{\mathcal{A}}$ that guarantees payoffs greater than the payoffs they are given in π.

Let $t \in \mathcal{S}_{\mathcal{P}}$ be any strategy vector with projection onto $\mathcal{A}$ equal to s. Let $t_\mu = (\mathcal{B}, 0)$. Let the payoff vector realized by t be ν.

Since s guarantees payoffs greater than π for the players in $\mathcal{A}$, all extentions of s to a strategy vector in $\mathcal{S}_{\mathcal{P}}$ must realize payoff vectors π' with $\pi'_a > \pi_a \geq 0$ for $a \in \mathcal{A}$. The existance of some $a \in \mathcal{B} - \mathcal{A}$ contradicts this statement, since the supplier a sets their bid arbitrarily high in some extensions of s, resulting in a negative utility for the buyer. Thus, we have that there is no such a and $\mathcal{B} \subseteq \mathcal{A}$.

Since π satisfies Equation (3.3) we have $\pi_\mu = M - \text{MinEval}(\mathcal{C}) - \sum_{a \in \mathcal{C}} \pi_a$. By Lemma 3.3.12, we have $\nu_\mu = M - \text{Eval}(\mathcal{B}) - \sum_{a \in \mathcal{C}} \nu_a$.

Since following s guarantees a payoff greater than the payoff given in π for

every player in $\mathcal{A}$, we have $\pi_\mu < \nu_\mu$. Thus, we have

$$\begin{aligned}
0 &< \nu_\mu - \pi_\mu \\
&= M - \text{Eval}(\mathcal{B}) - \sum_{a \in \mathcal{C}} \nu_a - [M - \text{MinEval}(\mathcal{C}) - \sum_{a \in \mathcal{C}} \pi_a] \\
&= \text{MinEval}(\mathcal{C}) - \text{Eval}(\mathcal{B}) + \sum_{a \in \mathcal{C}} \pi_a - \sum_{a \in \mathcal{C}} \nu_a
\end{aligned}$$

Let $\mathcal{F} \subseteq \mathcal{C}$ be such that $\text{Eval}(\mathcal{F}) = \text{MinEval}(\mathcal{C})$. From Equation (3.2) with the singleton sets, we have that $\pi_a = 0$ for all $a \notin \mathcal{F}$. From the definition of ν, we have that $\nu_a = 0$ for all $a \notin \mathcal{B}$. Let $\mathcal{U} = \mathcal{A} - \{\mu\}$. Thus, we have

$$\begin{aligned}
0 &< \text{MinEval}(\mathcal{C}) - \text{Eval}(\mathcal{B}) + \sum_{a \in \mathcal{F}} \pi_a - \sum_{a \in \mathcal{B}} \nu_a \\
&= \text{MinEval}(\mathcal{C}) - \text{Eval}(\mathcal{B}) + \sum_{a \in \mathcal{F} - \mathcal{U}} \pi_a + (\sum_{a \in \mathcal{F} \cap \mathcal{U}} \pi_a - \sum_{a \in \mathcal{F} \cap \mathcal{B}} \nu_a) - \sum_{a \in \mathcal{B} - \mathcal{F}} \nu_a
\end{aligned}$$

By the definition of ν and the utility functions in NOSP, we have that $\nu_a = 0$ for all $a \in \mathcal{C} - \mathcal{B}$. We also have $\nu_a > \pi_a \geq 0$ for all $a \in \mathcal{B}$. Thus, we can drop the last term in the above expression to get

$$0 < \text{MinEval}(\mathcal{C}) - \text{Eval}(\mathcal{B}) + \sum_{a \in \mathcal{F} - \mathcal{U}} \pi_a + (\sum_{a \in \mathcal{F} \cap \mathcal{U}} \pi_a - \sum_{a \in \mathcal{F} \cap \mathcal{U}} \nu_a)$$

By the definitions of ν and $\mathcal{U}$, we also have that $\nu_a > \pi_a$ for all $a \in \mathcal{U}$. Thus, we can drop the parenthesized term in the above expression to get

$$\begin{aligned}
0 &< \text{MinEval}(\mathcal{C}) - \text{Eval}(\mathcal{B}) + \sum_{a \in \mathcal{F} - \mathcal{U}} \pi_a \\
\text{Eval}(\mathcal{B}) - \text{MinEval}(\mathcal{C}) &< \sum_{a \in \mathcal{F} - \mathcal{U}} \pi_a
\end{aligned}$$

Let $\mathcal{K} = \mathcal{F} - \mathcal{U}$. Since $\mathcal{B} \subseteq \mathcal{A}$ and $\mathcal{B} \subseteq \mathcal{C}$, we have $\mathcal{B} \subseteq \mathcal{U}$. Thus, we have $\mathcal{B} \subseteq \mathcal{C} - \mathcal{K}$. By the definition of MinEval, we have $\text{MinEval}(\mathcal{C} - \mathcal{K}) \leq \text{Eval}(\mathcal{B})$. Thus, we have

$$\text{MinEval}(\mathcal{C} - \mathcal{K}) - \text{MinEval}(\mathcal{C}) \leq \text{Eval}(\mathcal{B}) - \text{MinEval}(\mathcal{C}) < \sum_{a \in \mathcal{K}} \pi_a.$$

This statement contradicts the fact that π satisfies Equation (3.2) for $\mathcal{K}$. □

Lemma 3.3.18. *If payoff vector π satisfies Equations (3.1), (3.2), and (3.3), then $\pi \in V(\mathcal{P})$ in* NOSP*.*

Proof. Let $\mathcal{F} \subseteq \mathcal{C}$ be such that $\text{Eval}(\mathcal{F}) = \text{MinEval}(\mathcal{C})$. Define $s \in \mathcal{S}_{\mathcal{P}}$ such that

$$\begin{aligned} s_\mu &= (\mathcal{F}, 0) \\ s_a &= (\pi_a + \tau(a), 0) && \text{for all } a \in \mathcal{F} \\ s_a &= (0, 0) && \text{for all } a \in \mathcal{C} - \mathcal{F} \end{aligned}$$

Straight forward calculations with the game's utility functions show that $u_a(s) = \pi_a$ for each supplier $a \in \mathcal{F}$.

Consider

$$\begin{aligned} u_\mu(s) &= M - [\text{Bcost}(\mathcal{F}) + \sum_{a \in \mathcal{F}} (\pi_a + \tau(a))] \\ &= M - [\text{Eval}(\mathcal{F}) + \sum_{a \in \mathcal{F}} \pi_a] \\ &= M - \text{Eval}(\mathcal{F}) - \sum_{a \in \mathcal{F}} \pi_a \end{aligned}$$

Since π satisfies Equation (3.2), we have $\pi_a = 0$ for all $a \in \mathcal{C} - \mathcal{F}$. Thus, we have

$$\begin{aligned} u_\mu(s) &= M - \text{Eval}(\mathcal{F}) - \sum_{a \in \mathcal{C}} \pi_a \\ &= M - \text{MinEval}(\mathcal{C}) - \sum_{a \in \mathcal{C}} \pi_a \\ &= \pi_\mu \end{aligned}$$

where the second equality comes from the definition of $\mathcal{F}$ and the last equality comes from the fact that π satisfies Equation (3.3).

Finally, we have $u_a(s) = \pi_a$ for each supplier $a \in \mathcal{C} - \mathcal{F}$, since $\pi_a = 0$ for such a.

Thus, s realizes π and $\pi \in V(\mathcal{P})$. □

Lemma 3.3.19. *If payoff vector π satisfies Equations (3.1), (3.2), and (3.3) then π is in the core of* NOSP*.*

Proof. The statement follows from Lemmas 3.3.16, 3.3.17, and 3.3.18. □

In the following theorem, the parameter τ is made explicit, though its value is clear from the definition of the buyer-supplier game, NOSP.

Theorem 3.3.20. *A payoff vector π is in the core of* NOSP *if and only if it satisfies*

$$\pi_a \geq 0 \qquad \text{for all } a \in \mathcal{P} \tag{3.1}$$

$$\sum_{a \in \mathcal{A}} \pi_a \leq \text{MinEval}(\tau, \mathcal{C} - \mathcal{A}) - \text{MinEval}(\tau, \mathcal{C}) \qquad \text{for all } \mathcal{A} \subseteq \mathcal{C} \tag{3.2}$$

$$\pi_\mu = M - \text{MinEval}(\tau, \mathcal{C}) - \sum_{a \in \mathcal{C}} \pi_a \tag{3.3}$$

Proof. The statement follows from Lemmas 3.3.15 and 3.3.19. □

Lemma 3.3.21. *Let* $\text{Bcost}^*(\mathcal{A}) = \sum_{a \in \mathcal{A}} \tau(a) + \text{Bcost}(\mathcal{A})$. *The core of the buyer-supplier games defined by* $(\mathcal{C}, \tau, \text{Bcost})$ *and* $(\mathcal{C}, 0, \text{Bcost}^*)$ *is the same.*

Proof. We have $\text{Eval}(\tau, \text{Bcost}, \mathcal{B}) = \text{Eval}(0, \text{Bcost}^*, \mathcal{B})$ for all $\mathcal{B} \subseteq \mathcal{C}$ by the definition of Eval and Bcost^*. Thus, we have $\text{MinEval}(\tau, \text{Bcost}, \mathcal{A}) = \text{MinEval}(0, \text{Bcost}^*, \mathcal{A})$ for all $\mathcal{A} \subseteq \mathcal{C}$. The result follows from Theorem 3.3.20. □

3.4 Polynomial Time Optimization Over the Core Vectors

We define the separation problem on a set of linear inequalities $\mathcal{A}$ as follows. Given a vector π, if π satisfies all of the inequalities in $\mathcal{A}$, then do nothing; otherwise, output a violated inequality $a \in \mathcal{A}$. It is well known that the separation problem is polynomial time equivalent to linear function optimization over the same set of inequalities [41, p. 161].

Let $(\mathcal{C}, \tau, \text{Bcost})$ define a buyer-supplier game. In this section, to simplify the notation, we will omit the parameter Bcost from Eval and MinEval since it is fixed by the buyer-supplier game.

In this section, we will analyze an algorithm to solve the separation problem for the exponentially sized set of inequalities given in Equations (3.1), (3.2), and (3.3). We now give the algorithm, which we call the separation algorithm. Given the payoff vector π as input,

1 Iterate over Equations (3.1) and (3.3) to check that they hold. If some equation does not hold, output that equation and halt.

2 Compute $\mathcal{F} \subseteq \mathcal{C}$ such that $\text{Eval}(\tau, \mathcal{F}) = \text{MinEval}(\tau, \mathcal{C})$. If there is some $a \in \mathcal{C} - \mathcal{F}$ with $\pi_a > 0$, output the inequality from Equation (3.2) corresponding to $\{a\}$ and halt.

3 Define $\hat{\tau}(a) = \tau(a) + \pi_a$ for $a \in \mathcal{C}$. Now, compute $\hat{\mathcal{F}} \subseteq \mathcal{C}$ such that $\text{Eval}(\hat{\tau}, \hat{\mathcal{F}}) = \text{MinEval}(\hat{\tau}, \mathcal{C})$. If $\text{Eval}(\hat{\tau}, \hat{\mathcal{F}}) < \text{Eval}(\hat{\tau}, \mathcal{F})$, output the inequality from Equation (3.2) corresponding to $\mathcal{F} - \hat{\mathcal{F}}$. Otherwise, halt.

Theorem 3.4.1. *If given an input $\hat{\tau} : \mathcal{C} \to \Re_+$ it is possible to compute both $\text{Eval}(\hat{\tau}, \mathcal{A})$, for any $\mathcal{A} \subseteq \mathcal{C}$, and $\mathcal{F} \subseteq \mathcal{C}$ such that $\text{Eval}(\hat{\tau}, \mathcal{F}) = \text{MinEval}(\hat{\tau}, \mathcal{C})$ in polynomial time, then the separation problem for Equations (3.1), (3.2), and (3.3) is solvable in polynomial time. By the equivalence of separation and optimization, optimizing any linear function of π over Equations (3.1), (3.2), and (3.3) is also possible in polynomial time.*

Proof. It is clear that given the theorem's assumptions, the separation algorithm runs in polynomial time. The statement follows from Lemmas 3.4.2 and 3.4.3. □

Lemma 3.4.2. *If the separation algorithm returns an inequality on input π, then π violates the returned inequality.*

Proof. If the algorithm returns an inequality in step 1, then the inequality is violated since the algorithm performed a direct check.

If the algorithm returns an inequality in step 2, then the inequality is violated since $\pi_a > 0$, but $\text{MinEval}(\tau, \mathcal{C} - \{a\}) = \text{MinEval}(\tau, \mathcal{C}) = \text{Eval}(\tau, \mathcal{F})$.

Suppose the algorithm returns an inequality in step 3. Thus, $\text{Eval}(\hat{\tau}, \hat{\mathcal{F}}) < \text{Eval}(\hat{\tau}, \mathcal{F})$. Applying the definitions of Eval and $\hat{\tau}$, we have $\sum_{a \in \hat{\mathcal{F}}} \pi_a + \text{Eval}(\tau, \hat{\mathcal{F}}) < \sum_{a \in \mathcal{F}} \pi_a + \text{Eval}(\tau, \mathcal{F})$.

Since the algorithm reaches step 3, we know that $\pi_a = 0$ for all $a \in \mathcal{C} - \mathcal{F}$. Thus, we have $\sum_{a \in \hat{\mathcal{F}} \cap \mathcal{F}} \pi_a + \text{Eval}(\tau, \hat{\mathcal{F}}) < \sum_{a \in \mathcal{F}} \pi_a + \text{Eval}(\tau, \mathcal{F})$, which in turn gives $\text{Eval}(\tau, \hat{\mathcal{F}}) - \text{Eval}(\tau, \mathcal{F}) < \sum_{a \in \mathcal{F} - \hat{\mathcal{F}}} \pi_a$.

Let $\mathcal{A} = \mathcal{F} - \hat{\mathcal{F}}$. From the algorithm, we know that the set $\mathcal{F}$ satisfies $\text{Eval}(\tau, \mathcal{F}) = \text{MinEval}(\tau, \mathcal{C})$. Since $\hat{\mathcal{F}} \subseteq \mathcal{C} - \mathcal{A}$, the definition of MinEval implies that $\text{MinEval}(\tau, \mathcal{C} - \mathcal{A}) \leq \text{Eval}(\tau, \hat{\mathcal{F}})$. Thus, we have $\text{MinEval}(\tau, \mathcal{C} - \mathcal{A}) -$

$\text{MinEval}(\tau, \mathcal{C}) \leq \text{Eval}(\tau, \hat{\mathcal{F}}) - \text{Eval}(\tau, \mathcal{F}) < \sum_{a \in \mathcal{A}} \pi_a$, which shows that the inequality output by the algorithm is violated. □

Lemma 3.4.3. *If π violates some inequality in Equations (3.1), (3.2), and (3.3), then the separation algorithm run on input π returns an inequality.*

Proof. If the violation is in Equations (3.1) or (3.3), the violated inequality will be output by the direct check in step 1. If some inequality is output by step 2, we are done. Otherwise, since steps 1 and 2 output no inequality, we know that $\pi_a = 0$ for all $a \in \mathcal{C} - \mathcal{F}$, where $\mathcal{F}$ is as computed in the algorithm.

Now, suppose the inequality from Equation (3.2) for set $\mathcal{A} \subseteq \mathcal{C}$ is violated. In other words, we have, $\sum_{a \in \mathcal{A}} \pi_a > \text{MinEval}(\tau, \mathcal{C} - \mathcal{A}) - \text{MinEval}(\tau, \mathcal{C})$. Let $\mathcal{B}$ be such that $\text{Eval}(\tau, \mathcal{B}) = \text{MinEval}(\tau, \mathcal{C} - \mathcal{A})$.

Thus, we have $\sum_{a \in \mathcal{A}} \pi_a > \text{MinEval}(\tau, \mathcal{C} - \mathcal{A}) - \text{MinEval}(\tau, \mathcal{C}) = \text{Eval}(\tau, \mathcal{B}) - \text{Eval}(\tau, \mathcal{F})$.

Since $\pi_a = 0$ for all $a \in \mathcal{C} - \mathcal{F}$, we have $\text{Eval}(\tau, \mathcal{F}) + \sum_{a \in \mathcal{F} \cap \mathcal{A}} \pi_a > \text{Eval}(\tau, \mathcal{B})$.

Adding $\sum_{a \in \mathcal{F} - \mathcal{A}} \pi_a$ to both sides of the above inequality and substituting the definition of Eval, we have $\text{Bcost}(\mathcal{F}) + \sum_{a \in \mathcal{F}} \tau(a) + \sum_{a \in \mathcal{F}} \pi_a > \text{Bcost}(\mathcal{B}) + \sum_{a \in \mathcal{B}} \tau(a) + \sum_{a \in \mathcal{F} - \mathcal{A}} \pi_a$.

Since $\pi_a = 0$ for all $a \in \mathcal{C} - \mathcal{F}$ and $\mathcal{B} \subseteq \mathcal{C} - \mathcal{A}$, we can alter the right hand side of the above inequality to get $\text{Bcost}(\mathcal{F}) + \sum_{a \in \mathcal{F}} \tau(a) + \sum_{a \in \mathcal{F}} \pi_a > \text{Bcost}(\mathcal{B}) + \sum_{a \in \mathcal{B}} \tau(a) + \sum_{a \in \mathcal{B}} \pi_a + \sum_{a \in \mathcal{F} - \mathcal{A} - \mathcal{B}} \pi_a$.

By applying the definition of $\hat{\tau}$ and Eval, we have $\text{Eval}(\hat{\tau}, \mathcal{F}) > \text{Eval}(\hat{\tau}, \mathcal{B}) + \sum_{a \in \mathcal{F} - \mathcal{A} - \mathcal{B}} \pi_a$. We know that $\pi_a \geq 0$ for all $a \in \mathcal{P}$ since the algorithm does not output anything in step 1. Thus, $\text{Eval}(\hat{\tau}, \mathcal{F}) > \text{Eval}(\hat{\tau}, \mathcal{B}) \geq \text{MinEval}(\hat{\tau}, \mathcal{C}) = \text{Eval}(\hat{\tau}, \hat{\mathcal{F}})$, where $\hat{\mathcal{F}}$ is as computed in the algorithm. So, step 3 outputs an inequality. □

The following lemma illustrates a key difference between Garg et al. and this work.

Lemma 3.4.4. *If suppliers are substitutes, then all but the $|\mathcal{C}|$ singleton equations of Equation (3.2) are not constraining. Thus, if suppliers are substitutes, optimization over the core of the buyer-supplier game is reduced to solving a polynomially sized linear program.*

Proof. Suppose that the suppliers are substitutes. By the definition of suppliers are substitutes, we have that $\tilde{V}(\mathcal{P}) - \tilde{V}(\mathcal{P} - \mathcal{A}) \geq \sum_{a \in \mathcal{A}} [\tilde{V}(\mathcal{P}) - \tilde{V}(\mathcal{P} - \{a\})]$ for all $\mathcal{A} \subseteq \mathcal{C}$. By the definition of $\tilde{V}$, we have

$$\begin{aligned} &\text{MinEval}(\tau, \text{Bcost}, \mathcal{C} - \mathcal{A}) - \text{MinEval}(\tau, \text{Bcost}, \mathcal{C}) \\ &\geq \sum_{a \in \mathcal{A}} [\text{MinEval}(\tau, \text{Bcost}, \mathcal{C} - \{a\}) - \text{MinEval}(\tau, \text{Bcost}, \mathcal{C})] \end{aligned}$$

for all $\mathcal{A} \subseteq \mathcal{C}$. This implies that if the singleton equations in Equation (3.2) are satisfied, then so are all equations in Equation (3.2). Thus, if suppliers are substitutes, we may drop all non-singleton equations from Equation (3.2) and reduce the number of inequalities to a polynomial in the number of players. □

3.5 Inapproximability of Optimization Over Core Solutions

Consider a buyer-supplier game defined by $(\mathcal{C}, \tau, \text{Bcost})$. We introduced the concept of the focus point price in the introduction. The concept leads us to ask the natural question: What is the difference between the best and worst core outcome for the buyer? In other words, the value of interest is the solution to the linear program: maximize $\sum_{a \in \mathcal{C}} \pi_a$ subject to Equations (3.1), (3.2), and (3.3). This natural question leads us to define the focus point price (FFP) problem as follows: on input $(\mathcal{C}, \tau, \text{Bcost})$, output the optimal value of the afore mentioned linear program.

Define the Necessary Element (NEL) problem as follows. Given parameters $(\mathcal{C}, \tau, \text{Bcost})$ return TRUE if there exist an element $a \in \mathcal{C}$ such that for all $\mathcal{F} \subseteq \mathcal{C}$ satisfying $\text{Eval}(\tau, \text{Bcost}, \mathcal{F}) = \text{MinEval}(\tau, \text{Bcost}, \mathcal{C})$ we have $a \in \mathcal{F}$. Otherwise, return FALSE.

Define the OPT-SET problem as follows. Given parameters $(\mathcal{C}, \tau, \text{Bcost})$, return $\mathcal{F}$ such that $\text{Eval}(\tau, \text{Bcost}, \mathcal{F}) = \text{MinEval}(\tau, \text{Bcost}, \mathcal{C})$.

In this section, we will show that the FPP problem, the OPT-SET problem and the NEL problem are polynomial time equivalent. In Section 3.5.1, we show how to solve the OPT-SET problem in polynomial time if the NEL problem is solvable in polynomial time. In Section 3.5.2, we show the polynomial time equivalence of NEL, OPT-SET, and separation over Equations (3.1), (3.2), and (3.3).

3.5.1 Polynomial Time Reduction from OPT-SET to NEL

In this section, we show that given a polynomial time algorithm to solve the NEL problem, we can solve the OPT-SET problem in polynomial time. All of the lemmas in this section are used solely to prove the section's main result, Lemma 3.5.5.

For a fixed tuple $(\mathcal{C}, \tau, \text{Bcost})$ we say we extend the tuple to contain a *shadow element* for an element $a \subseteq \mathcal{C}$ by creating the extended tuple $(\hat{\mathcal{C}}, \hat{\tau}, \text{Bcost}^*)$, where $\hat{\mathcal{C}} = \mathcal{C} \cup b$ with $b \notin \mathcal{C}$; $\hat{\tau}$ is the same as τ with the addition that $\hat{\tau}(b) = \tau(a)$; and for $\mathcal{A} \subseteq \hat{\mathcal{C}}$, if $b \notin \mathcal{A}$, then $\text{Bcost}^*(\mathcal{A}) = \text{Bcost}(\mathcal{A})$, otherwise $\text{Bcost}^*(\mathcal{A}) = \text{Bcost}((\mathcal{A} - \{b\}) \cup \{a\})$. We call b the *shadow element* corresponding to a.

The *full shadow extension* of $(\mathcal{C}, \tau, \text{Bcost})$ is the tuple $(\hat{\mathcal{C}}, \hat{\tau}, \text{Bcost}^*)$ resulting from extending $(\mathcal{C}, \tau, \text{Bcost})$ to contain a shadow element for each element in $\mathcal{C}$.

First, we reduce OPT-SET to NEL. To show the result, we analyze the following algorithm, which we call the shadow algorithm.

On input $(\mathcal{C}, \tau, \text{Bcost})$,

1 Let $(\hat{\mathcal{C}}^*, \hat{\tau}, \text{Bcost}^*)$ be the full shadow extension of $(\mathcal{C}, \tau, \text{Bcost})$. Let the program variable $\hat{\mathcal{C}}$ equal $\hat{\mathcal{C}}^*$.

2 For each $a \in \mathcal{C}$

- Remove a's corresponding shadow element from $\hat{\mathcal{C}}$.
- Run NEL on $(\hat{\mathcal{C}}, \hat{\tau}, \text{Bcost}^*)$.
- If the return value is TRUE, then add the shadow element back to $\hat{\mathcal{C}}$.
- If the return value is FALSE, then remove a from $\hat{\mathcal{C}}$.

3 Return $\hat{\mathcal{C}} \cap \mathcal{C}$. In other words, we return all elements from $\mathcal{C}$ remaining in $\hat{\mathcal{C}}$, disregarding any shadow elements.

Lemma 3.5.1. *Let $(\mathcal{C}, \tau, \text{Bcost})$ be the input to the shadow algorithm. Let the triple $(\hat{\mathcal{C}}^*, \hat{\tau}, \text{Bcost}^*)$ be the full shadow extension of $(\mathcal{C}, \tau, \text{Bcost})$. If for all $\mathcal{A} \subseteq \hat{\mathcal{C}}^*$ the NEL problem on input $(\mathcal{A}, \hat{\tau}, \text{Bcost}^*)$ is solvable in polynomial time, then the shadow algorithm runs in polynomial time.*

Proof. Creating $\hat{\mathcal{C}}^*$ takes polynomial time since there are $O(|\mathcal{C}|)$ elements. Defining $\hat{\tau}$ takes polynomial time since there are $O(|\mathcal{C}|)$ inputs. Queries to Bcost^* can

be implemented with polynomial overhead on top of queries to Bcost. Thus, the initialization step of the algorithm takes polynomial time.

Consider a single loop iteration. The first, third and forth lines of the loop each take $O(|\mathcal{C}|)$ time. The second step takes polynomial time by the lemma assumption. Thus, a single loop iteration takes polynomial time.

There are $|\mathcal{C}|$ loop iterations and computing the intersection in the algorithm's final step takes $O(|\mathcal{C}|)$ time. Thus the algorithm runs in polynomial time. □

Lemma 3.5.2. *The shadow algorithm maintains the invariant*

$$\text{MinEval}(\tau, \text{Bcost}, \mathcal{C}) = \text{MinEval}(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}}).$$

Proof. Initially, $\text{MinEval}(\tau, \text{Bcost}, \mathcal{C}) = \text{MinEval}(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}})$ by the definitions of $\hat{\mathcal{C}}$, $\hat{\tau}$, and Bcost^*.

Consider the loop iteration for $a \in \mathcal{C}$. Let the corresponding shadow element be b. When we remove or add b to $\hat{\mathcal{C}}$, we have $\text{MinEval}(\tau, \text{Bcost}, \mathcal{C}) = \text{MinEval}(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}})$ by the definitions of $\hat{\tau}$, and Bcost^* and the fact that a is still in $\hat{\mathcal{C}}$.

We only remove both a and b if NEL returned FALSE before the removal of a. Let $\hat{\mathcal{C}}_a$ and $\hat{\mathcal{C}}'_a$ be the value of the variable $\hat{\mathcal{C}}$ before and after the removal of a, respectively. Since NEL returned FALSE on $(\hat{\mathcal{C}}_a, \hat{\tau}, \text{Bcost}^*)$, there exists some $\mathcal{F} \subseteq \hat{\mathcal{C}}_a$ such that $a \notin \mathcal{F}$ and $\text{Eval}(\hat{\tau}, \text{Bcost}^*, \mathcal{F}) = \text{MinEval}(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}}_a)$. Thus, $\mathcal{F} \subseteq \hat{\mathcal{C}}'_a$ and $\text{MinEval}(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}}_a) = \text{MinEval}(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}}'_a)$.

Thus, throughout the algorithm the value of $\text{MinEval}(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}})$ does not change, which concludes the proof. □

Lemma 3.5.3. *Let $\hat{\mathcal{C}}_a$ be the value of the variable $\hat{\mathcal{C}}$ at the end of iteration corresponding to $a \in \mathcal{C}$. If $a \in \hat{\mathcal{C}}_a$, then a is in all OPT-SET solutions on input $(\mathcal{A}, \hat{\tau}, \text{Bcost}^*)$ where $\mathcal{A} = \hat{\mathcal{C}}_a \cap \mathcal{C}$.*

Proof. Let the arguments of the NEL problem which is solved during the iteration corresponding to a be $(\mathcal{B}, \hat{\tau}, \text{Bcost}^*)$.

Since $a \in \hat{\mathcal{C}}_a$, NEL returns TRUE during the iteration corresponding to a.

Suppose there is a solution $\mathcal{F} \subseteq \mathcal{C}$ to OPT-SET on input $(\mathcal{A}, \hat{\tau}, \text{Bcost}^*)$ which does not contain a. Consider $\mathcal{F}$ and $\hat{\mathcal{F}}$ where $\hat{\mathcal{F}}$ contains all the shadow elements of the elements of $\mathcal{F}$. The sets $\mathcal{F}$ and $\hat{\mathcal{F}}$ are disjoint and both subsets of $\mathcal{B}$.

Also, by the definition of $\mathcal{F}$, $\hat{\tau}$ and Bcost*, Eval$(\hat{\tau}, \text{Bcost}^*, \mathcal{F})$ = Eval$(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{F}})$ = MinEval$(\hat{\tau}, \text{Bcost}^*, \mathcal{B})$. Thus, the NEL problem run during the iteration corresponding to a should return FALSE, which is a contradiction. □

Lemma 3.5.4. *Let $\hat{\mathcal{C}}_a$ be the value of the variable $\hat{\mathcal{C}}$ at the end of iteration corresponding to $a \in \mathcal{C}$. We have* MinEval$(\tau, \text{Bcost}, \mathcal{C})$ = MinEval$(\tau, \text{Bcost}, \hat{\mathcal{C}}_a \cap \mathcal{C})$.

Proof. Let $\mathcal{F}$ be such that Eval$(\hat{\tau}, \text{Bcost}^*, \mathcal{F})$ = MinEval$(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}}_a)$. If we have Eval$(\hat{\tau}, \text{Bcost}^*, \mathcal{F}) = M$, then let $\hat{\mathcal{F}} = \emptyset$; otherwise, let $\hat{\mathcal{F}}$ be $\mathcal{F}$ with each shadow element replaced by the corresponding element in $\mathcal{C}$. By the definitions of $\hat{\tau}$ and Bcost*, we have Eval$(\hat{\tau}, \text{Bcost}^*, \mathcal{F})$ = Eval$(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{F}})$. But, by the construction of $\hat{\mathcal{F}}$, we have $\hat{\mathcal{F}} \subseteq \hat{\mathcal{C}}_a \cap \mathcal{C}$.

Thus,

$$\begin{aligned}\text{MinEval}(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}}_a) &= \text{Eval}(\hat{\tau}, \text{Bcost}^*, \mathcal{F}) \\ &= \text{Eval}(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{F}}) \quad \geq \text{MinEval}(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}}_a \cap \mathcal{C}).\end{aligned}$$

Also, by the definition of MinEval, we have that MinEval$(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}}_a)$ is at most MinEval$(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}}_a \cap \mathcal{C})$. So, we have

$$\text{MinEval}(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}}_a) = \text{MinEval}(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}}_a \cap \mathcal{C}).$$

Combining Lemma 3.5.2 with the result from the last paragraph, we have

$$\text{MinEval}(\tau, \text{Bcost}, \mathcal{C}) = \text{MinEval}(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}}_a) = \text{MinEval}(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}}_a \cap \mathcal{C}).$$

And, by the definition of $\hat{\tau}$ and Bcost*, we have MinEval$(\hat{\tau}, \text{Bcost}^*, \hat{\mathcal{C}}_a \cap \mathcal{C})$ equals MinEval$(\tau, \text{Bcost}, \hat{\mathcal{C}}_a \cap \mathcal{C})$. □

Lemma 3.5.5. *Let $(\mathcal{C}, \tau, \text{Bcost})$ be the input to the shadow algorithm. Let the triple $(\hat{\mathcal{C}}^*, \hat{\tau}, \text{Bcost}^*)$ be the full shadow extension of $(\mathcal{C}, \tau, \text{Bcost})$. If for all $\mathcal{A} \subseteq \hat{\mathcal{C}}^*$ the NEL problem on input $(\mathcal{A}, \hat{\tau}, \text{Bcost}^*)$ is solvable in polynomial time, then the OPT-SET problem on input $(\mathcal{C}, \tau, \text{Bcost})$ is solvable in polynomial time.*

Proof. By Lemma 3.5.1, the shadow algorithm runs in polynomial time.

Let $\hat{\mathcal{C}}'$ be the value of the variable $\hat{\mathcal{C}}$ at the end of the algorithm. By Lemma 3.5.4, $\hat{\mathcal{C}}' \cap \mathcal{C}$ is a superset of a solution to the OPT-SET problem. By

Lemma 3.5.3, $\hat{\mathcal{C}}' \cap \mathcal{C}$ is a subset of a solution to the OPT-SET problem. Thus, value returned by the shadow algorithm, $\hat{\mathcal{C}}' \cap \mathcal{C}$, is a solution to the OPT-SET problem. □

3.5.2 Polynomial Time Equivalence of NEL, OPT-SET, and Separation

In this section we show a polynomial time equivalence between the NEL problem, the OPT-SET problem and the separation problem over Equations (3.1), (3.2), and (3.3). The lemmas in this section are used to show the section's main result, Theorem 3.5.8. As a byproduct of the proofs, we also show an hardness of approximation result for the FPP problem with Lemma 3.5.9.

Lemma 3.5.6. *The solution to the FPP problem on input* $(\mathcal{C}, \tau, \text{Bcost})$ *is 0 if and only if the solution to the NEL problem on input* $(\mathcal{C}, \tau, \text{Bcost})$ *is FALSE.*

Proof. First, we prove that if the solution to the NEL problem is FALSE, then the solution to the FPP problem is zero. Consider all of the inequality pairs $\pi_a \leq \text{MinEval}(\tau, \text{Bcost}, \mathcal{C} - \{a\}) - \text{MinEval}(\tau, \text{Bcost}, \mathcal{C})$ and $\pi_a \geq 0$. Since the solution to the NEL problem is FALSE, for each $a \in \mathcal{C}$ there is a $\mathcal{F}_a \subseteq \mathcal{C}$ such that $\text{Eval}(\mathcal{F}_a, \text{Bcost}, \mathcal{C}) = \text{MinEval}(\tau, \text{Bcost}, \mathcal{C})$ and $a \notin \mathcal{F}_a$. Thus the first inequality in the pair reduces to $\pi_a \leq 0$, and the pair of inequalities imply $\pi_a = 0$. This is true for all $a \in \mathcal{C}$. Thus, optimal value of the FPP linear program is zero.

Second, we prove that if the solution to the NEL problem is TRUE, then the solution to the FPP problem is greater than zero. If the solution to NEL is TRUE, then there is some $a \in \mathcal{C}$ such that if $\text{Eval}(\tau, \text{Bcost}, \mathcal{F}) = \text{MinEval}(\tau, \text{Bcost}, \mathcal{C})$ then $a \in \mathcal{F}$. In other words, a is in all solutions to the OPT-SET problem on input $(\mathcal{C}, \tau, \text{Bcost})$.

Thus, for all $\mathcal{A} \subseteq \mathcal{C}$ with $a \in \mathcal{A}$, we have

$$\text{MinEval}(\tau, \text{Bcost}, \mathcal{C} - \mathcal{A}) - \text{MinEval}(\tau, \text{Bcost}, \mathcal{C}) > 0.$$

Let $\lambda = \min_{\substack{\mathcal{A} \subseteq \mathcal{C} \\ a \in \mathcal{A}}} [\text{MinEval}(\tau, \text{Bcost}, \mathcal{C} - \mathcal{A}) - \text{MinEval}(\tau, \text{Bcost}, \mathcal{C})]$. Consider the vector π with $\pi_b = 0$ for all $b \in \mathcal{C} - \{a\}$ and $\pi_a = \lambda$ and $\pi_\mu = M - \text{MinEval}(\tau, \text{Bcost}, \mathcal{C}) - \lambda$. Since $\lambda \leq M - \text{MinEval}(\tau, \text{Bcost}, \mathcal{C})$, this vector is feasible in the focus point price linear program and achieves a objective function value greater than zero. □

Lemma 3.5.7. *If it is possible to approximate the solution to the FPP problem on input* $(\mathcal{C}, \tau, \text{Bcost})$ *within any multiplicative factor, then the NEL problem on input* $(\mathcal{C}, \tau, \text{Bcost})$ *is solvable in polynomial time.*

Proof. Follows from Lemma 3.5.6. □

A set of $(\mathcal{C}, \tau, \text{Bcost})$ instances is *proper* if the following conditions hold:

- Given that $(\mathcal{C}, \tau, \text{Bcost})$ is in the set, then so is $(\mathcal{C}, \hat{\tau}, \text{Bcost})$, where $\hat{\tau}(a) = \tau(a) + \pi_a$ for a vector $\pi \in \Re_+^{|\mathcal{C}|}$.
- Given that $(\mathcal{C}, \tau, \text{Bcost})$ is in the set, then so is $(\mathcal{A}, \hat{\tau}, \text{Bcost}^*)$, where the triple $(\hat{\mathcal{C}}, \hat{\tau}, \text{Bcost}^*)$ is the full shadow extension of $(\mathcal{C}, \tau, \text{Bcost})$ and $\mathcal{A}$ is a subset of $\hat{\mathcal{C}}$.

The definition of proper has a natural interpretation when applied to the transformations of combinatorial minimization problems to buyer-supplier games. For example, for the shortest path problem, the first condition implies that the set of instances is closed with respect to lengthening the edges of the graph. On the other hand, the second condition implies that the set of instances is closed with respect to adding parallel edges or removing a subset of the edges.

Theorem 3.5.8. *On a proper set of instances, the separation problem over Equations (3.1), (3.2), and (3.3), the NEL problem and the OPT-SET problem are polynomial time equivalent.*

Proof. If we can solve the NEL problem on a proper set of instances in polynomial time, then, by Lemma 3.5.5, we can solve the OPT-SET problem in polynomial time.

If we can solve the OPT-SET problem on a proper set of instances in polynomial time, then, by Theorem 3.4.1, we can solve the separation problem over Equations (3.1), (3.2), and (3.3) in polynomial time.

If we can solve the separation problem over Equations (3.1), (3.2), and (3.3) on a proper set of instances in polynomial time, then, by the polynomial time equivalence of separation and optimization, we can optimize linear objective functions over Equations (3.1), (3.2), and (3.3) in polynomial time. If we can optimize linear objective functions in polynomial time, by Lemma 3.5.7 we can solve the NEL problem in polynomial time. □

Lemma 3.5.9. *On a proper set of instances, if it is not possible to solve the OPT-SET problem in polynomial time, it is not possible to approximate the solution to the FPP problem to within any multiplicative factor in polynomial time.*

Proof. Follows from Theorem 3.5.8 and Lemma 3.5.7. □

3.6 A Complementary Combinatorial Algorithm

In this section, we present an efficient combinatorial algorithm for solving the FPP problem for the buyer-supplier minimum spanning tree (MST) game. Before we go on to give the algorithm in Section 3.6.2, we present a useful simplification of the linear program representing the FPP problem for general buyer-supplier games that may be of independent interest.

3.6.1 A Simplification of the FPP Problem

In this section, we give a simplified linear program that may be used to solve the FPP problem.

For this section, fix a buyer-supplier game defined by $(\mathcal{C}, \tau, \text{Bcost})$. Let the buyer-supplier game be derived from the combinatorial minimization problem MinProb as described in Section 3.1. We omit the parameters τ and Bcost from MinEval since they are fixed by the game.

Lemma 3.6.1. *For all $\mathcal{A} \subseteq \mathcal{C}$, we have* $\text{MinEval}(\mathcal{A}) = \min(M, \text{MinProb}(\mathcal{A}))$.

Proof. By the definition of MinEval and Eval, we have

$$\begin{aligned}\text{MinEval}(\mathcal{A}) &= \min_{\mathcal{B}\subseteq\mathcal{A}}[\text{Eval}(\mathcal{B})] \\ &= \min_{\mathcal{B}\subseteq\mathcal{A}}[\text{Bcost}(\mathcal{B}) + \sum_{a\in\mathcal{B}}\tau(a)]\end{aligned}$$

We explicitly instantiate the case when $\mathcal{B} = \emptyset$. Since $\text{Bcost}(\emptyset) = M$, we have

$$\text{MinEval}(\mathcal{A}) = \min(M, \min_{\mathcal{B}\subseteq\mathcal{A}}[\text{Bcost}(\mathcal{B}) + \sum_{a\in\mathcal{B}}\tau(a)])$$

Since $P(\mathcal{B}) = 0$ implies $\text{Bcost}(\mathcal{B}) = M$ and since $\tau(a) \geq 0$ for all $a \in \mathcal{C}$, we have

$$\begin{aligned}\text{MinEval}(\mathcal{A}) &= \min(M, \min_{\substack{\mathcal{B} \subseteq \mathcal{A} \\ P(\mathcal{B})=1}} [\text{Bcost}(\mathcal{B}) + \sum_{a \in \mathcal{B}} \tau(a)]) \\ &= \min(M, \text{MinProb}(\mathcal{A}))\end{aligned}$$

□

Consider the linear program from the FPP problem for the given game. In particular, consider the variable π_μ. The variable can be viewed as a slack variable for the constraint arising from Equation (3.3). In specific, we can write $0 \leq \pi_\mu = M - \text{MinEval}(\mathcal{C}) - \sum_{b \in \mathcal{C}} \pi_b$, where the inequality comes from the constraint $\pi_\mu \geq 0$ and the equality comes from the constraint arising from Equation (3.3). Thus, the following linear program is equivalent to the linear program from the FPP problem

$$\begin{aligned}\max & \sum_{b \in \mathcal{C}} \pi_b \\ \text{s.t.} & \sum_{b \in \mathcal{A}} \pi_b \leq \text{MinEval}(\mathcal{C} - \mathcal{A}) - \text{MinEval}(\mathcal{C}) && \text{for all } \mathcal{A} \subseteq \mathcal{C} \\ & \sum_{b \in \mathcal{C}} \pi_b \leq M - \text{MinEval}(\mathcal{C}) \\ & \pi_b \geq 0 && \text{for all } b \in \mathcal{C}.\end{aligned}$$

Which, in turn, by Lemma 3.6.1 is equivalent to

$$\begin{aligned}\max & \sum_{b \in \mathcal{C}} \pi_b \\ \text{s.t.} & \sum_{b \in \mathcal{A}} \pi_b \leq \min(M, \text{MinProb}(\mathcal{C} - \mathcal{A})) - \min(M, \text{MinProb}(\mathcal{C})) && \forall \mathcal{A} \subseteq \mathcal{C} \\ & \sum_{b \in \mathcal{C}} \pi_b \leq M - \min(M, \text{MinProb}(\mathcal{C})) \\ & \pi_b \geq 0 && \forall b \in \mathcal{C}.\end{aligned}$$

Call the above linear program LP1 and let its optimal value be O_1.

Consider the following linear program

$$\begin{aligned} \max & \sum_{b \in \mathcal{C}} \pi_b & & \\ \text{s.t.} & \sum_{b \in \mathcal{A}} \pi_b \leq \text{MinProb}(\mathcal{C} - \mathcal{A}) - \text{MinProb}(\mathcal{C}) & & \text{for all } \mathcal{A} \subseteq \mathcal{C} \\ & \pi_b \geq 0 & & \text{for all } b \in \mathcal{C}. \end{aligned}$$

Call the above linear program LP2 and let its optimal value be O_2.

Lemma 3.6.2. *If* $\text{MinProb}(\mathcal{C}) \geq M$*, then* $O_1 = 0$.

Proof. Consider the inequality that must be satisfied by all vectors feasible in LP1, $\sum_{b \in \mathcal{C}} \pi_b \leq \min(M, \text{MinProb}(\mathcal{C} - \mathcal{C})) - \min(M, \text{MinProb}(\mathcal{C}))$. The right hand side of this inequality is equal to 0, since $M \leq \text{MinProb}(\mathcal{C})$ and $\text{MinProb}(\emptyset) = \infty$. Thus, we know that $O_1 \leq 0$. We also have $O_1 \geq 0$ since the all zero vector is feasible for LP1. □

Lemma 3.6.3. *If* $\text{MinProb}(\mathcal{C}) < M$ *and* $O_2 \leq M - \text{MinProb}(\mathcal{C})$*, then* $O_1 = O_2$.

Proof. Since $\text{MinProb}(\mathcal{C}) < M$, for any $\mathcal{A} \subseteq \mathcal{C}$ we have

$$\begin{aligned} &\min(M,\text{MinProb}(\mathcal{C} - \mathcal{A})) - \min(M, \text{MinProb}(\mathcal{C})) \\ &\qquad = \min(M, \text{MinProb}(\mathcal{C} - \mathcal{A})) - \text{MinProb}(\mathcal{C}) \\ &\qquad \leq \text{MinProb}(\mathcal{C} - \mathcal{A}) - \text{MinProb}(\mathcal{C}) \end{aligned}$$

Thus, if a vector π is feasible in LP1, then π is also feasible in LP2. Thus, we have $O_1 \leq O_2$.

Let π^* be an optimal vector for LP2. Let $\mathcal{A}$ be any subset of $\mathcal{C}$. Since π^* is feasible in LP2, we have

$$\sum_{b \in \mathcal{A}} \pi^*_b \leq \text{MinProb}(\mathcal{C} - \mathcal{A}) - \text{MinProb}(\mathcal{C}).$$

Since $O_2 \leq M - \text{MinProb}(\mathcal{C})$ we also have

$$\sum_{b \in \mathcal{A}} \pi^*_b \leq \sum_{b \in \mathcal{C}} \pi^*_b = O_2 \leq M - \text{MinProb}(\mathcal{C}).$$

Thus, we have

$$\begin{aligned}\sum_{b\in\mathcal{A}} \pi_b^* &\leq \min(M, \text{MinProb}(\mathcal{C}-\mathcal{A})) - \text{MinProb}(\mathcal{C}) \\ &= \min(M, \text{MinProb}(\mathcal{C}-\mathcal{A})) - \min(M, \text{MinProb}(\mathcal{C}))\end{aligned}$$

Thus, π^* is feasible in LP1. Thus, we have $O_1 \geq O_2$.

Thus, $O_1 = O_2$. □

Lemma 3.6.4. *If* $\text{MinProb}(\mathcal{C}) < M$ *and* $O_2 > M - \text{MinProb}(\mathcal{C})$*, then* $O_1 = M - \text{MinProb}(\mathcal{C})$*.*

Proof. Let π^* be an optimal vector for LP2. Thus, we have

$$\sum_{a\in\mathcal{C}} \pi_a^* = O_2 > M - \text{MinProb}(\mathcal{C})$$

Let $\nu^* \in \Re^{|\mathcal{C}|}$ be any vector such that

$$\begin{aligned}&0 \leq \nu_a^* \leq \pi_a^* && \text{for all } a \in \mathcal{C} \\ &\sum_{a\in\mathcal{C}} \nu_a^* = M - \text{MinProb}(\mathcal{C})\end{aligned}$$

We know that such a ν^* exists since we can decrease each of the coordinates of π^* in turn, not decreasing any coordinate past zero, until we have $\sum_{a\in\mathcal{C}} \nu_a^* = M - \text{MinProb}(\mathcal{C})$.

Thus, ν^* is feasible in the constraint $\sum_{a\in\mathcal{C}} \nu^* \leq M - \min(M, \text{MinProb}(\mathcal{C}))$ of LP1.

Let $\mathcal{A}$ be any subset of $\mathcal{C}$. We have

$$\sum_{a\in\mathcal{A}} \nu_a^* \leq \sum_{a\in\mathcal{C}} \nu_a^* = M - \text{MinProb}(\mathcal{C}).$$

Since π^* is feasible in LP2, we also have

$$\sum_{a\in\mathcal{A}} \nu_a^* \leq \sum_{a\in\mathcal{A}} \pi_a^* \leq \text{MinProb}(\mathcal{C}-\mathcal{A}) - \text{MinProb}(\mathcal{C}).$$

Thus,

$$\sum_{a\in\mathcal{A}} \nu_a^* \leq \min(M, \text{MinProb}(\mathcal{C}-\mathcal{A})) - \text{MinProb}(\mathcal{C})$$
$$= \min(M, \text{MinProb}(\mathcal{C}-\mathcal{A})) - \min(M, \text{MinProb}(\mathcal{C}))$$

and ν^* is feasible in LP1.

The value of LP1 for vector ν^* is $M - \text{MinProb}(\mathcal{C})$. But, the constraint $\sum_{a\in\mathcal{C}} \nu \leq M - \min(M, \text{MinProb}(\mathcal{C}))$ of LP1, which must be satisfied by all feasible vectors ν, tells us that the value of LP1 can be at most $M - \text{MinProb}(\mathcal{C})$. Thus, we have $O_1 = M - \text{MinProb}(\mathcal{C})$. □

By Lemmas 3.6.2, 3.6.3, and 3.6.4, finding the value of MinProb($\mathcal{C}$) and the solution to LP2 is sufficient to solve the FPP problem for given buyer-supplier game.

3.6.2 The Combinatorial Algorithm

Let a graph $G = (\mathcal{V}, \mathcal{E})$ and edge weights $w : \mathcal{E} \to \Re_+$ be given. Let MSTVal : $2^{\mathcal{E}} \to \Re_+$ be a function that takes as input a set of the edges $\mathcal{A} \subseteq \mathcal{E}$ and returns the weight of the minimum spanning tree of the graph induced by the edges of $\mathcal{A}$. If no spanning tree exists, MSTVal returns ∞.

By the transformation in Section 3.1 and Lemma 3.3.21 in the buyer-supplier minimum spanning tree game, we have $\mathcal{C} = \mathcal{E}$, $\tau(a) = w(a)$, and Bcost($\mathcal{A}$) = M if $\mathcal{A}$ does not connect all nodes in $\mathcal{V}$, or 0 otherwise. We omit the parameters τ and Bcost from MinEval, since they are fixed by the game.

We begin with some basics on MSTs.

Lemma 3.6.5. *Let $H = (\mathcal{V}_1, \mathcal{E}_1)$ be any graph. Let $T = (\mathcal{V}_1, \mathcal{E}_1')$ be a minimum spanning tree of the graph H. For $e \in \mathcal{E}_1'$ let the cut created in T by the removal of e be $(\mathcal{A}_e, \mathcal{B}_e)$ for some $\mathcal{A}_e \subseteq \mathcal{V}_1$ and $\mathcal{B}_e \subseteq \mathcal{V}_1$.*

- *The edge e is a minimum weight edge spanning the cut $(\mathcal{A}_e, \mathcal{B}_e)$.*
- *The tree T restricted to vertices in $\mathcal{A}_e$ is a minimum spanning tree of the graph induced by the vertices $\mathcal{A}_e$. A symmetric statement is true for $\mathcal{B}_e$*
- *Let $\mathcal{U}$ be the set of edges spanning the cut and let $a = arg\ min_{b\in\mathcal{U}-\{e\}} w(b)$. We have $w(a) - w(e) = \text{MSTVal}(\mathcal{E}_1 - \{e\}) - \text{MSTVal}(\mathcal{E}_1)$.*

Proof. We use a proof by contradiction. Suppose there is some other edge e' which spans the cut such that $w(e') < w(e)$. Then, we could get a new spanning tree T' of the graph H by replacing the edge e in $\mathcal{E}_1'$ with the edge e'. The tree T' has a strictly smaller weight than the tree T since $w(e') < w(e)$, contradicting the fact that T is a minimum spanning tree of H.

Similarly, if T restricted to the vertices in $\mathcal{A}_e$ is not a minimum spanning tree of the graph induced by the vertices in $\mathcal{A}_e$, we can construct a spanning tree of H with weight strictly smaller than T by connecting e, the tree T restricted to $\mathcal{B}_e$, the minimum spanning tree induced by the vertices in $\mathcal{A}$.

By first second result of this lemma, a minimum spanning tree of the graph induced by $\mathcal{E}_1 - \{e\}$ can be obtained by keeping the portions of T which lie wholly in exactly one of $\mathcal{A}_e$ or $\mathcal{B}_e$ and attaching the smallest weight edge which spans the cut. Thus, we get $w(a) - w(e) = \text{MSTVal}(\mathcal{E}_1 - \{e\}) - \text{MSTVal}(\mathcal{E}_1)$. □

The following lemma is due to Bikhchandani et al., who show that in the MST buyer-supplier game, the suppliers are substitutes [9].

Lemma 3.6.6. *Suppose the graph G is connected and thus* $\text{MSTVal}(\mathcal{E})$ *is finite. For any set $\mathcal{A} \subseteq \mathcal{E}$, we have* $\text{MSTVal}(\mathcal{E} - \mathcal{A}) - \text{MSTVal}(\mathcal{E}) \geq \sum_{e \in \mathcal{A}} [\text{MSTVal}(\mathcal{E} - \{e\}) - \text{MSTVal}(\mathcal{E})]$

Let LP2 be as in Section 3.6.1 with MinProb equal to MSTVal.

Lemma 3.6.7. *The inequalities corresponding to singleton sets $\{e\}$ for $e \in \mathcal{E}$ form an optimal basis for* LP2*. In particular, setting $\pi_e = \text{MSTVal}(\mathcal{E} - \{e\}) - \text{MSTVal}(\mathcal{E})$ for all $e \in \mathcal{E}$ gives an optimal vector for* LP2*.*

Proof. For $\mathcal{A} \subseteq \mathcal{E}$, by Lemma 3.6.6, we have

$$\begin{aligned} \sum_{e \in \mathcal{A}} \pi_e &= \sum_{e \in \mathcal{A}} [\text{MSTVal}(\mathcal{E} - \{e\}) - \text{MSTVal}(\mathcal{E})] \\ &\leq \text{MSTVal}(\mathcal{E} - \mathcal{A}) - \text{MSTVal}(\mathcal{E}). \end{aligned}$$

Thus, π is feasible in LP2. Each inequality of type $\pi_e \leq \text{MSTVal}(\mathcal{E} - \{e\}) - \text{MSTVal}(\mathcal{E})$ is tight, thus it is not possible to increase any coordinate of π. Thus π is an optimal vector and the singleton inequalities form an optimal basis. □

We give a modified Kruskal Algorithm which can be used to compute the optimal value of LP2.

Run Kruskal Algorithm with the following modifications.

- Throughout the algorithm's execution we will keep an auxiliary set of edges, $\mathcal{A}$, which is initially empty.
- When edge e is added to the minimum spanning forest, also add e to the set $\mathcal{A}$.
- Suppose edge e is rejected from addition to the minimum spanning Forest because it creates a cycle. Let the cycle created be $H = (\mathcal{V}', \mathcal{E}')$. For each edge $a \in \mathcal{E}' - \{e\}$, if $a \in \mathcal{A}$, label a with $w(e) - w(a)$ and remove a from $\mathcal{A}$.

Lemma 3.6.8. *Let G be connected and $T = (\mathcal{V}_1, \mathcal{E}_1)$ be the minimum spanning tree computed by the modified Kruskal Algorithm when it is run on $G = (\mathcal{V}, \mathcal{E})$. If $e \in \mathcal{E}_1$ has been labeled, the label is equal to* $\text{MSTVal}(\mathcal{E} - \{e\}) - \text{MSTVal}(\mathcal{E})$. *Otherwise,* $\text{MSTVal}(\mathcal{E} - \{e\}) - \text{MSTVal}(\mathcal{E}) = \infty$.

Proof. Let $\mathcal{U}$ be the set of edges spanning the cut created in T by the removal of e. Also, let a' be any $\arg\min_{b \in \mathcal{U} - \{e\}} w(b)$. By Lemma 3.6.5 we have $w(a') - w(e) = \text{MSTVal}(\mathcal{E} - \{e\}) - \text{MSTVal}(\mathcal{E})$.

Let e be labeled by the modified Kruskal Algorithm when the edge a creates a cycle. For the first part of the lemma, all we must show is $a = a'$.

Let $\mathcal{K}$ be the set of edges which create a cycle involving e during the algorithm. If $b \in \mathcal{E} - \mathcal{U}$, then b does not span the cut created in T by the removal of e. Thus, both vertices of b lie on the same side of the cut. Thus, b cannot create a cycle involving e, since all edges in the cycle except b must be in the tree T. Also, the edge e cannot form a cycle with itself. Thus, we have $\mathcal{K} \subseteq \mathcal{U} - \{e\}$.

Consider any $b \in \mathcal{U} - \{e\}$. The edge b must be rejected by the modified Kruskal algorithm, otherwise the edge would be in T and wouldn't span the cut created in T by the removal of e. Thus, b must create a cycle during the algorithm. Since all edges except b which make up the cycle must be in T and b spans the cut created by e, the cycle created by b includes e. Thus, we have $\mathcal{U} - \{e\} \subseteq \mathcal{K}$, and $\mathcal{U} - \{e\} = \mathcal{K}$.

Since the modified Kruskal Algorithm process edges of G in ascending order of weight and e is labeled by the element from $\mathcal{K}$ which is processed first, we know

$a = \arg\min_{b \in \mathcal{K}} w(b)$. Since $\mathcal{K} = \mathcal{U}$, we also have $a = a'$ and we have shown the first part of the lemma.

If e is not labeled, then $\mathcal{K}$ must be empty. Since $\mathcal{K} = \mathcal{U}$, the set $\mathcal{U}$ must also be empty. And thus, the removal of e must disconnect the graph G. Thus we have, $\text{MSTVal}(\mathcal{E} - \{e\}) - \text{MSTVal}(\mathcal{E}) = \infty$. □

By Lemmas 3.6.7 and 3.6.8, we can find the optimal value to LP2 using the modified Kruskal algorithm. By Lemmas 3.6.2, 3.6.3, and 3.6.4, the minimum spanning tree weight of G and the optimal value of LP2 give us the optimal value of LP1 which is the focus point price of the buyer-supplier minimum spanning tree game on graph G.

Chapter 4

Incomplete Information

4.1 Introduction

Motivated by applications related to auctions, mechanism design, and revenue management, Babaioff et al. recently introduced a generalization of the secretary problem called the online matroid problem [8]. In the online matroid problem, the goal is to build a maximum weight independent set, but we are constrained from knowing the full input to the problem. Instead, the matroid elements are revealed one at a time, and we must immediately decide whether to include the revealed element in the independent set. In such a setting, an online algorithm is said to be *c-competitive* if it is able to produce an independent set with weight within a factor of c of the weight of a maximum weight independent [11]. We say that an online algorithm is *competitive* if it is c-competitive for some constant c.

Babaioff et al. present competitive algorithms for the online matroid problem on bounded left-degree transversal matroids and graphic matroids. They also present a reduction showing that if we have a competitive algorithm for a matroid M, then we can construct a competitive algorithm for a truncated version of M. Babaioff et al. leave open the general online matroid problem and the central case of transversal matroids. As discussed later in this section, the case of transversal matroids unifies the existing results on the online matroid problem. In this chapter we present a competitive online algorithm for weighted matching in transversal matroids, generalizing the results of Babaioff et al. Along with the reduction in Babaioff et al., our results also lead to competitive algorithms on truncated transversal ma-

troids.

Informally, the online weighted transversal matroid matching problem can be described as follows. Consider a bipartite graph, with a set of m left-vertices and a set of n right-vertices. All edges adjacent to the same left-vertex have the same weight – we associate this weight with the left-vertex. The weighted transversal matroid matching problem (WTMM) asks us to find a maximum weight matching in this bipartite graph, and is solvable with the standard matroid greedy algorithm. In the *online* weighted transversal matroid matching problem (OWTMM), we are initially given only the total number of left-vertices, and then a uniformly random permutation of the left-vertices is revealed, one left-vertex at a time. When a vertex is revealed, we learn of both its weight and its incident edges. Upon seeing a particular left-vertex, without knowing the details of the remaining unrevealed left-vertices, we must immediately decide which right-vertex to match it to, if any. An open problem left by Babaioff et al. is to find an algorithm for OWTMM returning a matching with expected weight within a constant of the optimal matching in the corresponding WTMM problem. Theorem 4.7.1 presents such an algorithm.

In the literature, a transversal matroid is often specified by a set of elements E, and a set of subsets $A_1, \ldots, A_n$ of E [33]. A subset $I = \{a_1, \ldots, a_k\}$ of E is considered independent if there is an injective function f mapping I to $\{A_1, \ldots, A_n\}$ such that $x \in f(x)$ for all inputs x. In our presentation, the set of elements E corresponds to the left-vertices, the sets $A_1, \ldots, A_n$ correspond to the right-vertices, and there is an edge between an element of E and a set A_j if the element belongs to the set. An independent set then corresponds to a set of left-vertices for which there exists a matching to the right-vertices.

Perhaps the most well studied online matroid problem is the secretary problem, which first appeared as a folklore problem in the 1950's and has a long history [20, 21]. The problem was first solved by Lindley, who also presents a competitive algorithm for the secretary problem [36]. Competitive algorithms also exist for uniform matroids [32], bounded left-degree transversal matroids, graphic matroids, and truncated matroids [8]. For general matroids, the best known competitive ratio is $O(\log r)$ where r is the rank of the matroid [8].

With the exception of truncated matroids, where the result depends on Karger's matroid sampling theorem [30], all of the matroids for which a competitive algorithm is known are a special case of the transversal matroid. For example,

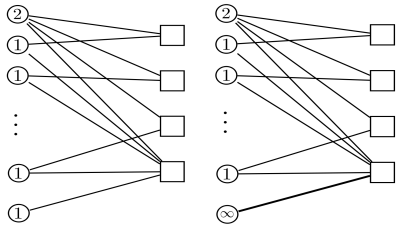
the secretary problem is a transversal matroid with a single right-vertex. The uniform matroid of rank r is a transversal matroid on a complete bipartite graph with r right-vertices. Of course, bounded left-degree transversal matroids are a special case of the transversal matroids. And, finally, the competitive results for graphic matroids follow from a reduction to bounded left-degree transversal matroids. Thus, indeed, transversal matroids play a central role to the theory. For some remarks on the strong connection between general matroids and transversal matroids, see Section 4.8.

4.1.1 Algorithm Motivations

Recall that the secretary problem is OWTMM with a single right-vertex and consider the following classic algorithm for the secretary problem. We sample the first $m/2$ left-vertices we see, rejecting all of them, but recording their edge weights. We set a price for the right-vertex equal to the maximum weight edge we see in the sample. We then match the right-vertex with the first non-sampled left-vertex whose edge weight exceeds the price, if we see such a left-vertex. The algorithm is competitive since with probability at least $1/4$, the second heaviest edge is sampled and the heaviest edge is not sampled.

This simple sample-and-price algorithm is the motivation for most of the competitive algorithms known for online matroid problems, again with the exception of truncated matroids. However, extending this algorithm to work for all, general transversal matroids is not straightforward. For example, Babaioff et al. show that a sample-and-price algorithm with an adaptive sampling time which sets the same price for all the right-vertices does not work. Babaioff et al. also show that a more complicated scheme, where the price required of a non-sampled left-vertex is determined by a circuit of sampled left-vertices also does not work.

One of the main issues that arises in trying to generalize the sample-and-price algorithm is a tension between the need to use sampled heavy left-vertices to price the right-vertices and the requirement that we not over-price too many right-vertices. Consider the example in Figure 4.1a. If in the sample we see the left-vertex of weight 2, we should not over-price all the right-vertices at 2, since that prevents us from matching a large number of vertices of weight 1. The figure is only meant as an illustration, but can be extended to a counter-example for a



(a) Example 1 (b) Example 2

Figure 4.1: Two example transversal matroids exhibiting the tension between using sampled heavy left-vertices for pricing and over-pricing the right-vertices. The figures are meant only to be illustrative, but can be extended to become counter-examples for certain pricing strategies. In 4.1a, we do not want to price all the right-vertices at 2, since we would miss many left-vertices of weight 1. In 4.1b, we want to price the bottom right-vertex at 2, since otherwise we would miss the infinite weight left-vertex.

heavy pricing method by adding $\log m$ clones of the left-vertex of weight 2. On the other hand, consider the example in Figure 4.1b. If we do not set a price of 2 for the bottom-most right-vertex, we would prematurely match that right-vertex to a left-vertex of weight 1 instead of the infinite weight left-vertex. It is natural to consider more complex pricing schemes, such as dynamic prices that change throughout processing, or picking a random subset of the neighbors of a heavy left-vertex and pricing only those neighbors. However, it is both unclear if such schemes are effective and it is difficult to analyze them as they often introduce complicated probabilistic dependencies. It is this tension that leads Babaioff et al. to consider bounded left-degree transversal matroids.

For our results, we avoid the difficulties arising from more complex schemes with the concept of "candidate edges." The candidate edges we introduce have the following important properties. First, each left-vertex i has at most one candidate edges, uniquely determined by the sampled left-vertices heavier than i. In other words, given the sampled left-vertices heavier than i, the candidate edge is the same regardless of whether i is sampled, or where in the random order of non-sampled vertices it appears. Second, the candidate edges of the sampled left-vertices constitute a matching that is within a constant-factor of the max-weight matching on the sampled subgraph.

The analysis following from our definition of candidate edges is essentially the original sample-and-price analysis from the secretary problem, but applied to each right-vertex separately. The algorithm prices right-vertices using only the candidate edges. Furthermore, a non-sampled left-vertex can only be matched using its candidate edge. For a particular right-vertex, as in the secretary problem, we hope that the second-heaviest left-vertex with a candidate edge to the right-vertex is sampled, but the heaviest left-vertex with a candidate edge to the right vertex not sampled. Similarly to the secretary problem, this happens with at least $1/4$ probability.

The overall argument structure is as follows. In Section 4.2, we define some useful notation. In Section 4.3, we define candidate edges and show that they constitute a matching with weight within a constant factor of optimal on the sampled subgraph. In Section 4.4, to avoid any confusion from probabilistic dependencies, we analyze sampled and non-sampled matchings through counting arguments. Our counting argument immediately imply that a matching resulting from candidate edges of non-sampled left-vertices has expected weight within a constant factor of the expected weight of the matching of candidate edges of sampled left-vertices. In Section 4.5, we show that the expected weight of the sampled candidate edge matching is within a constant factor of the max-weight matching on the entire graph. This completes the main technical arguments, since the non-sampled matching is within a constant factor of the sampled matching, which is within a constant of the optimal matching on the whole graph. In Section 4.6, we present a small but clarifying intermediate algorithm between the final online algorithm and the counting arguments presented earlier. Finally, in Section 4.7, we present the online algorithm and conclude the analysis.

4.2 Definitions

In this section, we formally define some quantities and notation we will use throughout the chapter.

Fix a set of n right-vertices, numbered 0 to $n-1$.

Fix a set of m left-vertices, where each left-vertex i is described by a triple of 1) a real number weight, $w(i)$ 2) a unique integer ID and 3) a subset of the right vertices, $\text{Right}(i)$. We define a total order on the left-vertices: we say a left-vertex i

is less than a left-vertex i' if $w(i) > w(i')$ or $w(i) = w(i')$ and i has a smaller unique integer ID. We draw the reader's attention to the fact that smaller left-vertices have greater weight. From here on, we use the integers to denote the left-vertices, with 0 denoting the minimum left-vertex, 1 denoting the second minimum left-vertex and so forth. We draw the reader's attention to the fact that the ordering on the left-vertices is the same as the ordering on the corresponding integers.

For a nonempty subset A of left-vertices or right-vertices let $\text{Min}(A)$ return the minimum vertex, as defined by the corresponding total order.

An edge is a pair (i, j), where i is a left-vertex and j belongs to $\text{Right}(i)$. A matching is a set of edges M such that each vertex appears in at most one edge. For a matching M, let $\text{Left}(M)$, $\text{Right}(M)$, denote the left-vertices, right-vertices, in the matching, respectively.

For a set of left-vertices, A, we say $w(A) = \sum_{i \in A} w(i)$. For a matching M, we say $w(M) = w(\text{Left}(M))$.

To facilitate our proofs, we define the following notation. For a subset of left-vertices L, let $\text{Prefix}(L, i) = \{i' \in L \mid i' < i\}$. Similarly, for a matching M, let $\text{Prefix}(M, i) = \{(i', j) \in M \mid i' < i\}$.

4.3 Algorithm A

In this section we define candidate edges and show the two main properties discussed in Section 4.1.1. The first property, "each left-vertex i has exactly zero or one candidate edges, uniquely determined by the sampled left-vertices heavier than i" corresponds to Lemmas 4.3.1 and 4.3.2. The second property, "the candidate edges of the sampled left-vertices constitute a matching that is within a constant-factor of the max-weight matching on the sampled subgraph" corresponds to Lemma 4.3.5.

First, we define a function $\text{Cands}(i, M)$ that receives a left-vertex i and a matching M, and returns an edge set. The $\text{Cands}(i, M)$ function is as follows:

$M' := \text{Prefix}(M, i)$
$A := \text{Right}(i) - \text{Right}(M')$
if $A = \emptyset$
    return $\emptyset$
else
    return $\{(i, \text{Min}(A))\}$.

Lemma 4.3.1. *For any left-vertex i and matching M,* $\text{Cands}(i, M)$ *either returns the empty set, or* $\{(i, j)\}$*, where j is a right-vertex unmatched in* $\text{Prefix}(M, i)$.

Proof. Follows from the definition of Cands. □

Lemma 4.3.2. *For any left-vertex i and matchings M and M' with* $\text{Prefix}(M, i) = \text{Prefix}(M', i)$*, we have* $\text{Cands}(i, M) = \text{Cands}(i, M')$.

Proof. Follows from the definition of Cands. □

We now define an algorithm for WTMM. Algorithm $\text{AlgA}(L)$ takes a subset of left-vertices L and returns a matching and the algorithm is performed as follows:

$M := \emptyset$
for i in increasing order in L
    $M := M \cup \text{Cands}(i, M)$
return M

Recall that the total order on left-vertices is defined such that i is less than i' if $w(i) > w(i')$ or $w(i) = w(i')$ and i has a smaller unique integer ID.

Lemma 4.3.3. *For a subset of left-vertices L, let $\tilde{M} = \text{AlgA}(L)$, then $\tilde{M}$ is a matching on L and $\tilde{M} = \cup_{k \in L} \text{Cands}(k, \tilde{M})$.*

Proof. We prove the lemma by first proving the following loop invariant in $\text{AlgA}(L)$: M is a matching on $\text{Prefix}(L, i)$ and $M = \cup_{k \in \text{Prefix}(L,i)} \text{Cands}(k, M)$.

The claimed invariant hold initially since $M := \emptyset$ and $i = \text{Min}(L)$. Suppose the claim is true for M and i on entering the loop on which we process i. Let $M' = M \cup \text{Cands}(i, M)$ and i' be the next left-vertex in order from L. We must show the claim holds for M' and i'.

Let $A = \text{Cands}(i, M)$. We split the analysis in two cases. First, suppose $A = \emptyset$. Then, $M' = M$ and the claim holds for M' and i' simply because it holds for M and i.

Second, suppose $A = \{(i, j)\}$, for a right-vertex j unmatched in $\text{Prefix}(M, i)$ (Lemma 4.3.1). Since $\text{Prefix}(L, i') = \text{Prefix}(L, i) \cup \{i\}$, the first part of the invariant holds.

For the second part of the invariant, we have

$$\begin{aligned} M' &= M \cup \text{Cands}(i, M) \\ &= \bigcup_{k \in \text{Prefix}(L,i)} \text{Cands}(k, M) \cup \text{Cands}(i, M) \\ &= \bigcup_{k \in \text{Prefix}(L,i')} \text{Cands}(k, M) \\ &= \bigcup_{k \in \text{Prefix}(L,i')} \text{Cands}(k, M'). \end{aligned}$$

The second equality holds because the loop invariant holds for M and i, the third equality holds by the definition of i', and the final equality holds by Lemma 4.3.2. This proves the invariant.

The lemma statement follows from following the same reasoning as in the inductive step above, but taken for the final iteration of the loop. □

Lemma 4.3.4. *Let M^* be a max-weight matching on L and M be the matching returned by* $\text{AlgA}(L)$*. If $(i, j) \in M^*$ and i is unmatched in M, then there is a i' such that $(i', j) \in M$ and $w(i') \geq w(i)$.*

Proof. By Lemma 4.3.3, $M = \cup_{k \in L} \text{Cands}(k, M)$. Since i is not matched in M and by Lemma 4.3.1, we have $\emptyset = \text{Cands}(i, M)$. By the definition of Cands, the empty set can only be returned if $\text{Right}(i) \subseteq \text{Right}(\text{Prefix}(M, i))$. In other words, every right-vertex in $\text{Right}(i)$ is matched to a left-vertex less than i in M, completing the proof. □

Lemma 4.3.5. *Let M^* be a max-weight matching on L, and M be the matching returned by* $\text{AlgA}(L)$*. Then $w(M) \geq \frac{1}{2} w(M^*)$.*

Proof. By summing the inequality in Lemma 4.3.4 over left-vertices matched in $M^* - M$, we have $w(M) \geq w(M^* - M)$. By definition of intersection, we have $w(M) \geq w(M^* \cap M)$. Combining the two inequalities, we have $2w(M) \geq w(M^*)$. □

4.4 Counting Arguments

In this section, to avoid confusion with probabilistic dependencies, we analyze sampled and non-sampled matchings through counting arguments. As stated in Section

4.1.1, our counting arguments, in specific Lemma 4.4.9, immediately imply that a matching resulting from candidate edges of non-sampled left-vertices has expected weight at least 1/4 of the expected weight of the matching of candidate edges of sampled left-vertices. From this section we only export Lemma 4.4.1, which is used to connect the counting arguments with the final online algorithm, and Lemma 4.4.9.

Let α be a binary string and α_i be the i'th character in the string. Intuitively, the reader should think of a 0 in the i'th position of α as sampling the left-vertex i and of a 1 in the i'th position as not sampling i. For two binary strings α and β, let $\alpha\beta$ denote concatenation. For a binary string α of length at most m, we define the sets of edges $M_0(\alpha), M_2(\alpha), E_0(\alpha)$ recursively as follows.

$$\begin{aligned} M_0(\epsilon) &= E_0(\epsilon) = \emptyset \\ M_2(\alpha) &= \text{Cands}(|\alpha|, M_0(\alpha)) \\ M_0(\alpha 0) &= M_0(\alpha) \cup M_2(\alpha) \\ M_0(\alpha 1) &= M_0(\alpha) \\ E_0(\alpha 0) &= E_0(\alpha) \\ E_0(\alpha 1) &= E_0(\alpha) \cup M_2(\alpha) \end{aligned}$$

Finally, we also define $E_1(\alpha) = M_0(\alpha) \cup E_0(\alpha)$ and $M_1(\alpha)$ to be $\{(i,j) \in E_0(\alpha) \mid j$ appears at most once in $E_0(\alpha)\}$. It is not difficult to show that $M_0(\alpha)$, $M_1(\alpha)$ and $M_2(\alpha)$ are matchings while $E_0(\alpha)$ and $E_1(\alpha)$ are sets of edges.

We give the reader a loose intuitive interpretation of these definitions. Intuitively, one can think of processing the left-vertices in order of increasing weight as we increase the length of α. Then, $M_2(\alpha)$ represents the $|\alpha|$'th candidate edge; $M_0(\alpha)$ represents a matching created from the sampled left-vertices; $E_0(\alpha)$ represents a set of edges created from the non-sampled left-vertices such that each non-sampled left-vertex appears at most once; $E_1(\alpha)$ represents a set of all candidate edges, regardless of whether the corresponding left-vertex is sampled; and $M_1(\alpha)$ represents a matching created from the non-sampled left-vertices.

Lemma 4.4.1. *For a binary string α of length at most m, let $A = \{i \mid \alpha_i = 0\}$ and $B = \{i \mid \alpha_i = 1\}$. We have, $M_0(\alpha) = \bigcup_{i \in A} \text{Cands}(i, M_0(\alpha))$ and $E_0(\alpha) =$*

$\bigcup_{i \in B} \text{Cands}(i, M_0(\alpha))$.

Proof. We prove the claim by induction on the length of α. For $\alpha = \epsilon$, the claim follows from the definition of $M_0(\epsilon)$ and $E_0(\epsilon)$. The inductive claim follows from Lemma 4.3.2 and the recursive definitions of $M_0(\alpha)$ and $E_0(\alpha)$. □

For a set of edges A, let $\deg(A, j)$ denote the degree of the right-vertex j in A. For a left-vertex i and a right-vertex j, we partition the set of binary strings to assist in our counting arguments as follows.

$$\alpha \in S_0(i,j) \text{ if } |\alpha| < i, \deg(E_1(\alpha), j) = 0$$
$$\alpha \in S_1(i,j) \text{ if } |\alpha| = i, \deg(E_1(\alpha), j) = 0, M_2(\alpha) = \{(i,j)\}$$
$$\alpha \in S_2(i,j) \text{ if } \deg(E_0(\alpha), j) = \deg(E_1(\alpha), j) = 1, \alpha = \beta\gamma, \beta \in S_1(i,j)$$
$$\alpha \in S_3(i,j) \text{ if } \deg(E_0(\alpha), j) = 1, \deg(E_1(\alpha), j) > 1, \alpha = \beta\gamma, \beta \in S_2(i,j)$$
$$\alpha \in S_4(i,j) \text{ otherwise.}$$

We give the reader some intuitive interpretation of the above sets. For a particular pair (i, j): $S_0(i, j)$ represents strings where j has never been returned by Cands and we have not yet reached i; $S_1(i, j)$ represents strings where Cands has never before returned j, we have just now reached i and Cands returns $\{(i, j)\}$; $S_2(i, j)$ represents strings where j has been returned exactly once by Cands, when j was returned by Cands it was along with i and i was non-sampled; $S_3(i, j)$ represents strings where the first time Cands returned j it was along with i and i was non-sampled, then j was returned again with some other, sampled vertex i'; finally, $S_4(i, j)$ represents all other strings.

Lemma 4.4.2.

$$\alpha \in S_1(i,j) \Rightarrow \alpha 0 \in S_4(i,j), \alpha 1 \in S_2(i,j)$$
$$\alpha \in S_2(i,j), \exists i' M_2(\alpha) = \{(i',j)\} \Rightarrow \alpha 0 \in S_3(i,j), \alpha 1 \in S_4(i,j)$$
$$\alpha \in S_2(i,j), \forall i' M_2(\alpha) \neq \{(i',j)\} \Rightarrow \alpha 0, \alpha 1 \in S_2(i,j)$$
$$\alpha \in S_3(i,j) \Rightarrow \alpha 0, \alpha 1 \in S_3(i,j)$$
$$\alpha \in S_4(i,j) \Rightarrow \alpha 0, \alpha 1 \in S_4(i,j)$$

Proof. The statement follows straightforwardly by case analysis from the recursive definitions of $M_0(\alpha)$, $E_0(\alpha)$, $E_1(\alpha)$ and the partition $S_0(i,j)$, $S_1(i,j)$, $S_2(i,j)$, $S_3(i,j)$, $S_4(i,j)$. We provide some intuitive discussion.

For the first implication, appending 0 to α places $\{(i,j)\}$ in M_0, which places $\alpha 0$ in S_4. On the other hand, appending 1 to α places $\{(i,j)\}$ in E_0, which places $\alpha 1$ in S_2.

For the second implication, appending 0 to α places $\{(i',j)\}$ in M_0, which places $\alpha 0$ in S_3. On the other hand, appending appending 1 to α places $\{(i',j)\}$ in E_0, which places $\alpha 1$ in S_4.

For the third implication, neither appending 0 nor 1 to α increases the degree of j in E_1, so both extensions of α are in S_2.

For the fourth implication, since α is in S_3, we know that j is in M_0. This means that j can no longer be returned by Cands (Lemma 4.3.1). Thus regardless of the appended character, the extension of α is in S_3.

For the fifth implication, since appending a character to α does not decrease the length of the string or decrease the degree of j in E_0 or E_1, both extensions of α are in S_4. □

Lemma 4.4.3. *For any right-vertex j, left-vertex i and integer k such that $i < k \leq m$, we have*

$$|S_2(i,j) \cap \{0,1\}^k| + 2|S_3(i,j) \cap \{0,1\}^k| = 2^{k-i-1}|S_1(i,j)|.$$

Proof. We prove the lemma by induction on k with a base case $k = i+1$.

For the base case, we have $|S_2(i,j) \cap \{0,1\}^{i+1}| \geq |S_1(i,j)|$ since for every $\alpha \in S_1(i,j)$ we have that $|\alpha| = i$ by the definition of $S_1(i,j)$ and by Lemma 4.4.2 we have $\alpha 1 \in S_2(i,j)$. By the definition of $S_2(i,j)$, all strings in $S_2(i,j)$ have length at least $i+1$. Also by the definition of $S_2(i,j)$, all strings of length $i+1$ in $S_2(i,j)$ are equal to $\alpha 1$ for some string in $S_1(i,j)$. Thus, we have $|S_2(i,j) \cap \{0,1\}^{i+1}| \leq |S_1(i,j)|$. By the definition of $S_3(i,j)$, all strings in $S_3(i,j)$ have length at least $i+2$. Thus, $|S_3(i,j) \cap \{0,1\}^{i+1}| = 0$, completing the base case.

To show the inductive step, notice that the right-hand side of the claimed equality exactly doubles as we increase k by one. We must show that the left-hand

side of the equality also exactly doubles. By Lemma 4.4.2, for every $\alpha \in S_2(i,j)$, either $\alpha 0 \in S_3(i,j)$ or both $\alpha 0, \alpha 1 \in S_2(i,j)$. In either case, the count from the first summand of the left-hand side of the equality doubles when we increase k by one. Again, by Lemma 4.4.2, for every $\alpha \in S_3(i,j)$, we have $\alpha 0, \alpha 1 \in S_3(i,j)$. So, the count from the second summand of the left-hand side of the equality also doubles. So the left-hand side at least doubles when we increase k by one.

By Lemma 4.4.2, the only ways for a string extended by one character to be in $S_2(i,j)$ or $S_3(i,j)$ is by extending a string in $S_1(i,j)$, $S_2(i,j)$ or $S_3(i,j)$. In the previous paragraph, we accounted for extensions from strings in $S_2(i,j)$ or $S_3(i,j)$. All strings in $S_1(i,j)$ have length i, but in the inductive case $k > i+1$. Thus, the left-hand side of the claimed equality exactly doubles. □

Lemma 4.4.4. *For any right-vertex j, left-vertex i and integer k such that $i < k \leq m$, we have*

$$|(S_2(i,j) \cup S_3(i,j)) \cap \{0,1\}^k| \geq 2^{k-i-2}|S_1(i,j)|.$$

Proof. By Lemma 4.4.3, we have that $|S_2(i,j) \cap \{0,1\}^k| + 2|S_3(i,j) \cap \{0,1\}^k| = 2^{k-i-1}|S_1(i,j)|$. We can increase the left-hand side to get $2|S_2(i,j) \cap \{0,1\}^k| + 2|S_3(i,j) \cap \{0,1\}^k| \geq 2^{k-i-1}|S_1(i,j)|$. Since $S_2(i,j)$ and $S_3(i,j)$ are disjoint, we have $|(S_2(i,j) \cup S_3(i,j)) \cap \{0,1\}^k| \geq 2^{k-i-2}|S_1(i,j)|$. □

Lemma 4.4.5. *For any right-vertex j and integer k such that $k \leq m$, we have*

$$\sum_{\alpha \in \{0,1\}^k} w(M_1(\alpha), j) \geq \sum_{0 \leq i < k} w(i) 2^{k-i-2} |S_1(i,j)|,$$

where $w(M_1(\alpha), j)$ denotes the weight of the left-vertex matched to j in $M_1(\alpha)$ or zero if j is unmatched.

Proof. By the definitions of M_1, S_2 and S_3, we have that $(i,j) \in M_1(\alpha)$ if and only if $\alpha \in (S_2(i,j) \cup S_3(i,j))$. Furthermore, by the definition of M_1, if α has length k, then only left-vertices less than k can be matched in M_1. The left-hand side of the claimed inequality is thus equal to $\sum_{0 \leq i < k} w(i)|(S_2(i,j) \cup S_3(i,j)) \cap \{0,1\}^k|$. Applying Lemma 4.4.4 gives the desired result. □

Lemma 4.4.6. *For any integer k such that $k \leq m$, we have*

$$\sum_{\alpha \in \{0,1\}^k} w(M_1(\alpha)) \geq \sum_{0 \leq i < k} \sum_{0 \leq j < n} w(i) 2^{k-i-2} |S_1(i,j)|.$$

Proof. Follows from summing the result of Lemma 4.4.5 over all $0 \leq j < n$. □

Lemma 4.4.7. *For any right-vertex j and integer k such that $k \leq m$, we have*

$$\sum_{\alpha \in \{0,1\}^k} w(M_0(\alpha), j) \leq \sum_{0 \leq i < k} w(i) 2^{k-i} |S_1(i,j)|,$$

where $w(M_0(\alpha), j)$ is equal to the weight of the left-vertex matched to j in $M_0(\alpha)$ or zero if j is unmatched.

Proof. To prove this lemma, we first introduce some helpful claims and definitions. For any binary string α of length at most k define $f(\alpha)$ as the set of proper prefixes β of α such that $M_2(\beta) = \{(|\beta|, j)\}$. The following two claims follow directly from the definitions.

Claim 1: For any binary string α of length at most k, we have $\deg(E_1(\alpha), j) = |f(\alpha)|$.

Claim 2: For any binary string α of length at most k, we have $|f(\alpha)| = 0$ implies $w(M_0(\alpha), j) = 0$.

Let A denote all $\alpha \in \{0,1\}^k$ such that $f(\alpha) \neq \emptyset$. For all $\alpha \in A$ let $g(\alpha)$ denote the shortest string in $f(\alpha)$.

Claim 3: For any α in A, we have $f(g(\alpha)) = \emptyset$. Since any proper prefix of $g(\alpha)$ is also a proper prefix of α.

Claim 4: For any α in A, we have $\deg(E_1(g(\alpha)), j) = 0$ and $M_2(g(\alpha)) = \{(|g(\alpha)|, j)\}$. Follows from Claims 1 and 3.

Claim 5: For any α in A, we have $0 \leq |g(\alpha)| < k$ and $g(\alpha) \in S_1(|g(\alpha)|, j)$. Follows from Claim 4, the definition of S_1 and since $g(\alpha) \in f(\alpha)$.

Claim 6: For any α in A, we have $w(M_0(\alpha), j) \leq w(|g(\alpha)|)$. Since $M_0(\alpha)$ is a matching, $\deg(M_0(\alpha), j)$ is either zero or one. If it is zero, the claim is trivial. If it is one, then $M_0(\alpha)$ contains a unique (i, j) for some left-vertex i. Thus, $M_2(\beta) = \{(i, j)\}$ for some proper prefix β of α of length i. By the definition of g, $|g(\alpha)| \leq |\beta| = i$. So, the claim follows.

Claim 7: For all $0 \leq i < k$ and β in $S_1(i,j)$ we have $|g^{-1}(\beta)| \leq 2^{k-i}$. Since $\beta \in S_1(i,j)$, we have $|\beta| = i$. Since $g(\alpha)$ is a prefix of α, $|g^{-1}(\beta)|$ is at most the number of k bit extensions of β, which is 2^{k-i}.

We are now ready to prove the lemma

$$\begin{aligned}
\sum_{\alpha \in \{0,1\}^k} w(M_0(\alpha), j) &= \sum_{\alpha \in A} w(M_0(\alpha), j) \\
&= \sum_{0 \leq i < k} \sum_{\beta \in S_1(i,j)} \sum_{\alpha \in g^{-1}(\beta)} w(M_0(\alpha), j) \\
&\leq \sum_{0 \leq i < k} \sum_{\beta \in S_1(i,j)} \sum_{\alpha \in g^{-1}(\beta)} w(|g(\alpha)|) \\
&= \sum_{0 \leq i < k} \sum_{\beta \in S_1(i,j)} \sum_{\alpha \in g^{-1}(\beta)} w(i) \\
&\leq \sum_{0 \leq i < k} \sum_{\beta \in S_1(i,j)} w(i) 2^{k-i} \\
&= \sum_{0 \leq i < k} w(i) 2^{k-i} |S_1(i,j)|,
\end{aligned}$$

where the first step follows from Claim 2 and the definition of A; the second step follows from Claim 5; step three follows from Claim 6; step four follows since $\alpha \in g^{-1}(\beta)$ and $\beta \in S_1(i,j)$ implies $i = |\beta| = |g(\alpha)|$; step five follows from Claim 7; and step 6 is immediate. □

Lemma 4.4.8. *For any integer k such that $k \leq m$, we have*

$$\sum_{\alpha \in \{0,1\}^k} w(M_0(\alpha)) \leq \sum_{0 \leq i < k} \sum_{0 \leq j < n} w(i) 2^{k-i} |S_1(i,j)|.$$

Proof. Follows from summing the result of Lemma 4.4.7 over all $0 \leq j < n$. □

Lemma 4.4.9. *For any integer k such that $k \leq m$, we have*

$$\sum_{\alpha \in \{0,1\}^k} w(M_1(\alpha)) \geq \frac{1}{4} \sum_{\alpha \in \{0,1\}^k} w(M_0(\alpha)).$$

Proof. Follows from Lemmas 4.4.8 and 4.4.6. □

4.5 Analysis Under a Probability Distribution

In this section we begin working with probability distributions and show that the expected weight of the sampled candidate edge matching is within a constant factor of the max-weight matching on the entire graph (Lemma 4.5.2). We tie these results with Section 4.4, to show that the expected weight of the non-sampled matching is within a constant the weight of a max-weight matching on the entire graph (Lemma 4.5.4), completing the main technical portion of the argument. The only result exported from this section is Lemma 4.5.4.

Define a function Sample, which takes an m-bit binary string string α such that $\text{Sample}(\alpha) = \{i \mid \alpha_i = 0\}$. We introduce a probability distribution $\mathcal{P}$ on m-bit binary strings α. In $\mathcal{P}$, each α_i independently has an equal chance of $\alpha_i = 0$ and $\alpha_i = 1$.

Lemma 4.5.1. *Let M^* be a max-weight matching and M_α denote a max-weight matching on the subgraph induced by* $\text{Sample}(\alpha)$ *for a binary string α. Then,* $\text{Exp}[w(M_\alpha)] \geq \frac{1}{2}w(M^*)$.

Proof. We have $\text{Exp}[w(M_\alpha)] \geq \sum_{i \in \text{Left}(M^*)} \Pr[\alpha_i = 0]w(i) = \frac{1}{2}w(M^*)$, where the first step follows from the linearity of expectation and observing that the matching M_α as a weight at least as big as the weight of a matching $M'_\alpha = \{(i,j) \in M^* \mid \alpha_i = 0\}$. □

Lemma 4.5.2. *Let M^* be a max-weight matching. Then, the following inequality holds:*

$$\text{Exp}[w(\text{AlgA}(\text{Sample}(\alpha)))] \geq \frac{1}{4}w(M^*).$$

Proof. Let M_α be a max-weight matching on $\text{Sample}(\alpha)$, for a string α. Then, we have

$$\begin{aligned} \text{Exp}[w(\text{AlgA}(\text{Sample}(\alpha)))] &\geq \text{Exp}[\frac{1}{2}w(M_\alpha)] \\ &\geq \frac{1}{4}w(M^*), \end{aligned}$$

where the first step follows from Lemma 4.3.5 and the second step follows from Lemma 4.5.1 and the linearity of expectation. □

Lemma 4.5.3. *Let α be any m-bit binary string and $A = \text{Sample}(\alpha)$. We have $M_0(\alpha) = \text{AlgA}(A)$.*

Proof. Follows from the definition of AlgA and M_0. □

Lemma 4.5.4. *Let M^* be a max-weight matching. We have $\text{Exp}[w(M_1(\alpha))] \geq \frac{1}{16} w(M^*)$.*

Proof. We have

$$\begin{aligned}\text{Exp}[w(M_1(\alpha))] &\geq \frac{1}{4}\text{Exp}[w(M_0(\alpha))] \\ &= \frac{1}{4}\text{Exp}[w(\text{AlgA}(\text{Sample}(\alpha)))] \\ &\geq \frac{1}{16} w(M^*),\end{aligned}$$

where the first inequality follows from Lemma 4.4.9, the equality follows from Lemma 4.5.3, and the final inequality follows from Lemma 4.5.2. □

4.6 Intermediate Algorithm

In this section we analyze a useful intermediate algorithm between the counting arguments and the final online algorithm. In specific, in the counting argument, we process the non-sampled left-vertices in decreasing order of weight. In this section we use Lemma 4.4.1 to argue that we can process the non-sampled left-vertices in an arbitrary order. This is similar to what happens in the original sample-and-price algorithm in the secretary problem. In the secretary problem, we depend on the fraction of time when the second highest bidder is sampled and the highest bidder is not. When this happens, we can process the non-sampled bidders in an arbitrary order, since only one of them meets the required price.

We define an algorithm AlgB(α) that takes an m-bit binary string α, and returns a matching. The AlgB(α) function is as follows:

$M := \text{AlgA}(\text{Sample}(\alpha))$
$A := \{0, \ldots, m\} - \text{Sample}(\alpha)$
$E := \emptyset$
for i in arbitrary order from A:

$E := E \cup \text{Cands}(i, M)$

return the matching of pairs (i, j) in E where j appears at most once in E.

Lemma 4.6.1. *For any m-bit binary string α, we have* $\text{AlgB}(\alpha) = M_1(\alpha)$.

Proof. The results of Lemma 4.5.3 give that for any m-bit binary string α we have $M_0(\alpha) = \text{AlgA}(\text{Sample}(\alpha))$. Applying Lemma 4.4.1, we also have that $E_0(\alpha) = \bigcup_{i \in A} \text{Cands}(i, M_0(\alpha))$, where A is as in AlgB. Thus, at the end of the loop in AlgB, E is equal to $E_0(\alpha)$. The lemma statement follows from the definition of $M_1(\alpha)$ and the last line of AlgB. □

4.7 Online Algorithm

In this section, we define and analyze the final online algorithm, which is closely related to the algorithm in Section 4.6. The main difference between the two algorithms is that the online algorithm must rely on the random permutation of left-vertices for sampling whereas up to now we have discussed a simpler direct sampling method, where each element has an equal chance of being sampled or not. With Lemma 4.7.2 we show that the direct sampling method and the method of sampling from the random permutation induce the same distribution. The main theorem follows immediately from our previous results.

Define the online algorithm as follows. Initially, we are given the set of right-vertices, and the total number of left-vertices we will see, m. The algorithm ONLINE proceeds in two phases.

First phase:

$k := \text{Bin}(m, \frac{1}{2})$, where Bin is the binomial distribution.

Reject the first k vertices, not matching them to anything.

Let B be the set of all the rejected vertices.

$M_0 := \text{AlgA}(B)$.

Second phase:

We are given M_0 from the first phase.

We build a matching M_1, initialized to $\emptyset$.

On receiving a left-vertex i:

$A := \text{Cands}(i, M_0)$

if $A \neq \emptyset$ and the right-vertex in A is unmatched in M_1
$M_1 := M_1 \cup A$
return the matching M_1

Lemma 4.7.1. *Let α be a m-bit binary string, $B = \text{Sample}(\alpha)$ and M_1^B be a matching returned by ONLINE when B is sampled in the first phase. Then, $w(M_1^B) \geq w(\text{AlgB}(\alpha))$.*

Proof. ONLINE and AlgB perform the same operations on the vertices that are not sampled, with the small optimization that ONLINE matches a right vertex j to the first left-vertex i such that $\{(i,j)\} = \text{Cands}(i, M_0)$, while AlgB does not match any right-vertex j that is returned twice by Cands. □

Lemma 4.7.2. *Consider a set A of m elements. Let $\mathcal{P}$ be the probability distribution where each $a \in A$ independently has an equal chance of being sampled or not sampled.*

Let $\mathcal{P}'$ be the probability distribution where we first pick a k from $\text{Bin}(m, \frac{1}{2})$. Then, from a uniformly random permutation of A, we only sample the first k elements.

The probability distributions $\mathcal{P}$ and $\mathcal{P}'$ are equal.

Lemma 4.7.2. We simply prove that each particular sample set $B \subseteq A$ appears with the same probability in both distributions. The probability of any particular sample B under $\mathcal{P}$ is $\frac{1}{2^m}$. We must show the same is true under $\mathcal{P}'$. Let $|B| = k'$. The probability of B under $\mathcal{P}'$ is $\left[\frac{k'!(m-k')!}{m!}\right] \cdot \left[\frac{m!}{k'!(m-k')!}\frac{1}{2^m}\right]$, where the first term comes from the probability that the elements in B come first in the random permutation, and the second term comes from the probability that k' is chosen as the cutoff for the sampling. □

Theorem 4.7.1. *Let M^* be a max-weight matching. Given a uniformly random permutation of the left-vertices, ONLINE returns a matching whose weight is least $\frac{1}{16}w(M^*)$ in expectation.*

Proof. Let $\mathcal{P}$ and $\mathcal{P}'$ be as defined in Lemma 4.7.2. We must show that the expected weight of the matching produced by ONLINE under $\mathcal{P}'$ is at least $\frac{1}{16}w(M^*)$.

By Lemma 4.7.1, for each possible sample selection ONLINE returns a matching with weight at least as large as the matching returned by AlgB on the same

sample. Thus, the expectation of ONLINE under $\mathcal{P}'$ is at least as large as the expectation of AlgB under $\mathcal{P}'$. Lemma 4.7.2, gives us the result that the expectation of AlgB(α) when α is drawn from $\mathcal{P}'$ is the same as that when α is drawn from $\mathcal{P}$. Lemma 4.6.1 gives us the result that the expectation of AlgB(α) under $\mathcal{P}$ is the same as the expectation of $M_1(\alpha)$ under $\mathcal{P}$. Finally, Lemma 4.5.4 shows that the expectation of $M_1(\alpha)$ under $\mathcal{P}$ is at least $\frac{1}{16}w(M^*)$, completing the result. □

4.8 Concluding Remarks

In this section we make some remarks on the strong connection between transversal matroids and general matroids. The connection comes from the following characterization of a basis: B is a basis of a matroid M iff B is a minimal set having non-empty intersection with every co-circuit of M [43]. With this characterization, one can think of a general matroid as a bipartite graph in the following way. Let the matroid elements be the left-vertices and co-circuits be the right-vertices. Let there be an edge between an element and a co-circuit if the element belongs to the co-circuit. An independent set in the general matroid is then a combinatorial structure which is close to a matching, but not the same. Consider taking a particular element into an independent set we are constructing. On taking in the element, we cover all the co-circuits containing that element because they have non-empty intersection with the constructed independent set. After that, to increase the independent set, we can only take elements which cover some uncovered co-circuits. So, in a sense, independent sets match left-vertices to subsets of right-vertices. Perhaps it is possible to come up with a sample-and-price scheme for pricing co-circuits to extend the results of this chapter to general matroids, solving the online matroid problem?

Chapter 5

Summary and Future Directions

5.1 Summary

In this document, we present our contributions to three main algorithmic concerns arising from large-scale distributed computing: dynamic membership, selfishness, and incomplete information.

To address the concern of dynamic membership, in Chapter 2, we address load balancing in distributed hash tables by analyzing variants of a related graph-based coupon collector process. Using the scaffold of a delay sequence argument, we show that several variants of the graph-based coupon collector lead to linear cover times, both in expectation and with high probability. These results directly imply load balance guarantees in distributed hash tables. We also show super-linear lower bounds for the cover times of several variants, showing that slight differences in the process specification affect the cover time results.

To address the concern of selfishness, in Chapter 3, we study optimization over the core of buyer-supplier games. After describing a method to obtain buyer-supplier games from combinatorial minimization problems, we show that the transferable utility core and non-transferable utility core of buyer-supplier games have the same characterization. Using that characterization, we show both positive and negative results on optimization over the core. We show that optimizing over the core of a buyer-supplier game is polynomial time equivalent to solving the underlying combinatorial minimization problem.

To address the concern of incomplete information, in Chapter 4, we present

our results on the online matroid problem. Specifically, we show a competitive algorithm for solving the online weighted transversal matroid matching problem. Our results generalize several previous results on the online matroid problem and are based on a novel sample-and-price algorithm. When combined with a previously known reduction, our results also give competitive algorithms for the truncations of transversal matroids.

5.2 Future Directions

In this section we briefly list several other problems of interest related to dynamic membership, selfishness, and incomplete information.

Can we show load-balance results for joining and leaving a DHT? In Chapter 2, we analyzed a random process related to load balance in a distributed hash table. As discussed in Chapter 1, an $O(n)$ cover time can be used to show that, under the described DHT join procedure, if the DHT initially has a single machine with a load of 1, then after $n-1$ machines join the DHT, the load of every machine is $O(\frac{1}{n})$ with high probability. However, our analysis does not extend to machines dynamically joining and leaving the distributed hash table. Can we show load-balance guarantees after a sequence of n arbitrarily ordered join/leave operations?

Can we find a combinatorial algorithm for the FPP problem on the buyer-supplier shortest path game? In Chapter 3, we presented a combinatorial algorithm for the FPP problem on the buyer-supplier MST game. But, as discussed in that chapter, in the buyer-supplier MST game, suppliers are substitutes. This property makes the FPP problem particularly easy, as it can be expressed by a polynomially sized linear program. In the buyer-supplier shortest path game, on the other hand, suppliers are not substitutes. Currently, even though the shortest path problem itself can be solved by a simple greedy algorithm, the only known method of solving the FPP problem on the buyer-supplier shortest path game is the ellipsoid method.

Can we find algorithms to compute the focus point price in games other than buyer-supplier games? In Chapter 3, we presented our results on computing the focus point price in buyer-supplier games with respect to the core solution concept. The focus point price concept is widely applicable and easily

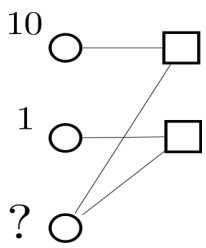


Figure 5.1: An example showing that the algorithm for the online transversal matroid problem presented in Chapter 4 does not produce a truthful mechanism. Suppose that the left-vertices with bids 10 and 1 are sampled and used to price the corresponding right-vertices. If the left-vertex, labeled ?, has a true weight of 11, it has incentive to be untruthful. Reporting 11 would result in being matched to the top right-vertex and a utility of 1, while reporting 2, would result in being matched to the bottom right-vertex and a utility of 10.

extends to scenarios other than the core of buyer-supplier games. The focus point price concept can be applied to any combination of a solution concept and a game. We are interested to see if we can find bounds on the focus point price, or algorithms to compute the focus point price in a wide class of games other than the buyer-supplier games which we have already studied.

When do repeated auctions lead to core solutions? There have already been investigations into the relationship between the core and auctions. For example, Bikhchandani and Ostoy show that under certain conditions, the outcome of a VCG auction is in the core of a corresponding game between the buyers and the auctioneer [10]. As we suggested in Chapter 1, the core solution concept was developed with the notion that players communicate freely in advance of playing the game. We are interested in whether repeatedly executing an auction, where each execution is done without communication, leads to core solutions. The idea behind this investigation is that the history of previous games serves as sufficient communication to lead to a core outcome. Indeed, investigations on the connections between competitive equilibria, such as Nash equilibria, and cooperative equilibria, such as the core, exist. However, many of the existing results deal with convergence as the number of players goes to infinity, or are limited to repeating a specific game between a small number of players [38, 57, 61].

Is there an algorithm for online weighted transversal matroid matching that produces a truthful mechanism? Several of the previous results on the

online matroid problem produce truthful auctions [8, 32]. Our algorithm, perhaps due to its more complex nature, does not produce a truthful auction. To illustrate this point, consider Figure 5.1. Suppose that the left-vertices with bids 10 and 1 are sampled and used to price the corresponding right-vertices. Furthermore, suppose that the left-vertex labeled ? has a true weight of 11. This third left-vertex has incentive to not be truthful, since reporting 11 would result in being matched to the top right-vertex and a utility of 1. On the other hand, reporting 2, would result in being matched to the bottom right-vertex and a utility of 10. This motivates our interest in an algorithm for the online transversal matroid matching problem that produces a truthful mechanism.

Can we solve the online matroid problem? In Chapter 4, we present an algorithm for solving the online weighted transversal matroid matching problem. At the end of the chapter, we discuss the connections between transversal matroids and general matroids. We are interested in resolving the general case of the online matroid problem. We do not have strong evidence or beliefs on whether there exists an algorithm for the general case. Either an algorithm, or a lower-bound result would be an interesting resolution to the online matroid problem.

Large-scale distributed computing has a rich algorithmic landscape with many unexplored problems. It is certain that the algorithmic concerns of dynamic membership, selfishness, and incomplete information will increase in prominence as future systems include interactions between larger numbers of individuals.

Bibliography

[1] M. Adler, E. Halperin, R. M. Karp, and V. V. Vazirani. A stochastic process on the hypercube with applications to peer-to-peer networks. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 575–584, 2003.

[2] M. Aghassi and D. Bertsimas. Robust game theory. *Mathematical Programming: Series A and B*, 107:231–273, June 2006.

[3] R. Aleliunas. Randomized parallel communication. In *Proceedings of the 1st annual Symposium on Principles of Distributed Computing*, pages 60–72, Ottawa, Canada, August 1982.

[4] N. Alon. Problems and results in extremal combinatorics, II. *Discrete Mathematics*, 2007. In press.

[5] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley, New York, NY, 1991.

[6] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proceedings of the 42nd annual IEEE Symposium on Foundations of Computer Science*, pages 482–491, October 2001.

[7] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. *SIAM Journal on Computing*, 29:180–200, 1999.

[8] M. Babaioff, N. Immorlica, and R. Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete algorithms*, pages 434–443, January 2007.

[9] S. Bikhchandani, S. de Vries, J. Schummer, and R. Vohra. Linear programming and vickrey auctions. In *Mathematics of the Internet: E-Auction and Markets*, pages 75–115. Springer-Verlag, New York, NY, 2002.

[10] S. Bikhchandani and J. Ostroy. The package assignment model. *Journal of Economic Theory*, 107:377–406, 2002.

[11] A. Borodin and El R. Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, Cambridge, U.K., 1998.

[12] A. Caprara and A. N. Letchford. Computing good allocations for combinatorial optimization games. 2006.

[13] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a nash equilibrium. In *Proceedings of the 38th annual ACM Symposium on Theory of Computing*, pages 71–78, Seattle, WA, May 2006.

[14] X. Deng, T. Ibaraki, and H. Nagamochi. Algorithmic aspects of the core of combinatorial optimization games. *Mathematics of Operations Research*, 24:751–766, August 1999.

[15] X. Deng, C. Papadimitriou, and S. Safra. On the complexity of equilibria. In *Proceedings of the 34th annual ACM Symposium on Theory of Computing*, pages 67–71, Montreal, Quebec, May 2002.

[16] F. Y. Edgeworth. *Mathematical psychics, an essay on the application of mathematics to the moral sciences*. A. M. Kelley, New York, NY, 1961.

[17] P. Erdös and A. Rényi. On a classical problem of probability theory. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 6:215–219, 1961.

[18] U. Faigle and W. Kern. On the core of ordered submodular cost games. *Mathematical Programming*, 87:483–499, May 2000.

[19] J. Feigenbaum, C. H. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. In *Proceedings of the 32nd annual ACM Symposium on Theory of Computing*, pages 218–227, May 2000.

[20] T. Ferguson. Who solved the secretary problem? *Statistical Science*, 4:282–289, August 1989.

[21] P. R. Freeman. The secretary problem and its extensions: A review. *International Statistical Review*, 51:189–206, August 1983.

[22] R. Garg, V. Kumar, A. Rudra, and A. Verma. Coalitional games on graphs: core structure, substitutes and frugality. In *Proceedings of the 4th ACM conference on Electronic Commerce*, pages 248–249, San Diego, CA, June 2003.

[23] D. B. Gillies. *Some Theorems on n-Person Games*. PhD thesis, Princeton University, 1953.

[24] M. X. Goemans and M. Skutella. Cooperative facility location games. *Journal of Algorithms*, 50:194–214, 2004.

[25] G. H. Gonnet. Expected length of the longest probe sequence in hash code searching. *Journal of the ACM*, 28:289–304, April 1981.

[26] A. Hald. A. de Moivre: 'De Mensura Sortis' or 'On the Measurement of Chance'. *International Statistical Review / Revue Internationale de Statistique*, 52:229–262, December 1984.

[27] J. C. Harsanyi. Games with incomplete information played by "Bayesian" players. *Management Science*, 14, 1968.

[28] S. Jukna. *Extremal Combinatorics*. Springer-Verlag, Berlin, 2001.

[29] N. Kaplan. A generalization of a result of erdös and rényi. *Journal of Applied Probability*, 14:212–216, March 1977.

[30] D. Karger. Random sampling and greedy sparsification for matroid optimizaiton problems. *Mathematical Programming*, 82:41–81, June 1998.

[31] A. R. Karlin, D. Kempe, and T. Tamir. Beyond VCG: frugality of truthful mechanisms. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 615–624, Oct. 2005.

[32] R. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete algorithms*, pages 630–631, January 2005.

[33] E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Dover Publications, Mineola, NY, 2001.

[34] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, and Hypercubes*. Morgan-Kaufmann, San Mateo, CA, 1991.

[35] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems. In *Proceedings of the 21st annual Symposium on Principles of Distributed Computing*, pages 233–242, Monterey, CA, July 2002.

[36] D. V. Lindley. Dynamic programming and decision theory. *Applied Statistics*, 10:39–51, March 1961.

[37] N. Madras and A. D. Sokal. The pivot algorithm: A highly efficient monte carlo method for the self-avoiding walk. *Journal of Statistical Physics*, 50:109–186, January 1988.

[38] V. S. Mirrokni and A. Vetta. Convergence issues in competitive games. pages 183–194. October 2004.

[39] M. D. Mitzenmacher. *The Power of Two Choices in Randomized Load Balancing*. PhD thesis, University of California at Berkeley, 1996.

[40] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, UK, 1995.

[41] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley and Sons, New York, NY, 1988.

[42] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proceedings of the 31st annual ACM Symposium on Theory of Computing*, pages 129–140, May 1999.

[43] J. Oxley. What is a matroid? *Cubo*, 5:179–218, 2003.

[44] M. Pál and E. Tardos. Group strategy proof mechanisms via primal-dual algorithms. In *Proceedings of the 44th annual IEEE Symposium on Foundations of Computer Science*, pages 584–593, October 2003.

[45] C. H. Papadimitriou. Algorithms, games, and the internet. In *Proceedings of the 33rd annual ACM Symposium on Theory of Computing*, pages 749–753, July 2001.

[46] C. H. Papadimitriou and T. Roughgarden. Computing equilibria in multi-player games. In *Proceedings of the 16th annual ACM-SIAM Symposium on Discrete algorithms*, pages 82–91, January 2005.

[47] A. G. Ranade. How to emulate shared memory. *Journal of Computer and System Sciences*, 42:307–326, 1991.

[48] D. Randall. Rapidly mixing markov chains with applications in computer science and physics. *Computing in Science & Engineering*, 8:30–41, March 2006.

[49] K. Roberts. The characterization of implementable choice rules. In *Aggregation and Revelation of Preferences*, pages 321–348. North-Holland, Amsterdam, 1979.

[50] T. Roughgarden. *Selfish Routing and the Price of Anarchy*. MIT Press, Cambridge, MA, 2005.

[51] T. Roughgarden and E. Tardos. How bad is selfish routing? *J. ACM*, 49:236–259, 2002.

[52] T. Roughgarden and E. Tardos. Bounding the inefficiency of equilibria in nonatomic congestion games. *Games and Economic Behaviour*, 47:389–403, 2004.

[53] L. S. Shapley. Notes on the n-person game III: Some variants of the von Neumann-Morgenstern definition of solution. Research memorandum, RAND Corporation, Santa Monica, CA, 1952.

[54] L. S. Shapley. Stochastic games. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 39, pages 1095–1100, Washington, DC, October 1953. National Academy of Sciences.

[55] L. S. Shapley and M. Shubik. The assignment game i: The core. *International Journal of Game Theory*, 1:111–130, 1972.

[56] M. Shubik. *Game Theory In The Social Sciences.* MIT Press, Cambridge, Massachussetts, 1984.

[57] B. T. Smith and J. H. Case. Nash equilibria in a sealed bid auction. *Management Science*, 22:487–497, December 1975.

[58] E Upfal. Efficient schemes for parallel communication. *Journal of the ACM*, 31:507–517, July 1984.

[59] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, Princeton, NJ, 1953.

[60] R. Weber. Multiple-object auctions. In *Auctions, bidding, and contracting: Uses and theory*, pages 165–191. New York University Press, New York, NY, 1983.

[61] R. Wilson. Competitive exchange. *Econometrica*, 46:577–585, May 1978.

Vita

Nedialko Dimitrov graduated from Grand Blanc High School, Grand Blanc, MI, in 1999. In the same year, he entered the University of Michigan. He received a Bachelor of Science from the University of Michigan, with concentrations in Mathematics and Computer Science. In September, 2002, he entered graduate studies in the Computer Science Department of the The University of Texas at Austin.

Permanent Address: 7205 Hart. ln. Apt 3013
Austin, TX, 78731

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.