NETWORK DESIGN AND MANAGEMENT WITH STRATEGIC AGENTS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Elliot Anshelevich

July 2005

NETWORK DESIGN AND MANAGEMENT WITH STRATEGIC AGENTS

Elliot Anshelevich, Ph.D.
Cornell University 2005

Network design is a fundamental problem for which it is important to understand the effects of strategic behavior. Given a collection of self-interested agents who want to form a network connecting certain endpoints, the set of stable solutions (the Nash equilibria) may look quite different from the centrally enforced optimum. We study the price of stability, i.e., the quality of the best Nash equilibrium compared to the optimum network cost. The best Nash equilibrium solution has a natural meaning of stability in this context: it is the optimal solution that can be proposed from which no user will "deviate".

We consider two versions of this game: one where agents may divide the cost of the edges they use in any manner they desire, and one where the cost of each such edge is divided equally between the agents whose connections make use of it. We also study the price of stability of a related game where agents are attempting to route their traffic while incurring minimal latency, instead of trying to construct a network of minimal cost.

Since "self-interest" could have various meanings, the notion of strategic agents enfolds many types of agents including malicious, altruistic, and obedient agents under a common terminology. For example, we can think of a decentralized algorithm with local knowledge as a set of "self-interested" agents performing local optimizations. The comparisons between stable solutions and the centrally enforced optimum now become concerns about the algorithm's performance compared to a centralized algorithm with global knowledge. We elaborate on this in the contexts of local load balancing and packet routing algorithms, by showing that simple decentralized algorithms with local knowledge can ensure good global properties even in the presence of a powerful adversary. Specifically, we show that in the dynamic load balancing problem, where the arrival and departure of jobs is modeled by an adversary restricted in its power, a classic local balancing algorithm keeps the loads approximately balanced. We extend these results to many other scenarios, with dynamic packet routing as one of them.

BIOGRAPHICAL SKETCH

Elliot Anshelevich was born in February 1979, in Moscow, Russia. He received a B.S. in Computer Science from Rice University in May 2000, an M.S. from Cornell University in 2004, and expects to receive a Ph.D. in Computer Science from Cornell University in July 2005.

To parents, love, and life.

ACKNOWLEDGEMENTS

My five years at Cornell have been wonderful, largely because of the outstanding direction I received from the Cornell faculty. First, I am greatly thankful to my Ph.D. advisor, Jon Kleinberg, for guiding me in every possible way, putting up with my endless questions, and never letting me falter. I also thank Éva Tardos, for giving me a lot of amazing and treasured advice, as well as working with me and pointing me to some great research. I thank David Shmoys and Al Demers for many fascinating discussions, and for some of the most interesting classes I have ever attended. I would also like to thank Eric Friedman, for introducing me to game theory.

The work in this thesis would not exist if I did not have the opportunity to collaborate with some amazing people while at Cornell. I would especially like to acknowledge Anirban Dasgupta, David Kempe, Martin Pál, Tom Wexler, and Tim Roughgarden for being my co-authors, and/or for the wonderful discussions we have shared on many research (and non-research) topics. I am also grateful to all of the people I met at Bell Labs for supporting me and providing an extremely fun and fruitful research environment. Dating back to my undergraduate days at Rice University, I am greatly indebted to Lydia Kavraki, who first introduced me to computer science research and in whose class my passion for algorithms and theory truly began.

There is no point in attempting to list all of the friends I made over the years, who have made my life what it is today. I would certainly like to thank all of my Cornell house-mates and friends, who have made my years there a truly memorable experience. I also want to mention M.A., H.L., M.H., B.W., S.O., K.G., S.Y., and F.L. You have shown me how entertaining and varied this world can be. Finally, I am deeply thankful to my parents and my fiancée, for my past, my future, and all that there is.

This work was supported by an NSF Graduate Research Fellowship and ITR grant 0311333.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1
Introduction

1.1 Self-Interested Agents in Networks

All networks, whether physical, virtual, or social, can be thought of as made up of individual agents. These agents can correspond to the nodes or edges of the network, or can correspond to something more complex. Traditional study of network design issues assumes that there is a single network designer, and its design is accepted and implemented by all parts of the network. Similarly, study involving network management assumes that some protocol is implemented in the network, and all parts of the network follow this protocol (or almost all parts, while a small part of the network is malicious). Many real-world networks such as the Internet, however, are developed, built, and maintained by a large number of independent agents, all of whom act in their own interests. These agents only have the desire to cooperate if it fits with their personal interests, and so we cannot assume that they would follow the directives of a central network designer or a protocol imposed on them. The outcomes of such self-interested behavior often have properties very different from the centrally designed or managed networks which traditional algorithmic results have focused on. Therefore, dealing with such systems requires quite different methods and considerations.

The main question we will address in this thesis is whether decentralized self-interested behavior of agents in a network can result in an outcome that benefits the network as a whole. Does the lack of centralized coordination destroy any chances of the network having good global properties in certain contexts? If the agents' behavior yields a network with good global properties, how good is this network compared to the one that could be formed by a centralized designer if the agents were not self-interested? And perhaps even more importantly, can we influence the agents slightly and induce them to create a "good" network?

Since "self-interest" could have various meanings, the above discussion enfolds many types of agents including malicious, altruistic, and obedient agents under a common terminology. For example, we can think of a decentralized algorithm with local knowledge as a set of "self-interested" agents performing local optimizations. The above questions now become concerns about the algorithm's performance compared to a centralized algorithm with global knowledge. In Chapter 6 we elaborate on this in the contexts of load balancing and packet routing algorithms. The main contributions of this thesis, however, are the results about network design and management by strategic agents given in Chapters 3 and 4. These results are for an important network scenario which we call *The Connection Game*.

1.2 The Connection Game

As mentioned above, most of the work done in traditional computer science focuses on obedient or malicious agents. Agents with rational self-interest which does not necessarily cause them to hurt the network have been more the province of game theoretical research. Game theory is a long-standing field that studies the behavior of rational agents in various contexts and with various types of interactions. These interactions are usually called "games". In this thesis we do not assume previous knowledge of game theory, and define basic game theoretic terminology in Chapter 2. We also talk about traditional game theoretic goals and how they relate to the goals of our network research.

In this thesis we study network settings where the system behavior arises from the actions of a large number of independent agents, each motivated by self-interest and optimizing an individual objective function. As a result, the global performance of the system may not be as good as in a case where a central authority can simply dictate a solution; rather, we need to understand the quality of solutions that are consistent with self-interested behavior. Recent theoretical work has framed this type of question in the following general form: how much worse is the

solution quality at a Nash equilibrium[1], relative to the quality at a centrally enforced optimum? This question is important because Nash equilibria are often the only viable outcomes of agent interactions. Questions of this genre have received considerable attention in recent years, for problems including routing [72, 75, 76], load balancing [22, 23, 52, 74], facility location [81], and flow control [4, 26, 78].

An important issue to explore in this area is the middle ground between centrally enforced solutions and completely unregulated anarchy. In most networking applications, it is not the case that agents are completely unrestricted; rather, they interact with an underlying protocol that essentially proposes a collective solution to all participants, who can each either accept it or defect from it. As a result, it is in the interest of the protocol designer to seek the *best* Nash equilibrium; this can naturally be viewed as the optimum subject to the constraint that the solution be *stable*, with no agent having an incentive to unilaterally defect from it once it is offered. Hence, one can view the ratio of the solution quality at the best Nash equilibrium relative to the global optimum as a *price of stability*, since it captures the problem of optimization subject to this constraint. This thesis, together with some recent work [7, 76], proposes this definition (termed the "optimistic price of anarchy" in [7]); it stands in contrast to the larger line of work in algorithmic game theory on the *price of anarchy* [67] — the ratio of the *worst* Nash equilibrium to the optimum — which is more suited to worst-case analysis of situations with essentially no protocol mediating interactions among the agents. Indeed, one can view the activity of a protocol designer seeking a good Nash equilibrium as being aligned with the general goals of mechanism design — producing a game that yields good outcomes when players act in their own self-interest.

In this thesis we consider a simple network design game where every agent has a specific connectivity requirement, i.e. each agent has a set of terminals and wants to build a network in which his terminals are connected. Possible edges in the network have costs and each agent's goal is to pay as little as possible. This game can be viewed as a simple model of network creation. Alternatively, by studying the best Nash equilibria, our game provides a framework for understanding those networks that a central authority could persuade selfish agents to purchase and maintain, by specifying to which parts of the network each agent contributes. An interesting feature of our game is that selfish agents will find it in their individual interests to *share* the costs of edges, and so effectively cooperate. In addition, our game demonstrates certain interesting properties of self-interested behavior on networks, some of which differ markedly from those observed in other games previously studied. Since we are dealing with algorithmic game theory, in this thesis we will use the terms "agent" and "player" interchangeably, since a "player" in a game-theoretic sense exactly corresponds to a self-interested agent.

More precisely, we study the following network game for N players, which we call the *Connection Game*. For each game instance, we are given an undirected graph G with non-negative edge costs. Players form a network by purchasing some subgraph of G. Each player has a set of specified terminal nodes that it would like to see connected in the purchased network. With this as their goal, players purchase some edges, so that these edges become built in our network. Each player would like to minimize its total payments, but insists on connecting all of its terminals. We allow the cost of any edge to be shared by multiple players. Finding the centralized optimum of the connection game, i.e. the network of bought edges which minimizes the sum of the players' contributions, is the classic network design problem of the generalized Steiner forest [1, 42]. We are most interested in deterministic Nash equilibria of the connection game, and in the price of stability, as the price of anarchy in our game can be quite bad. In a game theoretic context it might seem natural to also consider *mixed* Nash equilibria when agents can randomly choose between different strategies. However, since we are modeling the construction of large-scale networks, randomizing over strategies is not a realistic option for players.

In the above description of the Connection Game, we did not specify how several players share the cost of an edge that is used by all of them to connect their terminals. In this thesis, we

[1]Recall that a Nash equilibrium is a state of the system in which no agent has an interest in unilaterally changing its own behavior. See Chapter 2 for discussion and a formal definition.

will consider two cost-sharing mechanisms for the case where multiple players share an edge. In the first case, we will leave it up to the players how to share the costs of the edges. In this model, which we call the *Connection Game with Arbitrary Sharing*, players offer payments indicating how much they will contribute towards the purchase of each edge in G. If the players' payments for a particular edge e sum to at least the cost of e, then the edge is considered *bought*, which means that e is added to our network and can now be used by any player. Notice that once an edge is purchased, any player can use this edge to satisfy its connectivity requirement, even if that player contributed nothing to the cost of the edge.

This "arbitrary sharing" mechanism has some drawbacks, such as the fact that there can be free riders on an edge – players who do not contribute anything to the cost of the edge and yet use it to connect their terminals. Also, as we demonstrate in Chapter 3, the price of stability for the general version of this game can be quite large. In our second model, we instead impose a "fair sharing" rule, so that any player using an edge to connect its terminals must pay its fair share of the edge cost. Although there are in principle many possible cost-sharing mechanisms, research in this area has converged on a few mechanisms with good theoretical and empirical behavior; here we focus on the following particularly natural one: the cost of each edge is shared equally by the set of all users whose paths contain it. This equal-division mechanism has a number of basic economic motivations; it can be derived from the Shapley value [63], and it can be shown to be the unique cost-sharing scheme satisfying a number of different sets of axioms [32, 44, 63]. For the former reason, we will refer to it as the *Shapley cost-sharing mechanism*, and to this model as the *Fair Connection Game*. In the Fair Connection Game, we obtain fair cost sharing at the expense of the agents' freedom, as we take away their ability to decide how to share the costs of edges on their own. The properties of the Fair Connection Game drastically differ from the Connection Game with Arbitrary Sharing. In fact, we can look at cost-sharing mechanisms in this context from a "mechanism design" point of view: what small restrictions can we put on the players that would guarantee good game outcomes? We discuss this further in Section 2.2.2.

1.3 Results for the Connection Game with Arbitrary Sharing

In Chapter 3, we study deterministic Nash equilibria of this connection game, and prove bounds on the price of stability. We also explore the notion of an *approximate equilibrium*, and study the question of how far from a true equilibrium one has to get to be able to use the optimum solution, i.e. how unhappy would the agents have to be if they were forced to pay for the socially optimal design. We view this as a two parameter optimization problem: we would like to have a solution with cost close to the minimum possible cost, and where users would not have large incentives to deviate from their current behavior. Finally, we examine how difficult it is to find equilibria at all.

Our results include the following.

- In Section 3.2 we consider the special case when the goal of each player is to connect a single terminal to a common source. We prove that in this case, there is a Nash equilibrium, the cost of which is equal to the cost of the optimal network. In other words, with a single source and one terminal per player, the price of stability is 1. Furthermore, given an $\varepsilon > 0$ and an α-approximate solution to the optimal network, we show how to construct in polynomial time an $(1+\varepsilon)$-approximate Nash equilibrium (players only benefit by a factor of $(1+\varepsilon)$ in deviating from their current behavior) whose total cost is within a factor of α to the optimal network.

  We generalize these results in two ways. First, we can extend the results to the case when the graph is directed and players seek to establish a directed path from their terminal to the common source. Note that problems in directed graphs are often significantly more complicated than their undirected counterparts [17, 31]. Second, players do not have to insist on connecting their terminals at all cost, but rather each player i may have a maximum cost $\max(i)$ that it is willing to pay, and would rather stay unconnected if its cost exceeds $\max(i)$.

- In Section 3.3 we consider the general case, when players may want to connect more than 2

terminals, and they do not necessarily share a single source node. In this case, there may not exist a deterministic Nash equilibrium. When deterministic Nash equilibria do exist, the costs of different equilibria may differ by as much as a factor of N, the number of players, and even the price of stability may be nearly N. However, in Section 3.3 we prove that there is always a 3-approximate equilibrium that pays for the optimal network. Furthermore, we show how to construct in polynomial time a $(4.65+\varepsilon)$-approximate Nash equilibrium whose total cost is within a factor of 2 to the optimal network.

• Finally, in Section 3.4 we show that determining whether or not a Nash equilibrium exists is NP-complete when the number of players is part of the input. In addition, we give a lower bound on the approximability of a Nash on the centralized optimum in our game.

To avoid reader confusion about various extensions of our results, Table 1.1 summarizes which extensions hold for our main results of Chapter 3.

	Single-Source	Multi-Source
Result	$\exists$ Nash equilibrium with cost equal to OPT	$\exists$ 3-approx Nash eq. with cost equal to OPT
Can handle directed	Yes	No
Players can have more than 2 terminals	No	Yes
Players can have maximum they are willing to pay, max(i)	Yes	Yes
Polynomial time alg	Finds $(1+\varepsilon)$-approx Nash equilibrium that costs at most $1.55 \cdot OPT$	Finds $(4.65+\varepsilon)$-approx Nash equilibrium that costs at most $2 \cdot OPT$

Table 1.1: Extensions for our main results in the Connection Game with Arbitrary Sharing (OPT is the cost of the centralized optimum)

1.4 Results for the Fair Connection Game

The general question for the Fair Connection Game is to determine how this fair cost-sharing mechanism serves to influence the strategic behavior of the users, and what effect this has on the structure and overall cost of the network one obtains.

• In Section 4.2, our first main result is that in every instance of the network design problem with Shapley cost-sharing, there always exists a Nash equilibrium of total cost at most $\log N$ times optimal, and there are directed examples where there are no cheaper Nash equilibria. In other words, the price of stability for the Fair Connection Game is at most $\log N$, and this is tight for directed graphs.

We prove this result using a *potential function* method due to Monderer and Shapley [62] and Rosenthal [71] (see also [15]): one defines a potential function Φ on possible solutions and shows that any improving move by one of the users (i.e. to lower its own cost) reduces the value of Φ. Since the set of possible solutions is finite, it follows that any sequence of improving moves leads to a Nash equilibrium. The goal of Monderer and Shapley's and Rosenthal's work was to prove existence statements of this sort; for our purposes, we make further use of the potential function to prove a bound on the price of stability. Specifically, we give bounds relating the value of the potential for a given solution to the overall cost of that solution; if we then iterate best-response dynamics starting from an optimal solution, the potential does not increase, and hence we can bound the cost of any solution that we reach. Thus, for this network design game, best-response dynamics starting from the optimum does in fact always lead to a good Nash equilibrium.

We can extend our basic result to a number of more general settings. To begin with, the $\log N$ bound on the price of stability extends directly to the case in which users are selecting arbitrary subsets of a ground set (with elements' costs shared according to the Shapley value), rather than paths in a graph; it also extends to the case in which the cost of each edge is a non-decreasing concave function of the number of users on it. In addition, our results also hold if we introduce capacities into our model; each edge e may be used by at most u_e players, where u_e is the capacity of e.

• While the $\log N$ price of stability bound is tight for directed graphs, it is not for the undirected case. In Section 4.3, we give improved price of stability bounds for the undirected case of the Fair Connection Game.

• Since a number of our proofs are obtained by following the results of best-response dynamics via a potential function, it is natural to investigate the speed of convergence of best-response dynamics for this game. In Section 4.4, we show that it converges to a Nash equilibrium in polynomial time for the case of two players, but that with N players, it can run for a time exponential in N. Whether there is a way to schedule players' moves to make best-response converge in a polynomial number of steps for this game in general is an interesting open question.

• In Section 4.5 we consider a natural generalization of the cost-sharing model that carries us beyond the potential-function framework and raises interesting questions for further work. Specifically, suppose each user has a *weight* (perhaps corresponding to the amount of traffic it plans to send), and we change the cost-allocation so that user i's payment for edge e is equal to the ratio of its weight to the total weight of all users on e. In addition to being intuitively natural, this definition is analogous to certain natural generalizations of the Shapley value [61]. The weighted model, however, is significantly more complicated: there is no longer a potential function whose value tracks improvements in users' costs when they greedily update their solutions, and it is an open question whether best-response dynamics will always converge to a Nash equilibrium. We have obtained some initial results here, including the convergence of best-response dynamics when all users seek to construct a path from a node s to a node t (the price of stability here is 1), and in the general model of users selecting sets from a ground set, when each element appears in the sets of at most two users. We know that some results will necessarily look quite different in the weighted case; for example, using a construction involving user weights that grow exponentially in N, we can show that the price of stability can be as high as $\Omega(N)$. Recently, Chen and Roughgarden [18] proved general results on the price of stability and the existence of approximate Nash equilibria in this game.

• Finally, in Chapter 5, we arrive at a more technically involved set of extensions by adding latencies to the network design problem. Here each edge has a concave *construction cost* $c_e(x)$ when there are x users on the edge, and a *latency cost* $d_e(x)$; the cost experienced by a user is the full latency plus a fair share of the construction cost, $d_e(x) + c_e(x)/x$. We give general conditions on the latency functions that allow us to bound the price of stability in this case at $d \cdot \log N$, where d depends on the delay functions used. Moreover, we obtain stronger bounds in the case where users experience only delays, not construction costs; this includes a result that relates the cost at the best Nash equilibrium to that of an optimum with twice as many players, and a result that improves the potential-based bound on the price of stability for the single-source delay-only case.

1.5 Simple Agents in Load Balancing and Packet Routing

In the previous sections we consider a scenario with many independent agents with limited goals, and we ask questions about the global properties of the network resulting from their behavior. This is a very general framework, and can include agents which are not technically "self-interested", because we defined a self-interested agent as one that is trying to optimize its personal objective function without caring about the network as a whole. If we make this

objective function coincide with the global objective function of the network, however, then it is in the agents' best interest to form the centrally optimal network.

This still leaves the question of how the agents would actually form a good solution. In the above discussion of the price of stability we suggest that often a designer can suggest a good solution to the agents, or use the underlying protocol to ensure that the agents end up with a stable and globally good solution. This is not always possible, however. In many cases the agents must form a good solution themselves, in a fully decentralized manner, using only local knowledge.

Consider a load balancing scenario where there is a network of processors with different amounts of jobs running on each processor, and suppose that each processor is an independent agent with only local knowledge. Even if these are "obedient" agents and all they desire is to balance the overall load, this may be a very difficult thing to do without centralized coordination and knowledge. Nevertheless we can find a mechanism such that no matter how the agents behave and whether they form a stable solution or not, the global properties of the network will not be too bad. Specifically we show that a simple local protocol is such that the load imbalance in the network never becomes unbounded. We now define our load balancing model a bit more thoroughly.

We consider a basic model of load balancing in a distributed network, which has formed the basis of a number of earlier studies [2, 41, 58, 64, 69]. A network of identical processors is represented by an undirected graph $G = (V, E)$. There are a number of *jobs* to be processed in the system, abstractly represented by unit-size *tokens*. Time progresses in discrete steps called *rounds*; in a given round, each token is held by one of the nodes, which is viewed as processing the associated job, and the *load* on a node is defined to be the number of tokens it holds. The goal is to *balance* the loads, so that no single node has too many tokens; this can be accomplished by transmitting tokens between neighboring nodes of the graph, at a rate of one token per edge per round. We are particularly interested in *local* algorithms for this problem: rather than using a centralized approach to coordinate the movement of tokens, each node will simply compare its load to those of its neighbors, and decide whether to move a token across an edge based on this information.

This model is clearly very simple in a number of respects, particularly in the uniformity of the processors (nodes) and jobs (tokens), and the fact that any job can be executed on any processor. More subtly, it is not even clear in all settings that balancing load evenly is the optimal strategy in a distributed network of processors (see e.g. [21]). At the same time, however, the model cleanly captures the basic constraints imposed by an underlying interconnection topology in the process of distributing jobs through a network, as evidenced by the results of previous analysis [2, 41, 58, 64, 69]; the simplicity of the model allows one to reason very clearly about the effect of these constraints.

An Adversarial Model

Motivated by work in the related area of packet routing [11, 12, 16, 5], Muthukrishnan and Rajaraman proposed an *adversarial* framework for studying dynamic load balancing in the token-based model we have been discussing [64]. Rather than considering a probabilistic process that generates tokens, they posit an *adversary* that is allowed at the beginning of each round to introduce tokens at some nodes (corresponding to new jobs) and remove tokens from others (corresponding to jobs that have finished). Subsequently, an algorithm is allowed to move tokens across edges as above so as to try to maintain balanced loads. This alternation of moves by the adversary and algorithm continues for an arbitrary number of rounds. Note that by allowing the adversary to control the removal of tokens as well as their arrival, one is modeling a worst-case assumption that jobs may have arbitrary duration, and the algorithm does not know how much processing time a job has remaining until the moment it ends.

If we let a_t denote the average number of tokens per node in the system at the beginning of round t, and $h_t(v)$ denote the number of tokens at node v (the *height* of v) at round t, then

the goal of a dynamic load balancing algorithm in this model is to keep $h_t(v)$ close to a_t for all nodes v and rounds t. Formally, we say that an algorithm is *safe* [2] against a given adversary if there is a constant B such that $|h_t(v) - a_t| \leq B$ for all nodes v and rounds t. Note that safety in this context imposes a bound on deviation from the average; it is not required that the actual number of tokens in the system remains bounded.

As in the case of packet routing [16, 5], one needs to find a suitable restriction on the adversary: an arbitrarily powerful adversary could flood a particular set of nodes $S \subseteq V$ with tokens much faster than these nodes can spread the tokens out to the rest of the graph, and thereby prevent any algorithm from maintaining safety. This consideration motivates the following natural definition of an adversary [64] with *rate* r. For a set $S \subseteq V$, let $e(S)$ denote the set of edges with exactly one end in S, and $\delta_t(S)$ the net increase in tokens in set S due to the addition and removal of jobs in round t (note that $\delta_t(S)$ could be negative). If the heights of nodes in S were to change precisely according to average, then the net change in tokens in S would be $|S| \cdot (a_{t+1} - a_t)$. One wants the difference between these two quantities to be "accounted for" by the edges in $e(S)$. We say that the adversary has *rate* r if for all $S \subseteq V$, one has

$$|\delta_t(S) - |S|(a_{t+1} - a_t)| \quad \leq \quad r \cdot |e(S)|, \tag{1.1}$$

For rate $r > 1$, there are adversaries against which no algorithm (whether online or offline) can be safe. Muthukrishnan and Rajaraman gave a local-control algorithm that is safe against all adversaries of rate r, for every $r < 1$. As an open question, they asked whether there exists a local-control algorithm that is safe against all adversaries of rate 1.

Our Results

We begin by providing a local-control load balancing algorithm that is safe against every adversary of rate 1, thereby resolving the open question of Muthukrishnan and Rajaraman. In fact, we show that the following simple rule has this safety property, for every value of the parameter θ:

> At any round t, if the number of tokens on node u exceeds the number of tokens on its neighbor v by at least θ, then u moves a token to v.

This type of algorithm was considered in earlier work on the static model by Aiello et al. [2], as well as by many of the subsequent papers. Setting $\theta = 2\Delta + 1$, where Δ is the maximum node degree in G, yields the specific algorithm studied by Muthukrishnan and Rajaraman.

Beyond simply showing the safety of local algorithms at the *critical rate* $r = 1$, our analysis is based on a new proof technique in which a potential function bound is maintained not only for the entire node set V, but for every subset of V. Compared to [64], we obtain significantly improved bounds on the deviation from the average, and a simpler proof. Specifically, we show that the maximum possible deviation from the average is $O(\Delta n)$, where n is the number of nodes of G, and this is asymptotically optimal in the worst case; the analysis in [64] had established a bound of $O(\Delta^2 n^{2.5}(1-r)^{-1})$ when $r < 1$. Our analysis also shows safety in a more general model where edges of G can appear and disappear over time.

Following this, we introduce a *multi-commodity* version of this load balancing model. We consider a system in which there are k distinct types of jobs. The jobs of one given type induce the same load on each processor; but the different types of jobs place different resource requirements on the nodes, and so we require the load balancing condition to apply to each type separately. Formally, we have the same adversarial model as before with a network G and a collection of tokens; but now the tokens are partitioned into k *commodities* and the safety requirement must hold when the tokens of each commodity are considered separately. In a single round, at most

[2] In most literature and in [8] such an algorithm is called *stable*. To avoid confusion with the stability of Nash equilibria, in this thesis we will refer to such an algorithm as *safe*.

one token in total can be sent across any one edge. (This is in keeping with the standard multi-commodity notion that constraints at nodes must be satisfied by each commodity separately, while shared edge capacities must be respected by the commodities cumulatively.)

We show that the natural rate condition on adversaries — essentially obtained by summing Equation (1.1) over the commodities — can be related in a precise sense to the cut condition for standard multi-commodity network flow. As a result, applying well-known results on the cut condition [45, 54, 56], we find that for every $k > 2$, there is a k-commodity adversary of rate $r_k \leq 1$ against which no load balancing algorithm can be safe.

For $k = 2$, however, the cut condition does not pose an obstacle to having algorithms that are safe all the way up to rate 1. Indeed, we are able to generalize our first result to show that for 2-commodity load balancing, there is a simple local-control algorithm that is safe against every adversary of rate 1. We also use the relationship between adversaries and cut conditions to provide a new proof of Hu's Max-Flow Min-Cut Theorem for 2-commodity flow [45]. While our proof is not necessarily shorter than other proofs discovered subsequent to Hu's [56, 77], it is arguably more elementary: it does not require linear programming duality (as in [56]) or even the traditional Max-Flow Min-Cut Theorem for single-commodity flow (as in [77]).

Finally, we further develop the connection between dynamic load balancing and network flows by extending our analysis to packet routing in the adversarial model considered by Aiello et al. [3] and Gamarnik [39]. For a description of this model see Section 2.3. We give an adaptive routing algorithm that is safe against adversaries of rate 1 in the case where packets can be injected at multiple sources but are destined for a single sink; our algorithm is safe in a dynamic network model where edges can appear and disappear. A safe algorithm for this version of the problem was previously given in a recent paper of Awerbuch et al. [10], using a different, but essentially more general, notion of a dynamic network; our proof, a direct adaption of the analysis of our single-commodity load balancing algorithm, is considerably shorter and simpler.

Chapter 2
Background and Related Work
2.1 Basics of Game Theory

In this section we will introduce the reader to the basics of game theory needed for the reading of this thesis. These concepts can be found in any game theory textbook, for example Fudenberg and Tirole [38].

The field of game theory is concerned with the behavior of rational agents in various contexts. Usually the agents are called "players", and the system of their interactions is called a "game". Nevertheless, game theory enfolds a lot more than what people traditionally think of as "games". Any context where the agents have personal goals and are somewhat independent can be put into game theoretic terminology. Bargaining and auctions, as well as the behavior of markets, are an important topic of game theoretic study, for example. Users sharing bandwidth on a network is a classic game theory problem. Assigning various tasks to subordinates so that they are satisfied, or balancing web server loads, are also problems that easily fall into the game theory domain. In general, game theory results apply to any situation where multiple people or agents are deciding on courses of action.

2.1.1 Games in Strategic Form

When talking about games in this thesis, we will represent them using the *strategic* (or *normal*) form. A game in this form can be precisely defined as follows.

Suppose we are given a set of *players* P of size N and for each player $i \in P$ we are given a set S_i of player i's strategies. The set S_i is the set of all possible actions that player i could take. Finally, suppose each player has a personal utility function $u_i : (S_1 \times S_1 \times \ldots \times S_N) \to \mathbb{R}$. This utility function represents how happy player i is with any particular outcome of a game. An *outcome* of a game is a pick of a strategy $s_i \in S_i$ for each player, and $u_i(s_1, s_2, \ldots, s_N)$ is the happiness or the payoff of player i in this situation. An important thing to realize here is that each player is not necessarily trying to make the other players "lose", but instead simply desires to maximize its utility function.

Definition 2.1.1 *A game (in strategic form) is defined by the set P, together with S_i and u_i for each player $i \in P$.*

As defined above, a strategy of player i is an element of the set S_i, and the goal of player i is to maximize its utility. A game consists of each player choosing a strategy. Which strategy should player i choose, then? The problem (from i's perspective) is that its utility function does not just depend on the strategy i chooses, but on the strategies the other players choose as well.

What do the players know when choosing which strategy to pick? Throughout most of this thesis, we will assume that the entire structure of the game is common knowledge. In the case of the Connection Game, this means that the players know the entire network, as well as all the terminals of the other players. In Chapter 6, however, we will assume that the players only know a limited amount of information about the game (and the state of the network) when making their decisions.

In this thesis, we will only consider *pure* strategies, i.e., the strategies available to player i are exactly the strategies in S_i. A common subject of study in game theory is that of *mixed* strategies, where each player can randomize between the strategies in S_i and in essence the strategies available to a player i are anything of the form $(\alpha_1 s_i^1, \alpha_2 s_i^2, \alpha_3 s_i^3 \ldots)$ where all s_i^j's are in S_i and the α_j's add up to 1. Games with mixed strategies have several good properties that games using only pure strategies do not. The use of mixed strategies always guarantees the existence of a Nash equilibrium, for example, while many games with pure strategies do not have a Nash equilibrium at all.

Randomized strategies do not make sense in the contexts of network design which are dealt with in this thesis, however. In the Connection Game, a player's strategy is to purchase some edges, which then become built, leading to the construction of a (hopefully desirable) network. With randomized strategies, a player may pick the edges it is purchasing probabilistically, but eventually it actually has to pick a concrete set of edges to purchase. This is the strategy that the other players see, and respond to. Most importantly, this is the strategy that dictates the payoff of all the players, so we may as well assume that the only strategies available to player i are the pure (non-randomized) ones.

2.1.2 Stability and Nash Equilibria

One of our main goals in this thesis is to compare the solution or the network resulting from interactions of self-interested agents with the optimal solution. To do this in a meaningful manner, we must first define what is the "optimal" solution and what are the solutions "constructed by strategic agents".

Optimal solutions. When considering a centrally optimal solution, there are many objective functions that could be used. For example, what constitutes an "optimal" network in the real world? Is the cheapest network optimal? Or is it the network with small average packet delay? Or is it the one most resilient to edge failures? Recall that an *outcome* of a game is a vector of strategies $(s_1, s_2, \ldots, s_N)$ such that s_i is the strategy player i decides to follow (i.e., $s_i \in S_i$). In the context of general games, which outcome is the optimal one? Once again, there are many possible evaluations of the "goodness" of the outcome. In this thesis, we will consider the optimal outcome as the one that maximizes social welfare, which is the sum of the utilities of all players. In other words, the social welfare of an outcome $(s_1, \ldots, s_N)$ is $\sum_i u_i(s_1, \ldots, s_N)$, and the optimal outcome maximizes this function. In the case of the Connection Game, the network that maximizes social welfare also happens to be the network of least cost, which is a desirable property even if it does not imply high social welfare.

Stable solutions. We now address the question of what constitutes an outcome "formed by strategic agents". Suppose we have an outcome where a player is not happy, and could become happier by changing its strategy. Then this player would change its strategy, and we would end up with a different outcome. The original outcome is "unstable", and so we will not consider such outcomes as ones that can be formed by strategic agents, since if we ended up in such an outcome, one of the agents would immediately change strategies, putting us into a different outcome altogether.

It is worth pointing out here that we are considering non-cooperative players in this thesis. This means that players cannot make deals with one another, or agree to transfer parts of their utility to another player. Consider the opposite case, for example, where all the agents building a network agree to split the profit they get from this network equally. This means that the new utility function for player i is just the old social welfare function divided by N, the number of players. In that case, it is in everybody's interest to build the optimal possible network that maximizes social welfare, and there is no difference here from a centralized designer trying to build the best possible network. Such cooperation is rare in the real world, especially without truly enforceable contracts. Suppose that utility was tied to a particular player, and could not be traded from one to another. How would a player that sees it could improve its personal utility by changing its strategy (and possibly lowering the utility of other players) resist from doing so, especially if this was a one-time game and the players did not need to build a "reputation" with the other players? If the players only care about maximizing their personal utility function, then any outcome where a player could change its strategy and improve its utility, is inherently unstable.

Let us now define what is a stable outcome more rigorously. Suppose at some point, the strategies picked by the players are $(s_1, \ldots, s_N)$. Now consider this situation from the point of

view of player 1. Player 1 has no control over the strategies of other players, but can change its own strategy to anything in S_1. If there exists some strategy $s_1' \in S_1$ such that

$$u_i(s_1', s_2, \ldots, s_N) > u_i(s_1, s_2, \ldots, s_N), \tag{2.1}$$

then by switching its strategy from s_1 to s_1', player 1 could increase its utility. We will assume that the players are *myopic*, i.e., they only look at how their utilities change if they change their strategies while the rest of the players keep theirs the same. If myopic players see that a change of strategy benefits them by increasing their utility given the rest of the players keep their strategies constant, then they would indeed switch strategies. This means that any solution satisfying Inequality 2.1 is unstable, while any outcome that does not satisfy it is stable. Such a stable solution is exactly what is meant by a Nash equilibrium.

Definition 2.1.2 *A Nash equilibrium is an outcome* $(s_1, s_2, \ldots, s_N)$ *such that for all players* i*, all* $s_i' \in S_i$*, we have that* $u_i(s_1', s_2, \ldots, s_N) \leq u_i(s_1, s_2, \ldots, s_N)$*.*

In other words, a Nash equilibrium is a solution such that no single player has any incentive to change its strategy, since there is no possible way that a single player could improve its utility by changing its strategy. These are the only solutions we will consider as the "outcomes of games with myopic players", since any other outcomes will be unstable.

The question still remains of how the players in a game would arrive at a Nash equilibrium. This depends on the exact specification of how a game functions and how players pick their strategies. In a *static* game, the players pick their strategies simultaneously, and cannot change them afterwards. In a *dynamic* game, players can change their strategies after they see what the other players have decided. An important example of game dynamics is *best response dynamics*. In best response dynamics, players take turns changing their strategy. Since the players are myopic, they always change their strategy to the one that maximizes their utility function. In other words, assuming that the current strategies chosen by the players are $(s_1, \ldots, s_N)$, the best response of each player i is $s_i' \in S_i$ such that $u_i(s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_N)$ is maximized. best response dynamics are ones where players take turns switching their strategies to their best response. Notice that this process only terminates exactly if the players end up in a Nash equilibrium.

There are many cases where best response dynamics would take a very long time to arrive at a Nash equilibrium, or would not arrive at one at all. Suppose some independent party calculated a Nash equilibrium, however, and then suggested it to all the players (and perhaps give them some small incentives to take this suggestion). Once the players accept this suggestion, they would be in a stable solution, and would not desire to change their strategy. If the calculation of a Nash equilibrium is possible, therefore, this is often a quick and efficient method to obtain a stable solution. Notice that the players are not being forced to pick this strategy, but instead find it to their advantage to stay in this Nash equilibrium.

In this work, we will consider both the case where the players converge to a Nash equilibrium on their own via some game dynamics, as well as the case where a Nash equilibrium is suggested to them. In either case, we will consider *Nash equilibria* as the outcomes of games of strategic agents.

2.2 Algorithmic Game Theory

In the past decade a new genre of game theory has emerged, known as *algorithmic game theory*. While in the past game theory was mostly the province of economists and mathematicians, recently it has become a hot spot of computer science. As mentioned in the Introduction, most of the work done in traditional computer science focuses on obedient or malicious agents. Agents with rational self-interest which does not necessarily cause them to hurt the network have been more the province of game theoretical research. This has caused theoretical computer scientists to use various methods from game theory to analyze the behavior of these agents. In fact, questions

of this sort have been asked in such different areas as sociology, multi-agent systems [82], and statistical mechanics. The goals and methods of these disciplines have varied dramatically as well. While it is very difficult to separate traditional game theory from algorithmic game theory (nor do we desire such a separation), algorithmic game theory tends to ask questions about the algorithms involved in playing a game and their complexity, as well as using such concepts as approximation algorithms. Since computer science is largely computer-application driven, a large part of algorithmic game theory is concerned with self-interested agents in networks.

Since in this thesis we are concerned specifically with network problems, below we give some background on the way that algorithmic game theory has dealt with network games. For some interesting surveys of algorithmic game theory, see for example [34, 66, 67, 80].

2.2.1 The Price of Anarchy and of Stability

One of the most studied and most important questions addressed in the past is how much is lost by having self-interested players. Usually the comparison is between the centralized optimum that would take place if the players/agents were obedient, and the Nash equilirbia, which are viewed as the outcomes of actions by rational self-interested players. In [67], Papadimitriou introduced the term *the price of anarchy* to denote the ratio between the worst Nash equilibrium and the centralized optimum. This ratio was previously called *the coordination ratio* in work by Koutsoupias and Papadimitriou [52]. There has since been much research on quantifying the price of anarchy for various contexts, including routing [72, 75, 76], load balancing [22, 23, 52, 74], facility location [81], and flow control [4, 26, 78]. In some cases [72, 75] the Nash equilibrium is unique, while in others [52] the best Nash equilibrium coincides with the optimum solution and the authors study the quality of the worst equilibrium. However, in some games the quality of even the best possible equilibria can be far from optimal. The best Nash equilibrium can be viewed as the best solution that self-interested agents can agree upon, i.e. once the solution is agreed upon, the agents do not find it in their interest to deviate. In this thesis, we focus more on studying the *price of stability*, which is the ratio between the *best* Nash equilibrium and the centralized optimum. The price of stability was first studied in [7, 76] (termed the *optimistic price of anarchy* in [7]), and has since received attention mostly in the contexts of network design [6, 51] and routing [6, 19].

How do the players reach a Nash equilibrium, however? As we mentioned above, usually either dynamics converging to a Nash equilibrium are assumed, or it is assumed that the Nash equilibrium is explicitly computed and then implemented by the players. In the first case, there has been much research analyzing best-response dynamics of various network games. It has been shown that while in various games best-response dynamics have good convergence properties (e.g. [28, 59]), nevertheless in many important network games most dynamics do not converge to an equilibrium [36, 37]. In those contexts, it makes more sense to compute a Nash equilibrium, and then implement it by suggesting it to the players. If we are computing a Nash equilibrium instead of letting the players converge to it, then we might as well compute the best Nash equilibrium instead of an arbitrary one. Because of this, the price of stability is much more important than the price of anarchy in many network games.

Computing Nash equilibria may not be easy, however. There has been a lot of work on this as well, in the study of market equilibria [24, 25, 48], as well as routing [30] and more general games [68]. In this thesis we provide results both about the existence of Nash equilibria and about the ability to compute them efficiently.

2.2.2 Mechanism Design and Cost Sharing

Mechanism Design. An important area of game theory is that of mechanism design. Instead of studying the behavior of rational agents, mechanism design is about formulating a game such that the rational outcomes of this game are desirable. For example, while above we were concerned with analyzing the price of stability of certain network games, mechanism design

would instead ask how to design an appropriate network game with low price of stability. Some examples of mechanism design for network games occur in the study of routing [9, 33, 40, 65, 66] and network design [46, 50].

Lately the term "algorithmic mechanism design" has been used mostly in connection with auctions and the VCG (Vickrey-Clarke-Groves) mechanism family. A mechanism in general, however, can be viewed as an arbitrary process for agent interaction and payoffs, usually intended to have some desirable properties (see e.g., [65]). In the study of auctions and mechanisms such as VCG, the focus is often on questions different from the concerns of this thesis, since the main such desirable property is usually truthfulness, and the power of the mechanism to enforce its desires is often very great. This is because it is assumed that the agents have some private information, and the first step to a good solution is finding out what this information is. Even if all the private information is known, however, there is still a great need to design implementable mechanisms which will induce the agents to form good solutions, with desirable properties such as high social welfare, short packet delivery time, or congestion minimization. For example, in the Connection Game, even if we know exactly what pairs of nodes each player would like to connect, it is quite difficult to induce the agents to behave. Moreover, the mechanisms formed in the above literature generally require a large amount of power to enforce, and this enforcement is done in a centralized manner. The VCG family of mechanisms, for example, requires a large amount of side payments from the central authority to the agents to achieve its goal of truthfulness. We require that the mechanisms be implementable in a distributed manner, and that the power wielded by the mechanism is very limited with respect to that of the agents. Mechanism design usually assumes that "the designer" has a lot of control over a game and the interactions and strategies of its players. In the cases of the Connection Game, routing in networks, and the Internet, however, the power lies with the players. They decide what the interactions between them are, and what the utility functions are. It would be wonderful if we could design a mechanism for the interactions on the Internet, and then implement it. It would be even better if we could design a wonderful social system for a large country and implement that as well. A mechanism which simply orders the agents to conform to a particular outcome, however, would not be acceptable, since we are assuming that the mechanism does not have enough power to enforce this or enough knowledge to detect that an agent disobeys it. If a mechanism suggests a particular outcome to the agents, therefore, it must be an outcome that actually makes all the agents happy. Furthermore, we focus on the cases where the agents have no private values, or where it is reasonable to assume that the values are generally known. Once we have a greater understanding of these cases, adding truthfulness to the framework would be an interesting direction.

Cost Sharing. One of the largest (and most relevant to this thesis) areas of mechanism design in networks is that of *cost sharing*. In cost sharing there is usually some network or a set of resources that is desired by a set of players. As in all mechanism design, the idea is to design a game that has some good properties, for example: truthfulness (the players would never try to conceal their utility function if they could), budget balance (the players pay for the entire network), and stability (the solutions that have other good properties are also stable).

The bulk of the work on network cost-sharing (see e.g. [32, 44] and the references there) tends to assume a fixed underlying set of edges. Jain and Vazirani [46] and Kent and Skorin-Kapov [50] consider cost-sharing for a single source network design game closely related to the Connection Game. They assume that each player i has a utility u_i for belonging to the Steiner tree (the maximum they would be willing to pay). Cost-sharing games assume that there is a central authority that designs and maintains the network, and decides appropriate cost-shares for each agent, depending on the graph and all other agents, via a complex algorithm. The agents' only role is to report their utility for being included in the network. The goal of the cost-sharing method is to pay for the network built, and produce cost-shares such that selfish agents are truthful, i.e. do not find it in their interest to misreport their utility (in hopes of being included in the network at a smaller cost). Jain and Vazirani give a truthful mechanism to share the cost of the minimum spanning tree, which is a 2-approximation for the Steiner tree problem.

The budget balanced cost-sharing game studied in [46] has some similarities with our model, and the differences are worth noting. Cost-sharing games assume that there is a central authority that designs and maintains the network, and decides appropriate cost-shares for each agent. In contrast, in our game the agents contribute to individual edges directly, rather than contributing money to a central authority. Therefore, each player is free to choose its own paths, instead of having the central authority specify paths for each player. In our game there is no central authority designing the Steiner tree or cost shares. Rather, we study Nash equilibria of our game. A game with low price of stability is one where any stable outcome of selfish interactions of agents is of low cost, without any help from a central authority. In our model, the edges to be used by the agents are paid for by them, and thus the notions of truthfulness and budget-balance become redundant. We also feel our model is more representative of real networks because the possible strategies of each agent are not restricted to any particular subset of edges.

The use of the price of stability as a measure of quality can be viewed as being half way between the traditional price of anarchy and cost-sharing. Selecting the best Nash equilibrium effectively assumes some coordination: a central authority that guides the players to the best Nash equilibrium. However, once the players settle on such an equilibrium strategy, it is self enforcing: no player finds it in his/her interest to deviate. The role of the central authority is limited to guiding the players to a better Nash equilibrium.

In Chapter 4 we face some cost-sharing concerns. A cost-sharing mechanism can be viewed as the underlying protocol that determines how much a network serving several participants will cost to each of them. We decide in Chapter 4 that any players using an edge must split its cost evenly, which is a type of cost-sharing constraint. Recall that cost-sharing games assume that there is a central authority that designs and maintains the network, and decides appropriate cost-shares for each agent, depending on the graph and all other agents, via a complex algorithm. The agents' only role is to report their utility for being included in the network. Here, on the other hand, we consider a simple cost-sharing mechanism, the Shapley-value, and ask what the strategic implications of a given cost-sharing mechanism are for the way in which a network will be designed. This question explores the feedback between the protocol that governs network construction and the behavior of self-interested agents that interact with this protocol. An approach of a similar style, though in a different setting related to routing, was pursued by Johari and Tsitsiklis [47]; there, they assumed a network protocol that priced traffic according to a scheme due to Kelly [49], and asked how this protocol would affect the strategic decisions of self-interested agents routing connections in the network.

This Shapley-value cost-sharing mechanism, though, is only one of many possible. We choose this particular equal-division mechanism to study because it has a number of basic economic motivations; it can be derived from the Shapley value [63], and it can be shown to be the unique cost-sharing scheme satisfying a number of different sets of axioms [32, 44, 63]. In theory, there are many other cost-sharing constraints we could use instead, some of which would guarantee much better player behavior. These constraints and protocols may be too powerful and unenforcible, however. Any useful mechanisms in this context must be fully distributed, use only local information, and use only limited power, since while a mechanism could be designed which guaranteed centralized optimal behavior, the cost-sharing must be done quickly and locally, and therefore on an edge-by-edge basis. In essence, there is a tradeoff here between the power available to the mechanism, and the quality of the solutions it can produce, and in settings where the agents have most of the power, we must consider only a limited space of possible mechanisms, unlike most mechanism design and cost sharing.

2.2.3 Selfish Routing

The most relevant body of work to this thesis (especially to Chapter 5) is the work on selfish (i.e. self-interested) routing done largely by Roughgarden and Tardos [72, 73, 75]. In their model, they assume a graph with latency functions on each edge, and some traffic demands in this graph. There, if there is a demand of 1 from node u to node v, it will route itself so that every part

of this demand achieves minimal latency. For example, suppose that there is a simple network with two nodes and two edges between these nodes, one with latency function $1 - x$ (where x is the amount of traffic on this edge), and one with the constant latency function $1/2$. If there is one unit of traffic that is trying to route itself from u to v, then it will split itself so that $1/2$ goes on each edge, since then the latencies on both edges would be the same. We can think of this unit of traffic as consisting of an infinite number of infinitesimal players, each of whom try choose the route to minimize their latency. The routing of traffic such that from the perspective of each demand, the latencies are equal, exactly corresponds to a Nash equilibrium.

In the model from [73], the Nash equilibrium of any such routing game is essentially unique, so the price of stability and the price of anarchy are the same. Roughgarden and Tardos have done a lot of work in bounding the price of anarchy in this routing model, which we will address in more detail in Chapter 5. Roughgarden [72] has also shown that the price of anarchy is independent of the network topology in this context. Specifically, he has shown that the worst-case price of anarchy always occurs on a simple two-node two-link graph.

This model is quite relevant to the Connection Game, even though this is a routing model, and the Connection Game is modeling network creation. There is a similar feel to the problem, however, since players are still choosing paths connecting their terminals. The fact that the edges have costs instead of latency functions, and therefore players prefer to share edges with others (to take care of extra cost) instead of preferring to use the edge alone (to not incur extra latency) is what makes the problem completely different in both feel and the type of results obtained.

In Chapter 5 we look at what happens if the "cost functions" on the edges are arbitrary, or if they are latencies instead. The special case of our game with only delays/latencies is closely related to the congestion game of [72, 73, 75]. They consider a game where the amount of flow carried by an individual user is infinitesimally small (a *non-atomic game*), while in this thesis we assume that each user has a unit of flow, which it needs to route on a single path. In the non-atomic game of [72, 73, 75] the Nash equilibrium is essentially unique (hence there is no distinction between the price of anarchy and stability), while in our atomic game there can be many equilibria, and the price of anarchy can become quite large. Fabrikant, Papadimitriou, and Talwar [30] consider our atomic game with delays only. They give a polynomial time algorithm to find a Nash equilibrium in the case that all users share a common source, and show that finding any equilibrium solution is PLS-complete for multiple source-sink pairs. In Chapter 5 we are able to show, however, that bounds on the price of anarchy for the non-atomic selfish routing model in [72, 73, 75] still hold for our model of single-source atomic games, but they become bounds on the price of stability instead.

2.2.4 More Related Work

We view our Connection Game as a simple model of how different service providers build and maintain the Internet or local network topologies. We use a game theoretic version of network design problems considered in approximation algorithms [42]. Fabrikant et al [29] study a different network creation game. Network games similar to that of [29] have also been studied for modeling the creation and maintenance of social networks [13, 43]. In the network game considered in [13, 29, 43] each agent corresponds to a single node of the network, and agents can only buy edges adjacent to their nodes. This model of network creation seems extremely well suited for modeling the creation of social networks. However, in the context of communication networks like the Internet, agents are not directly associated with individual nodes, and can build or be responsible for more complex networks. There are many situations where agents will find it in their interest to *share* the costs of certain expensive edges. An interesting feature of our model which does not appear in [13, 29, 43] is that we allow agents to share costs in this manner. To keep our model simple, in Chapter 3, we assume that each agent's goal is to keep its terminals connected, and agents are not sensitive to the length of the connecting path (although in Chapter 4 we assume a variety of goals). In contrast, [13, 29, 43] assume that longer paths are associated with decreased benefit, and as a result also model a form of quality of service.

A weighted game similar to our game in Section 4.5 is presented by Libman and Orda [55], with a different mechanism for distributing costs among users. They do not consider the price of stability, and instead focus on convergence in parallel networks. Recently, Chen and Roughgarden [18] proved general results on the price of stability and the existence of approximate Nash equilibria in the weighted game we present. They show that there exist weighted games with no pure Nash equilibria, settling an open question from [6]. They also prove that any weighted game has a $O(\log w_{\max})$-approximate Nash equilibrium that costs at most $O(\log W)$ times the centralized optimum (where $w_{\max}$ is the maximum weight, and W is the sum of all player weights). Their results show that this upper bound is essentially tight.

2.3 Dynamic Load Balancing and Packet Routing

In Chapter 6 we consider a network with simple agents that are trying to locally balance the load in the network. The load is being added by an adversary. We show that in this scenario (and in a similar packet routing context) a simple local protocol has good global properties. Specifically, we show that a simple local balancing protocol keeps the load approximately balanced.

In a distributed network of computing hosts, the performance of the system can depend crucially on dividing up work effectively across the participating nodes. This type of *load balancing problem* has been studied in many different models, centered around the idea that an algorithm should avoid creating "hot spots" that degrade system performance [79]. Moreover, it is of interest to understand what can be accomplished in this setting without centralized control, by mechanisms in which nodes simply monitor the state of their neighbors and behave according to a simple local rule.

We consider a basic model of load balancing in a distributed network as described in the Introduction, which has formed the basis of a number of earlier studies [2, 41, 58, 64, 69]. A network of identical processors is represented by an undirected graph $G = (V, E)$. There are a number of *jobs* to be processed in the system, abstractly represented by unit-size *tokens*. Time progresses in discrete steps called *rounds*; in a given round, each token is held by one of the nodes, which is viewed as processing the associated job, and the *load* on a node is defined to be the number of tokens it holds. The goal is to *balance* the loads, so that no single node has too many tokens; this can be accomplished by transmitting tokens between neighboring nodes of the graph, at a rate of one token per edge per round. We are particularly interested in *local* algorithms for this problem: rather than using a centralized approach to coordinate the movement of tokens, each node will simply compare its load to those of its neighbors, and decide whether to move a token across an edge based on this information.

Early work on the model focused on the *static* version of the problem: each node is given a set of tokens initially, and nodes must exchange tokens as rapidly as possible so that they all end up with approximately the same number [2, 41, 58, 69]. However, load balancing is a natural setting in which to study algorithms designed to run indefinitely; jobs (tokens) may arrive and depart from the network, and at all times the algorithm must maintain an approximately uniform load across nodes. This is a type of *stability* condition: no load should diverge arbitrarily from the average as time progresses. For a number of different models, such *dynamic* algorithms have been studied in a probabilistic framework, where one assumes an underlying randomized process that generates job arrivals and departures; see e.g. [27, 60] and the references therein.

Motivated by work in the related area of packet routing [5, 11, 12, 16], Muthukrishnan and Rajaraman proposed an *adversarial* framework for studying dynamic load balancing in the token-based model we have been discussing [64]. Rather than considering a probabilistic process that generates tokens, they posit an *adversary* that is allowed at the beginning of each round to introduce tokens at some nodes (corresponding to new jobs) and remove tokens from others (corresponding to jobs that have finished). Subsequently, an algorithm is allowed to move tokens across edges as above so as to try to maintain balanced loads. This alternation of moves by the adversary and algorithm continues for an arbitrary number of rounds. Note that by allowing the

adversary to control the removal of tokens as well as their arrival, one is modeling a worst-case assumption that jobs may have arbitrary duration, and the algorithm does not know how much processing time a job has remaining until the moment it ends.

As described in the Introduction, if we let a_t denote the average number of tokens per node in the system at the beginning of round t, and $h_t(v)$ denote the number of tokens at node v (the *height* of v) at round t, then the goal of a dynamic load balancing algorithm in this model is to keep $h_t(v)$ close to a_t for all nodes v and rounds t. Formally, we say that an algorithm is *safe* [1] against a given adversary if there is a constant B such that $|h_t(v) - a_t| \leq B$ for all nodes v and rounds t. Note that safety in this context imposes a bound on deviation from the average; it is not required that the actual number of tokens in the system remains bounded.

As in the case of packet routing [16, 5], one needs to find a suitable restriction on the adversary: an arbitrarily powerful adversary could flood a particular set of nodes $S \subseteq V$ with tokens much faster than these nodes can spread the tokens out to the rest of the graph, and thereby prevent any algorithm from maintaining safety. This consideration motivates the following natural definition of an adversary [64] with *rate* r. For a set $S \subseteq V$, let $e(S)$ denote the set of edges with exactly one end in S, and $\delta_t(S)$ the net increase in tokens in set S due to the addition and removal of jobs in round t (note that $\delta_t(S)$ could be negative). If the heights of nodes in S were to change precisely according to average, then the net change in tokens in S would be $|S| \cdot (a_{t+1} - a_t)$. One wants the difference between these two quantities to be "accounted for" by the edges in $e(S)$. We say that the adversary has *rate* r if for all $S \subseteq V$, one has

$$|\delta_t(S) - |S|(a_{t+1} - a_t)| \quad \leq \quad r \cdot |e(S)|, \tag{2.2}$$

For rate $r > 1$, there are adversaries against which no algorithm (whether online or offline) can be safe. Muthukrishnan and Rajaraman gave a local-control algorithm that is safe against all adversaries of rate r, for every $r < 1$. As an open question, they asked whether there exists a local-control algorithm that is safe against all adversaries of rate 1.

In Chapter 6, we answer this question positively by proving that a simple local algorithm is safe against the above adversaries (see Section 1.5 for a thorough description of our other results for this model). This type of algorithm was considered in earlier work on the static model by Aiello et al. [2], as well as by many of the subsequent papers, including [64]. Compared to [64], however, we obtain significantly improved bounds on the deviation from the average, and a simpler proof. Specifically, we show that the maximum possible deviation from the average is $O(\Delta n)$, where n is the number of nodes of G, and this is asymptotically optimal in the worst case; the analysis in [64] had established a bound of $O(\Delta^2 n^{2.5}(1-r)^{-1})$ when $r < 1$.

Following this, we use the relationship between adversaries and cut conditions to provide a new proof of Hu's Max-Flow Min-Cut Theorem for 2-commodity flow [45]. While our proof is not necessarily shorter than other proofs discovered subsequent to Hu's [56, 77], it is arguably more elementary: it does not require linear programming duality (as in [56]) or even the traditional Max-Flow Min-Cut Theorem for single-commodity flow (as in [77]).

We further develop the connection between dynamic load balancing and network flows by extending our analysis to packet routing in the adversarial model considered by Aiello et al. [3] and Gamarnik [39]. In this model, packets are adversarially inserted into a network at various nodes, with the adversary specifying their destinations. Each node (representing a router) can send packets along edges connecting to it, but it must respect the capacity constraints of the edges. This is very similar to the above load balancing model, with tokens now being packets instead of jobs. The goal of the decentralized algorithm in the packet routing model is to get all the packets to their destinations without having too many packets at any single router (which would overload the queue of that router). In the model from [3] and [39], the usual constraint on the adversary is that, at each step, it can insert packets as long as there exists a flow that can get r times this number of packets to their destinations, and that respects the capacity constraints.

[1] In most literature and in [8] such an algorithm is called *stable*. To avoid confusion with the stability of Nash equilibria, in this thesis we will refer to such an algorithm as *safe*.

In other words, if an adversary inserts certain packets at at particular step, then such a flow must exist even if we replace each packet inserted with r packets instead. r is known as the rate of the adversary.

We give an adaptive routing algorithm that is safe against adversaries of rate 1 in the case where packets can be injected at multiple sources but are destined for a single sink; our algorithm is safe in a dynamic network model where edges can appear and disappear. A safe algorithm for this version of the problem was previously given in a paper of Awerbuch et al. [10], using a different, but essentially more general, notion of a dynamic network; our proof, a direct adaptation of the analysis of our single-commodity load balancing algorithm, is considerably shorter and simpler.

Chapter 3
The Connection Game with Arbitrary Cost Sharing

In this chapter, we examine a game where players try to connect their terminals in a network by purchasing links, and are able to share the cost of these links in arbitrary ways. Many of the results in this chapter initially appeared in [7].

In Chapter 4 we will also discuss a similar game, with the constraint that players must share the cost of the links they use in a fair manner. This sharing rule greatly changes the properties of the system, for both better and worse. See the beginning of Chapter 4 for a comparison of some of these properties.

3.1 Model and Basic Results

The Connection Game We now formally define the connection game for N players. Let an undirected graph $G = (V, E)$ be given, with each edge e having a nonnegative cost $c(e)$. Notice that unlike in later chapters of this thesis, in this chapter the cost of an edge is always a constant. Each player i has a set of terminal nodes that he must connect. The terminals of different players do not have to be distinct. A strategy of a player is a payment function p_i, where $p_i(e)$ is how much player i is offering to contribute to the cost of edge e. Any edge e such that $\sum_i p_i(e) \geq c(e)$ is considered *bought*, and G_p denotes the graph of bought edges with the players offering payments $p = (p_1, \ldots, p_N)$. Since each player must connect his terminals, all of the player's terminals must be connected in G_p. However, each player tries to minimize his total payments, $\sum_{e \in E} p_i(e)$.

A Nash equilibrium of the connection game is a payment function p such that, if players offer payments p, no player has an incentive to deviate from his payments. This is equivalent to saying that if p_j for all $j \neq i$ are fixed, then p_i minimizes the payments of player i. A $(1+\varepsilon)$-approximate Nash equilibrium is a function p such that no player i could decrease his payments by more than a factor of $1 + \varepsilon$ by deviating, i.e. by using a different payment function $p_i{}'$.

Some Properties of Nash Equilibria Here we present several useful properties of Nash equilibria in the Connection Game. Suppose we have a Nash equilibrium p, and let T^i be the smallest tree in G_p connecting all terminals of player i. It follows from the definitions that (1) G_p is a forest, (2) each player i only contributes to costs of edges on T^i, and (3) each edge is either paid for fully or not at all.

Property 1 holds because if there was a cycle in G_p, any player i paying for any edge of the cycle could stop paying for that edge and decrease his payments while his terminals would still remain connected in the new graph of bought edges. Similarly, Property 2 holds since if player i contributed to an edge e which is not in T^i, then he could take away his payment for e and decrease his total costs while all his terminals would still remain connected. Property 3 is true because if i was paying something for e such that $\sum_i p_i(e) > c_e$ or $c_e > \sum_i p_i(e) > 0$, then i could take away part of his payment for e and not change the graph of bought edges at all.

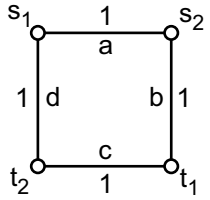

Figure 3.1: A game with no Nash equilibria.

Nash Equilibria May Not Exist It is not always the case that selfish agents can agree to pay for a network. There are instances of the connection game which have no deterministic Nash equilibria. In Figure 3.1, there are 2 players, one wishing to connect node s_1 to node t_1, and the other s_2 to t_2. Now suppose that there exists a Nash equilibrium p. By Property 1 above, in a Nash equilibrium G_p must be a forest, so assume without loss of generality it consists of the edges a, b, and c. By Property 2, player 1 only contributes to edges a and b, and player 2 only contributes to edges b and c. This means that edges a and c must be bought fully by players 1 and 2, respectively. At least one of the two players must contribute a positive amount to edge b. However, neither player can do that in a Nash equilibrium, since then he would have an incentive to switch to the strategy of only buying edge d and nothing else, which would connect his terminals with the player's total payments being only 1. Therefore, no Nash equilibria exist in this example.

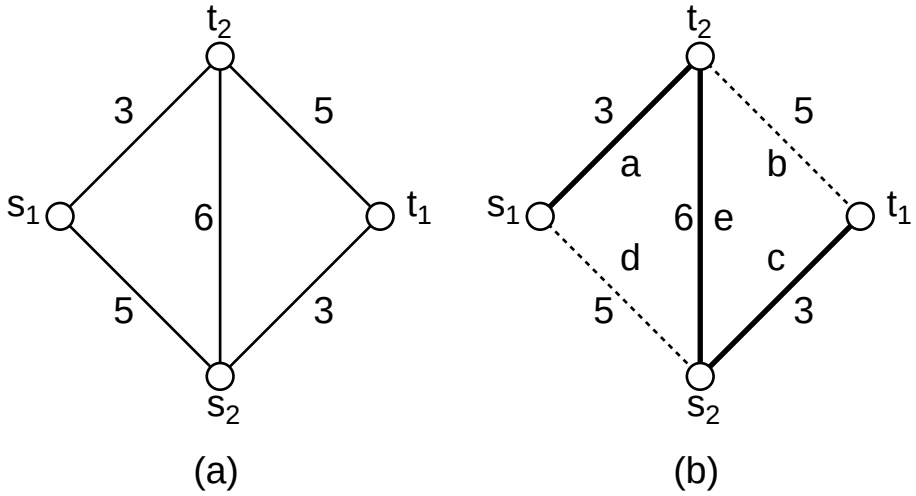

Figure 3.2: A game with only fractional Nash equilibria

Fractional Nash Equilibria When looking at the connection game, we might be tempted to assume that giving players the opportunity to share costs of edges is an unnecessary complication. However, sometimes players must share costs of edges for all players to agree on a network. There are game instances where the only Nash equilibria in existence require that players split the cost of an edge. We will call such Nash equilibria *fractional* and we will call Nash equilibria that do not involve players sharing costs of edges *non-fractional*.

Figure 3.2(a) is an example of a connection game instance where the only Nash equilibria are fractional ones. Once again, player 1 would like to connect s_1 and t_1, and player 2 would like to connect s_2 and t_2. First, note that there is a fractional Nash equilibrium, as shown in Figure 3.2(b). In this case we have that player 2 contributes 5 to edge e and player 1 contributes 1 to e and 3 to both of a and c. It is easy to confirm that neither player has an incentive to deviate.

Now we must show that there are no non-fractional Nash equilibria in this example. Observe that if edge e is not bought, then we have a graph which is effectively equivalent to the graph in which we showed there to be no Nash equilibria at all. Therefore any non-fractional Nash equilibria must buy edge e. Given that edge e must be bought, it is clear that player 2 will only contribute to edge e. For a Nash equilibrium p to be non-fractional, this would mean that player 2 either buys edge e fully or buys nothing at all. Suppose player 2 buys e. The only response for which player 1 would not want to deviate would be to buy a and c. But then player 2 has an incentive to switch to either edge b or d. Now suppose player 2 does not buy e. Then the only response for which player 1 would not want to deviate would be to either buy a and b or buy c

and d. Either way, player 2 does not succeed in joining his source to his sink, and thus has an incentive to buy an edge. Hence, there are no non-fractional Nash equilibria in this graph.

The Price of Anarchy We have now shown that Nash equilibria do not have to exist. However, when they exist, how bad can these Nash equilibria be? As mentioned above, the price of anarchy often refers to the ratio of the worst (most expensive) Nash equilibrium and the optimal centralized solution. In the connection game, the price of anarchy is at most N, the number of players. This is simply because if the worst Nash equilibrium p costs more than N times OPT, the cost of the optimal solution, then there must be a player whose payments in p are strictly more than OPT, so he could deviate by purchasing the entire optimal solution by himself, and connect his terminals with smaller payments than before. More importantly, there are cases when the price of anarchy actually equals N, so the above bound is tight. This is demonstrated with the example in Figure 3.3. Suppose there are N players, and G consists of nodes s and t which are joined by 2 disjoint paths, one of length 1 and and one of length N. Each player has a terminal at s and t. Then, the worst Nash equilibrium has each player contributing 1 to the long path, and has a cost of N. The optimal solution here has a cost of only 1, so the price of anarchy is N. Therefore, the price of anarchy could be very high in the connection game. However, notice that in this example the *best* Nash equilibrium (which is each player buying $\frac{1}{N}$ of the short path) has the same cost as the optimal centralized solution. We have now shown that the price of anarchy can be very large in the connection game, but the price of stability remains worth considering, since the above example shows that it can differ from the price of anarchy by as much as a factor of N.

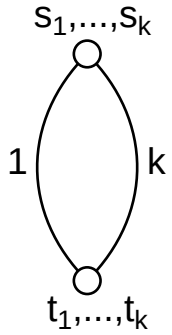

Figure 3.3: A game with price of anarchy of N

All the results in this section also hold if G is directed or if each player i has a maximum cost $\max(i)$ beyond which he would rather pay nothing and not connect his terminals.

3.2 Single Source Games

As we show in Section 3.4, determining whether or not Nash equilibria exist in a general instance of the connection game is NP-Hard. Furthermore, even when equilibria exist, they may be significantly more expensive than the centrally optimal network. In this section we define a class of games in which there is always a Nash equilibrium, and the price of stability is 1. Furthermore, we show how we can use an approximation to the centrally optimal network to construct a $(1+\epsilon)$-approximate Nash equilibrium in poly-time, for any $\epsilon > 0$.

Definition 3.2.1 *A* single source game *is a game in which all players share a common terminal s, and in addition, each player i has exactly one other terminal t_i.*

We will now show that the price of stability is 1 in single source games. To do this, we must argue that there is a Nash equilibrium that purchases T^*, the minimum cost Steiner tree on the

players' terminal nodes. There are a number of standard cost-sharing methods for sharing the cost of a tree among the terminals. The two most commonly studied methods are the Shapley value and the Marginal Cost mechanisms [32]. The Marginal Cost (or VCG) mechanisms are very far from being budget balanced, i.e. the agents do not pay for even a constant fraction of the tree built. The Shapley value mechanism is budget balanced: the cost of each edge is evenly shared by the terminals that use the edge for their connection (i.e., the terminals in the subtree below the edge e). However, the mechanism does not lead to a Nash equilibrium in our game: some players can have cheaper alternate paths, and hence benefit by deviating. Jain and Vazirani [46] give a truthful budget balanced cost-sharing mechanism to pay for the minimum spanning tree, which is a 2-approximate budget balanced mechanism for the Steiner tree problem. However, it is only a 2-approximation, and the cost-shares are not associated with edges that the agents use. Here we will show that while the traditional Steiner tree cost-sharing methods do not lead to a Nash equilibrium, such a solution can be obtained.

Theorem 3.2.2 *In any single source game, there is a Nash equilibrium which purchases T^*, a minimum cost Steiner tree on all players' terminal nodes.*

Proof. Given T^*, we present an algorithm to construct payment strategies p. We will view T^* as being rooted at s. Let T_e be the subtree of T^* disconnected from s when e is removed. We will determine payments to edges by considering edges in reverse breadth first search order. We determine payments to the subtree T_e before we consider edge e. In selecting the payment of agent i to edge e we consider c', the cost that player i faces if he deviates in the final solution: edges f in the subtree T_e are considered to cost $p_i(f)$, edges f not in T^* cost $c(f)$, while all other edges cost 0. We never allow i to contribute so much to e that his total payments exceed his cost of connecting t_i to s.

Algorithm 3.2.3 Initialize $p_i(e) = 0$ for all players i and edges e.
Loop through all edges e in T^* in reverse BFS order.
  Loop through all players i with $t_i \in T_e$ until e paid for.
    If e is a cut in G set $p_i(e) = c(e)$.
    Otherwise
      Define $c'(f) = p_i(f)$ for all $f \in T^*$ and
             $c'(f) = c(f)$ for all $f \notin T^*$.
      Define χ_i to be the cost of the cheapest path from s to
        t_i in $G \setminus \{e\}$ under modified costs c'.
      Define $p_i(T^*) = \sum_{f \in T^*} p_i(f)$.
      Define $p(e) = \sum_j p_j(e)$.
      Set $p_i(e) = \min\{\chi_i - p_i(T^*), c(e) - p(e)\}$.
    end
  end
end

We first claim that if this algorithm terminates, the resulting payment forms a Nash equilibrium. Consider the algorithm at some stage where we are determining i's payment to e. The cost function c' is defined to reflect the costs player i faces if he deviates in the final solution. We never allow i to contribute so much to e that his total payments exceed his cost of connecting t_i to s. Therefore it is never in player i's interest to deviate. Since this is true for all players, p is a Nash equilibrium.

We will now prove that this algorithm succeeds in paying for T^*. In particular, we need to show that for any edge e, the players with terminals in T_e will be willing to pay for e. Assume the players are unwilling to buy an edge e. Then each player has some path which explains why it can't contribute more to e. We can use a carefully selected subset of these paths to modify T^*, forming a cheaper tree that spans all terminals and doesn't contain e. This would clearly contradict our assumption that T^* had minimum cost.

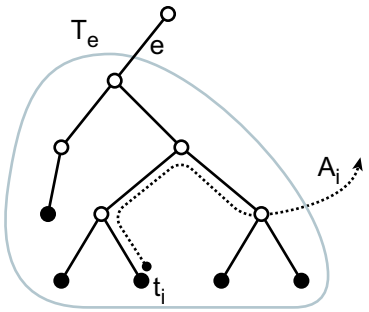

Figure 3.4: Alternate paths in single source games.

Define player i's *alternate path* A_i to be the path of cost χ_i found in Algorithm 3.2.3, as shown in Figure 3.4. If there is more than one such path, choose A_i to be the path which includes as many ancestors of t_i in T_e as possible before including edges outside of T^*. To show that all edges in T^* are paid for, we need the following technical lemma concerning the structure of alternate paths.

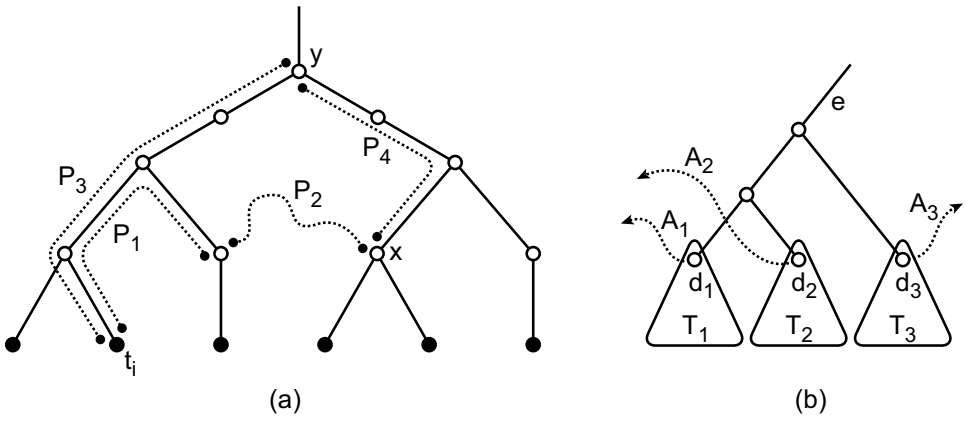

Figure 3.5: Alternate path structure in the proof of Theorem 3.2.2.

Lemma 3.2.4 *Suppose A_i is i's alternate path at some stage of the algorithm. Then there are two nodes v and w on A_i, such that all edges on A_i from t_i to v are in T_e, all edges between v and w are in $E \setminus T^*$, and all edges between w and s are in $T^* \setminus T_e$.*

Proof. Once A_i reaches a node w in $T^* \setminus T_e$, all subsequent nodes of A_i will be in $T^* \setminus T_e$, as all edges f in $T^* \setminus T_e$ have cost $c'(f) = 0$ and the source s is in $T^* \setminus T_e$. Thus, suppose A_i begins with a path P_1 in T_e, followed by a path P_2 containing only edges not in T^*, before reaching a node x in T_e, as shown in Figure 3.5(a). Let y be the lowest common ancestor of x and t_i in T_e. Observe that P_1 is strictly below y. Define P_3 to be the path from t_i to y in T_e, and define P_4 to be the path from y to x in T_e. We now show that under the modified cost function c', $P_3 \cup P_4$ is at least as cheap as $P_1 \cup P_2$. Since $P_1 \cup P_2$ includes a higher ancestor of t_i than A_i (namely y), this contradicts our choice of A_i.

Consider the iterations of the algorithm during which player i could have contributed to edges in P_3. At each of these steps the algorithm computes a cheapest path from t_i to s. At any time, player i's payments are upper bounded by the modified cost of his alternate path, which is in turn upper bounded by the modified cost of any path, in particular A_i. Furthermore, at each of these steps the modified costs of all edges in A_i above x are 0. Therefore i's contribution

to P_3 is always at most the modified cost of $P_1 \cup P_2$. The modified cost of P_4 is always 0, as none of the edges in P_4 are on player i's path from t_i to s in T^*. Together these imply that $c'(P_3 \cup P_4) = c'(P_3) \leq c'(P_1 \cup P_2)$. □

Thus, players' alternate paths may initially use some edges in T_e, but subsequently will exclusively use edges outside of T_e. We use this fact in the following lemma.

Lemma 3.2.5 *Algorithm 3.2.3 fully pays for every edge in T^*.*

Proof. Suppose that for some edge e, after all players have contributed to e, $p(e) < c(e)$.

For each player i, consider the longest subpath of A_i containing t_i and only edges in T_e. Call the highest ancestor of t_i on this subpath i's *deviation point*, denoted d_i. Note that it is possible that $d_i = t_i$. Let D be a minimum set of deviation points such that every terminal in T_e has an ancestor in D.

Suppose every player i with a terminal t_i in D deviates to A_i, as shown in Figure 3.5(b), paying his modified costs to each edge. Any player i deviating in this manner does not increase his total expenditure, as player i raised $p_i(e)$ until p_i matched the modified cost of A_i. The remaining players leave their payments unchanged.

We claim that now the edges bought by players with terminals in T_e connect all these players to $T^* \setminus T_e$. To see this, first consider any edge f below a deviation point d_i in D. By Lemma 3.2.4, player i is the only deviating player who could have been contributing to f. If i did contribute to f, then f must be on the unique path from t_i to d_i in T_e. But by the definition of d_i, this means that f is in A_i. Thus player i will not change his payment to f.

Define T_i to be the subtree of T_e rooted at d_i. We have shown that all edges in T_i have been bought. By Lemma 3.2.4, we know that A_i consists of edges in T_i followed by edges in $E \setminus T$ followed by edges in $T^* \setminus T_e$. By the definition of c', the modified cost of those edges in $E \setminus T^*$ is their actual cost. Thus i pays fully for a path connecting T_i to $T^* \setminus T_e$.

We have assumed that the payments generated by the algorithm for players with terminals in T_e were not sufficient to pay for those terminals to connect to $T^* \setminus T_e$. However, without increasing any players' payments, we have managed to buy a subset of edges which connects all terminals in T_e to $T^* \setminus T_e$. This contradicts the optimality of T^*. Thus the algorithm runs to completion. □

Since we have also shown that the algorithm always produces a Nash equilibrium, this concludes the proof of the theorem. □

We will now argue that Algorithm 3.2.3 works even if the graph is directed. It is still the case that if the algorithm does succeed in assigning payments to all edges, then we are done. Hence, to prove correctness, we will again need only show that failure to pay for an edge implies the existence of a cheaper tree, thus yielding a contradiction. The problem is that Lemma 3.2.4 no longer holds; it is possible that some of the players attempting to purchase an edge e have an alternate paths which repeatedly moves in and out of the subtree T_e. Thus, the argument is slightly more complex.

Lemma 3.2.6 *Algorithm 3.2.3 fully pays for every edge in T^* for directed graphs.*

Proof. Suppose the algorithm fails to pay for some edge e. At this point, every player i with a terminal in T_e has an alternate path A_i, as defined earlier. Define D to be the set of vertices contained in both T_e and at least one alternate path. Note that D contains all terminals that appear in T_e. We now create $D' \subseteq D$ by selecting the *highest* elements of D; we select the set of nodes from D that do not have ancestors with respect to T_e in D. Every terminal in T_e has a unique ancestor in D' with respect to T_e, and every node in D' can be associated with at least one alternate path.

For any node $v_j \in D'$, let A_j be the alternate path containing v_j. If more than one such path exists, simply select one of them. Define A'_j to be the portion of this path from v_j to the first node in $T \setminus T_e$. We can now form T' as the union of edges from $T \setminus T_e$, all paths A'_j, and every

subtree of T_e rooted at a node in D'. T' might not be a tree, but breaking any cycles yields a tree which is only cheaper.

It is clear that all terminals are connected to the root in T', since every terminal in T_e is connected to some node in D', which in turn is connected to $T \setminus T_e$. Now we just need to prove that the cost of our new tree is less than the cost of the original. To do so, we will show that the total cost of the subtrees below nodes in D', together with the cost of adding any additional edges needed by the paths A'_j, is no greater than the total payments assigned by the algorithm to the players in T_e thus far. Hence it will be helpful if we continue to view the new tree as being paid for by the players. In particular, we will assume that all players maintain their original payments for all edges below nodes in D', and the additional cost of building any path A'_j is covered by the player for which A_j was an alternate path. It now suffices to show that no player increases their payment.

For the case of those players who are not associated with a node from D', this trivially holds, since their new payments are just a subset of their original payments. Now consider a player i who must pay for any unbought edges in the path A'_j, which starts from node $v_j \in D'$. Note that player i's terminal might not be contained within the subtree rooted at v_j. If it is, then we are done, since in this case, player i's new cost is at most the cost of A_j, which is exactly i's current payment.

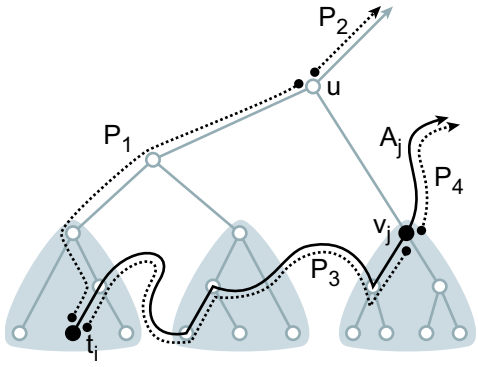

Figure 3.6: Alternate path structure in the proof of Lemma 3.2.6.

Thus suppose instead that player i's terminal lies in a subtree rooted at a node $v_k \in D'$ (this case is shown in Figure 3.6). Define u to be the least common ancestor of v_k and v_j in T_e. Observe that u can not be either v_k or v_j, as this would contradict the minimality of the set D'. Define P_1 to be the current payments made by player i from its terminal to u, and let P_2 be the current payments made by player i from u to e (inclusive). Define P_3 as the cost of $A_j \setminus A'_j$ and let P_4 be the cost of A'_j. By the definition of alternate path,

$$P_1 + P_2 = P_3 + P_4.$$

Furthermore, since we have already successfully paid for a connection to u, we know that

$$P_3 \geq P_1,$$

since otherwise, when we were paying for the edges between v_k and u, player i would have had an incentive to deviate by purchasing P_3 and then using the path from v_j to u in T_e, which would have been free for i. Hence $P_4 \leq P_2$.

Therefore we can bound player i's contribution to edges below D' by P_1 (since u lies above v_k), and we can bound player i's contribution to A'_j by P_2. Taken together, we have that player i's new cost has not increased. Thus no player has increased their payments, and yet the total cost of the tree has decreased, which is a contradiction. □

We have shown that the price of stability in a single source game is 1. However, the algorithm for finding an optimal Nash equilibrium requires us to have a minimum cost Steiner tree on hand. Since this is often computationally infeasible, we present the following result.

Theorem 3.2.7 *Suppose we have a single source game and an α-approximate minimum cost Steiner tree T. Then for any $\varepsilon > 0$, there is a poly-time algorithm which returns a $(1+\varepsilon)$-approximate Nash equilibrium on a Steiner tree T', where $c(T') \leq c(T)$.*

Proof. To find a $(1+\varepsilon)$-approximate Nash equilibrium, we start by defining $\gamma = \frac{\varepsilon c(T)}{(1+\varepsilon)n\alpha}$. We now use Algorithm 3.2.3 to attempt to pay for all but γ of each edge in T. Since T is not optimal, it is possible that even with the γ reduction in price, there will be some edge e that the players are unwilling to pay for. If this happens, the proof of Theorem 3.2.2 indicates how we can rearrange T to decrease its cost. If we modify T in this manner, it is easy to show that we have decreased its cost by at least γ. At this point we simply start over with the new tree and attempt to pay for that.

Each call to Algorithm 3.2.3 can be made to run in polynomial time. Furthermore, since each call which fails to pay for the tree decreases the cost of the tree by γ, we can have at most $\frac{(1+\varepsilon)\alpha n}{\varepsilon}$ calls. Therefore in time polynomial in n, α, and ε^{-1}, we have formed a tree T' with $c(T') \leq c(T)$ such that the players are willing to buy T' if the edges in T' have their costs decreased by γ.

For all players and for each edge e in T', we now increase $p_i(e)$ in proportion to p_i so that e is fully paid for. Now T' is clearly paid for. To see that this is a $(1+\varepsilon)$-approximate Nash equilibrium, note that player i did not want to deviate before his payments were increased. If we let m' be the number of edges in T', then i's payments were increased by

$$\gamma \frac{p_i(T')}{c(T') - m'\gamma} m' = \frac{\varepsilon c(T) p_i(T') m'}{(1+\varepsilon)n\alpha(c(T') - m'\gamma)} \leq \frac{\varepsilon c(T) p_i(T')}{\alpha(1+\varepsilon)(1-\varepsilon)c(T')} \leq \varepsilon p_i(T').$$

Thus any deviation yields at most an ε factor improvement. □

Extensions

Both theorems 3.2.2 and 3.2.7 can be proven for the case where our graph G is directed, and players wish to purchase paths from t_i to s. Once we have shown that our theorems apply in the directed case, we can extend our model and give each player i a maximum cost $\max(i)$ beyond which he would rather pay nothing and not connect his terminals. It suffices to make a new terminal t_i' for each player i, with a directed edge of cost 0 to t_i and a directed edge of cost $\max(i)$ to s.

3.3 General Connection Games

In this section we deal with the general case of players that can have different numbers of terminals and do not necessarily share the same source terminal. As stated before, in this case the price of anarchy can be as large as N, the number of players. However, even the price of stability may be quite large in this general case.

Consider the graph illustrated in Figure 3.7, where each player i owns terminals s_i and t_i. The optimal centralized solution has cost $1+3\varepsilon$. If the path of length 1 were bought, each player $i > 2$ will not want to pay for any ε edges, and therefore the situation of players 1 and 2 reduces to the example in Section 3.1 of a game with no Nash equilibria. Therefore, any Nash equilibrium must involve the purchase of the path of length $N-2$. In fact, if each player $i > 2$ buys $\frac{1}{N-2}$ of this path, then we have a Nash equilibrium. Therefore, for any $N > 2$, there exists a game with the price of stability being nearly $N-2$.

Because the price of stability can be as large as $\Theta(N)$, and sometimes pure Nash equilibria may not exist at all, we cannot hope to be able to provide cheap Nash equilibria for the multi-source case. Therefore, we consider how cheap α-approximate Nash equilibria with small α can

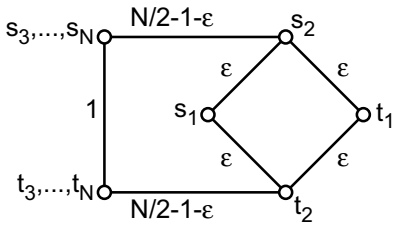

Figure 3.7: A game with high price of stability.

be, and obtain the following result, which tells us that there always exists a 3-approximate Nash equilibrium as cheap as the optimal centralized solution.

Theorem 3.3.1 *For any optimal centralized solution T^*, there exists a 3-approximate Nash equilibrium such that the purchased edges are exactly T^*.*

To prove this, we use Theorem 3.3.2, which provides sufficient conditions for an approximate Nash equilibrium. In Subsection 3.3.2 we complete the proof of Theorem 3.3.1. In Subsection 3.3.3 we address the very special case where the underlying graph is a path, and in Subsection 3.3.4 we give lower bounds and a polynomial time algorithm for finding an approximate Nash equilibrium.

3.3.1 Sufficient Conditions for Approximate Nash Equilibria

Given a set of bought edges T, denote by a *stable payment* p_i for player i a payment such that player i has no better deviation than p_i, assuming that the rest of T is bought by the other players. A Nash equilibrium must consist of stable payments for all players. However, what if in some solution, a player's payment p_i is not stable, but is a union of a small number of stable payments? This implies that each player's best deviation is not much less than its current payment. Specifically, we have the following general theorem.

Theorem 3.3.2 *Suppose we are given a payment scheme $p = (p_1, \ldots, p_N)$, with the set of bought edges T. If for all i, the payment p_i is a union of at most α stable payments (with respect to T), then p is an α-approximate Nash equilibrium.*

Proof. Let p_i' be the best deviation of player i given p, and let $p_i^1, p_i^2, \ldots, p_i^\alpha$ be the stable payments which together form p_i. The fact that p_i' is a valid deviation for i means that the set of bought edges T with p_i taken out and p_i' added still connects the terminals of i. p_i^j being a stable payment means that if i only pays for p_i^j and the rest of T is bought by other players, then the best deviation of i is al least as expensive than p_i^j. In this case, p_i' is still a possible deviation, since if taking out p_i and adding p_i' connects the terminals of i, then so does taking out p_i^j and adding p_i'. Therefore, we know that the cost of p_i' is no smaller than the cost of any p_i^j, and $\alpha \cdot cost(p_i') \geq cost(p_i)$, as desired. □

Notice that the converse of this theorem is not true. Consider an example where player i is contributing to an edge which it does not use to connect its terminals. If this edge is cheap, this would still form an approximate Nash equilibrium. However, this edge would not be contained in any stable payment of player i, so p_i would not be a union of stable payments.

To prove Theorem 3.3.1, we will construct a payment scheme on the optimal centralized solution such that each player's payment is a union of 3 stable payments. The stable payments we use for this purpose involve each edge being paid for by a single player, and have special structure. We call these payments *connection sets*. Since there is no sharing of edge costs by multiple players in connection sets, we will often use sets of edges and sets of payments interchangeably. T^* below denotes an optimal centralized solution, which we know is a forest.

Definition 3.3.3 *A connection set S of player i is a subset of edges of T^* such that for each connected component C of the graph $T^* \setminus S$, we have that either*
(1) any player that has terminals in C has all of its terminals in C, or
(2) player i has a terminal in C.

Intuitively, a connection set S is a set such that if we removed it from T^* and then somehow connected all the terminals of i, then all the terminals of all players are still connected in the resulting graph. We now have the following lemma, the proof of which follows directly from the definition of a connection set.

Lemma 3.3.4 *A connection set S of player i is a stable payment of i with respect to T^*.*

Proof. Suppose that player i only pays exactly for the edges of S, and the other players buy the edges in $T^* \setminus S$. Let Q be a best deviation of i in this case. In other words, let Q be a cheapest set of edges such that the set $(T^* \setminus S) \cup Q$ connects all the terminals of i. To prove that S is a stable payment for i, we need to show that $cost(S) \leq cost(Q)$.

Consider two arbitrary terminals of any player. If these terminals are in different components of $T^* \setminus S$, then by definition of connection set, each of these components must have a terminal of i. Therefore, all terminals of all players are connected in $(T^* \setminus S) \cup Q$, since $(T^* \setminus S) \cup Q$ connects all terminals of i. Since T^* is optimal, we know that $cost(T^*) \leq cost((T^* \setminus S) \cup Q)$. Since $S \subseteq T^*$ and Q is disjoint from $T^* \setminus S$, then $cost((T^* \setminus S) \cup Q) = cost(T^*) - cost(S) + cost(Q)$, and so $cost(S) \leq cost(Q)$. □

More generally, if we assume player i is paying exactly for a set $S \subseteq T^*$, consider the components of $T^* \setminus S$ which do not obey Property 1 of Definition 3.3.3. We will call such components *uncoupled*, since there are terminals in them which are not connected. If S is a connection set, each such component contains a terminal of i. Call these terminals the *certificate terminals* of S. If S is not a connection set, there may be components without certificate terminals in them.

3.3.2 Proof of Theorem 3.3.1 (Existence of 3-Approx Nash Equilibrium)

In this subsection, we prove that for any optimal centralized solution T^*, there exists a 3-approximate Nash equilibrium such that the purchased edges are exactly T^*.

Let T^i be the unique smallest subtree of T^* which connects all terminals of player i. For simplicity of the proof, we assume that T^* is a tree, since otherwise we can apply this proof to each component of T^*.

By Theorem 3.3.2 and Lemma 3.3.4, to prove Theorem 3.3.1 it is enough to find a payment scheme for T^* so that no player pays for more than 3 connection sets. We now exhibit such a scheme. First, each player i pays for the edges belonging only to T^i and no other tree T^j. This is clearly a connection set, so we want each player to pay for at most 2 more. We can contract the edges now paid for, forming a new tree T^* which the players must pay for, and on which each edge belongs to at least two different T^i's.

Intuition and Proof Outline By Lemma 3.3.5, we know how to make each player pay for at most two connection sets if T^* is just a path. The idea of the proof of Theorem 3.3.1 is to partition the tree T^* into paths, use Lemma 3.3.5 to form payments on these paths, and then "hook up" all these payments together so that in total, each player is still paying for 2 connection sets.

The partitioning process into paths works as follows. At first we take an arbitrary demand path R (i.e. take 2 terminals belonging to the same player, and take the path in T^* between them). Now we use Lemma 3.3.5 on this path. We get payments for this path, and a set A of terminals. We add the paths starting at terminals of A and ending at R to our partition, and continue like this recursively. (This is the purpose of the set A in Lemma 3.3.5).

Now we must prove that these payments which were done on a "path-by-path" basis form a "2-connection-set" payment on the entire tree T^*. This is shown in Lemma 3.3.6.

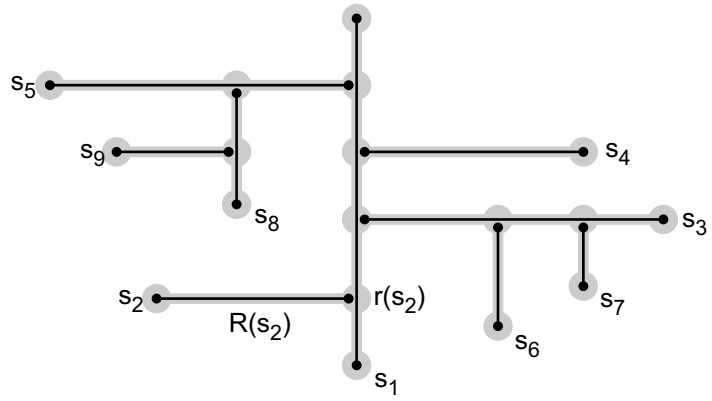

Figure 3.8: A decomposition of T^* into paths $R(t)$

The Payment Scheme and Path Partitioning Now we recursively assign players to the edges of T^*. Each edge will be assigned a player, which will pay for it. At every point in this process, we have a set of directed paths R to be paid for. Each of these paths is labeled with some terminal s at which it begins, and each of these paths ends at a node of a path paid for at a previous time. We will call such a path $R(s)$, since for each terminal there will be at most 1 path starting at that terminal, and we will call the last node of this path $r(s)$. Figure 3.8 shows a decomposition of T^* into these paths after this recursion is done. To explain how this process of path decomposition proceeds, we first need the following key lemma about connection sets in paths.

Suppose our entire graph is a path P, with nodes v_k, $k = 1 \dots n$. Denote the set of all terminals located at v_k by U_k. Select an arbitrary terminal $s \in U_1$ such that the player owning s also owns a terminal in U_n (if none such exists, things only become easier). The following technical result holds in this scenario.

Lemma 3.3.5 *If $U_1 \neq \{s\}$, then there exists a set of terminals $A = \{s\} \cup \{s_1, \dots, s_{n-1}\}$, $s_k \in U_k$, and a payment scheme on path P with each player i paying exactly for the set of edges S_i such that:*

1. *Any player i owning a terminal in U_n pays for at most 1 connection set, with all certificate terminals from A or U_n.*
2. *If player i owns s, it pays for at most 1 connection set, with all certificate terminals from A (not U_n).*
3. *If players i does not own a terminal in U_n, there is at most one uncoupled component of $P \backslash S_i$ without a certificate terminal of i from A.*

The proof of this lemma can be found in the next subsection. We can now explain the recursive process of path decomposition. Initially, select $R(s)$ to be a path from an arbitrary terminal s to another terminal of the same player in T^i, direct this path away from s, and set $R = \{R(s)\}$. At each time step t in the recursion, we have a subset T of T^* which has already been assigned payments, a set of paths R that needs to be paid for, and the rest of T^*, which has not been partitioned into paths yet. The recursion proceeds until all of T^* is paid for. For each $R(s) \in R$, we do the following:

1. Invoke Lemma 3.3.5, with $P = R(s)$ and U_k being all the terminals in the subtree of T^* containing v_k that we obtain by cutting the edges in $R(s)$ incident to v_k. $U_1 \neq \{s\}$ because we have contracted all edges belonging to a single T^i. From Lemma 3.3.5, we obtain a payment assignment for $R(s)$, and a set of terminals $A = \{s_1, \ldots\}$ such that $s_k \in U_k$.

2. For each k, let $R(s_k)$ be the path from $s_k \in A$ to v_k in T^*, and add this path to the set of paths R that needs to be paid for.

3. For every edge e incident to a node $v_k \in R(s)$ which is not already paid for, and which is not covered by R, let s' be an arbitrary terminal in the subtree not containing v_k obtained by cutting e, with the condition that if player i owns s', then $e \in T^i$. Let $R(s')$ be the path from s' to v_k, and add it to R.

In this recursive manner, we decompose all of T^* into paths, and assign which player pays for which edges. Suppose this process runs for τ time steps (so T^* is decomposed into τ paths). Let S_i be the edges paid for by player i after this process, and let $S_i(t, t')$ be the edges assigned to player i in the time steps t and t', inclusive. In other words, $S_i = S_i(1, \tau)$.

To prove Theorem 3.3.1, we now need to prove that in this payment scheme, each player pays for at most 2 connection sets. In fact, we now prove the following lemma, which is a stronger statement.

Lemma 3.3.6 *Suppose the recursive process assigns edges S_i to be paid for by player i. Then there is at most 1 connected component of $T^* \backslash S_i$ which is uncoupled and does not contain a terminal of i. Furthermore, this component must intersect the path $R(s)$ which is the first path paid for sharing edges with T^i.*

Proof. For each path $R(s)$ in the above recursive process, let $T(s)$ be the tree containing $R(s)$ obtained by removing the last node $r(s)$ of $R(s)$ from T^*. Along with this lemma, we will also prove the following invariant: *If i owns s, then the component of $T^* \backslash S_i(t, \tau)$ containing $r(s)$ always contains a terminal of i in $T(s)$.*

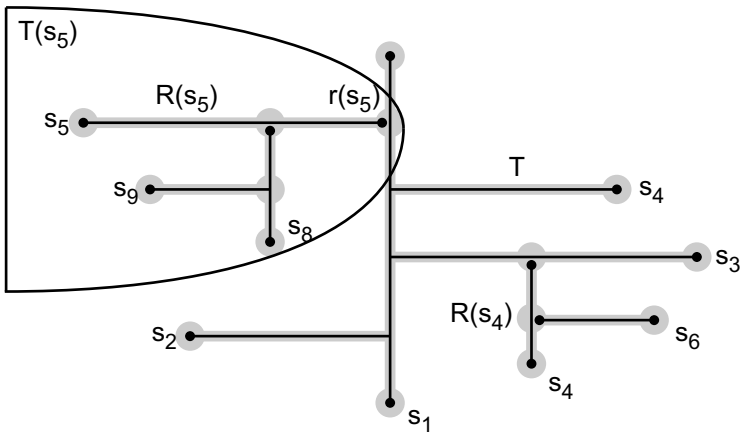

Figure 3.9: A snapshot of the recursive process at the point where $R(s_5)$ is being paid for.

We prove this inductively in reverse, i.e. at each step of the induction we show that the lemma holds for $S_i(t, \tau)$ instead of S_i, decreasing t at every step. In the base case of $S_i(\tau + 1, \tau) = \emptyset$, the players have not been assigned any payments, so the lemma and the invariant are trivially true. Now assume that the lemma and the invariant hold for $S_i(t + 1, \tau)$. We want to show that

they also hold for $S_i(t,\tau)$. At step t in the above recursive process, there is some subtree T of T^* which has already been assigned payments, there is a set of paths R we are currently trying to pay for, and there is the rest of T^*, which has not been partitioned into paths yet. Figure 3.9 shows the tree T^* at the time step when $R(s_5)$ is being paid for. In the figure, R consists of $R(s_5)$ and $R(s_4)$, with the parts of T^* involving s_6, s_8, and s_9 not having been paid for or put into R.

At step t, we invoke Lemma 3.3.5 to assign player payments for a path $R(s) \in R$. Suppose $R(s)$ is of length n, and let P and U_k be the path and sets which we give to Lemma 3.3.5 as inputs. We now have 3 cases to consider.

Case 1: Player i has no terminals outside $T(s)$. If all of i's terminals are in the same set U_k, then the Lemma still holds by the inductive hypothesis. Otherwise, every component of $T^*\backslash S_i(t,\tau)$ that does not intersect $R(s)$ is either coupled or has a terminal of i by the inductive hypothesis. By Lemma 3.3.5, we know that at time step t, all except 1 connected component of $T^*\backslash S_i(1,t)$ intersecting $R(s)$ is coupled or contains a certificate terminal of i. The recursive process at step t adds paths starting at these certificate terminals to R, and because of the invariant being true after step t, all the corresponding components of $T^*\backslash S_i$ still contain a terminal of i. [Maybe put in more explanation about what "corresponding" means, and that the coupled components of $T^*\backslash S_i(1,t)$ are still coupled in $T^*\backslash S_i(t,\tau)$] This proves the Lemma for this case. The fact that i has no terminals outside $T(s)$ implies that $R(s)$ is the first path being paid for that intersects T^i, as desired.

If i owns s, we must also show that the invariant still holds. In the recursive process, s can only be chosen as the terminal to head $R(s)$ if all of $R(s)$ is in T^i. Since i has no terminals outside $T(s)$, this implies that $r(s)$ is a terminal of i. Therefore, the invariant is satisfied.

Case 2: Player i has terminals outside $T(s)$ and owns s. As before, the components of $T^*\backslash S_i(t,\tau)$ not intersecting $R(s)$ must either be coupled or contain a terminal of i by the inductive hypothesis. When we invoke Lemma 3.3.5 at step t, U_n contains a terminal of i, so Lemma 3.3.5 tells us that i pays for at most 1 connection set $S_i(t,t)$ in $R(s)$, and all of the certificate terminals of $S_i(t,t)$ are from a set $A = \{s\} \cup \{s_1, \ldots, s_{n-1}\}$. This implies that every component of $T^*\backslash S_i(1,t)$ intersecting $R(s)$ is either coupled or contains a certificate terminal of i from A. By the same argument as in Step 1, the same must be true for all components of $T^*\backslash S_i(t,\tau)$.

We must also show that the invariant still holds. Consider the component C of $T^*\backslash S_i(t,\tau)$ containing $r(s)$. If it is coupled, then it must contain all terminals of i, since i has terminals outside $T(s)$. Therefore, it must contain s, since it is a terminal of i, and so the invariant is satisfied. If C is uncoupled, it must have contained a certificate terminal of i from A during the invocation of Lemma 3.3.5. That terminal s' is contained in $T(s)$, and a path $R(s')$ must have been added to R at step t. By the invariant, there must be a terminal of i in the component of $T(s')\backslash S_i(t+1,\tau)$ intersecting $R(s)$. Since $S_i(t,t)$ does not involve any payments to $T(s')$, this means the invariant still holds.

Case 3: Player i has terminals outside $T(s)$, but does not own s. This means when we invoke Lemma 3.3.5, U_n contains a terminal of i, so Lemma 3.3.5 tells us that i pays for at most 1 connection set $S_i(t,t)$ in $R(s)$, and all of the certificate terminals of $S_i(t,t)$ are from A or U_n. By an argument similar to Case 2, both the Lemma and the invariant still hold. □

3.3.3 Proof of Lemma 3.3.5 (Connection Sets in Paths)

In this section, we prove Lemma 3.3.5 about connection sets in paths. This lemma is a basic building block for our proof of Theorem 3.3.1, and is a rather technical result.

Proof. Throughout this proof, we order the nodes of P from v_1 to v_n. For example, the "first" terminal of i will mean the one closest to v_1.

For every terminal $u \in U_k$, with $k \neq n$, define a subpath $Q(u)$ as follows. If there exists a node v_ℓ with $\ell > k$ and the player who owns u also owns a terminal in U_ℓ, then set $Q(u)$ to be the subpath of P from v_k to v_ℓ. If there is no such node, set $Q(u)$ to be the subpath of P starting at the first terminal of the player who owns u, and ending at v_k.

In the special case where the same player owns both u and s, if there exists a node v_ℓ with $\ell < k$ and the player who owns u also owns a terminal in U_ℓ, then set $Q(u)$ to be the subpath from v_ℓ to v_k. If there is no such node v_ℓ, set $Q(u)$ to be empty.

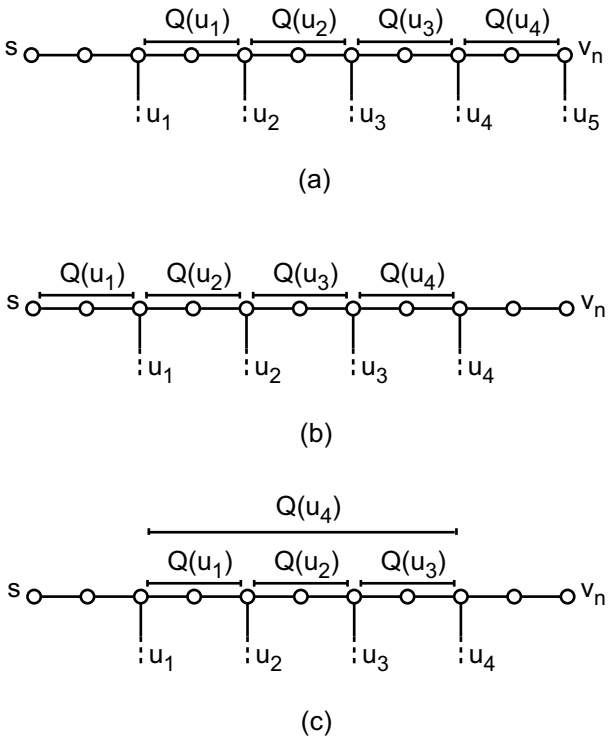

Figure 3.10: The paths $Q(u)$ for a single player i. (a) i does not own s, but has a terminal in U_n (b) i owns s (c) i does not own s or a terminal in U_n

Figure 3.10(a) illustrates what the paths $Q(u)$ for terminals u of i look like if i does not own s and owns a terminal in U_n. In this figure, the terminals u_1 through u_5 are all the terminals of i in any of the sets U_k. Figure 3.10(b) shows the same thing in the case that i owns s. Figure 3.10(c) shows this for the case that i owns neither s nor a terminal of U_n.

Suppose we were able to form payments S_i such that for each terminal u of i with $u \notin U_n$ we can include one edge of $Q(u)$ into S_i, but nothing else. Then the conditions in Lemma 3.3.5 (not involving A) would be satisfied. The proof of this is as follows.

Case 1: i does not own s, and has a terminal in U_n. Then every component of $P \backslash S_i$ contains a terminal of i, since there is a terminal of i between every $Q(u)$ for u belonging to i, as well as before the first such $Q(u)$, and after the last one (since the last such $Q(u)$ ends at v_n, and i has a terminal in U_n).

Case 2: i owns s. Similarly to Case 1, every component of $P \backslash S_i$ contains a terminal of i, since there is a terminal of i between and after every $Q(u)$ for u belonging to i by definition of $Q(u)$. There is a terminal of i before the first such $Q(u)$ because s belongs to i.

Case 3: i does not own s, and has no terminals in U_n. Let u' be the last terminal of i. For the edges of S_i associated with $Q(u)$, $u \neq u'$, the argument is the same as the previous cases. We now have an additional edge in $Q(u')$ which is in S_i. This edge can form at most 1 component of $P \backslash S_i$ which does not contain a terminal of i, as desired.

We have now shown that if we associate each edge of S_i with a path $Q(u)$ for terminals u owned by i so that there is at most 1 edge associated with each $Q(u)$, then the conditions of Lemma 3.3.5 are satisfied. Now suppose we form payments S_i such that for each terminal u of i with $u \notin U_n$ we include several edges of $Q(u)$ into S_i. Suppose these edges are $e_1, e_2, \ldots e_\ell \in Q(u)$

with e_1 being closest to s. As long as the components of $P\backslash S_i$ between e_j and e_{j+1} are coupled, then by the above argument the conditions of Lemma 3.3.5 still hold. Our goal now is to associate with each terminal u of i, $u \notin U_n$, a set of edges $e_1, e_2, \ldots e_\ell \in Q(u)$ so that each component of $P\backslash S_i$ between e_j and e_{j+1} is coupled.

Consider a pair of edges e and e' such that they are contained in exactly the same paths $Q(u)$. Let C be the component of $P\backslash\{e, e'\}$ between e and e'. Every path $Q(u)$ that starts in C must also end in C. This means that if a player i has a terminal $u' \in C$, then all of i's terminals must be in C, since by construction of the paths $Q(u)$, for every terminal of i there is a path $Q(u)$ which starts at that terminal and ends at another terminal of i. C is coupled, therefore. This implies that to prove this lemma, it is enough for us to associate with each terminal $u \notin U_n$ a set of edges $e_1, e_2, \ldots e_\ell \in Q(u)$ so that these edges lie in exactly the same paths $Q(u)$, and so that all edges are covered by this assignment.

Definition 3.3.7 *A Q-set L is a maximal set of edges of P such that for every edge $e \in L$, the set of paths $Q(u)$ that contain e is exactly the same, for $u \in \cup U_k$.*

We desire to form a matching between Q-sets and terminals $u \notin U_n$, covering the Q-sets, so that if L is matched to u, then $L \subseteq Q(u)$. Then if we set L to be paid for by the player that owns u, the conditions in Lemma 3.3.5 are satisfied.

We also would like to construct a set A as specified in the statement of Lemma 3.3.5. Notice that in the above case argument about components of $P\backslash S_i$, the certificate terminals if only some paths $Q(u)$ have edges in S_i are as follows. In Case 1, they are the terminals u such that S_i has an edge in $Q(u)$, together with a terminal in U_n. In Case 2, they are the similar terminals together with s. In Case 3, they are the similar terminals together (possibly) with the last terminal of i. Suppose that in the above matching, only a single terminal from each U_k is matched to a Q-set, and set A to be s together with the matched terminals. By this case analysis, for each player i, all the certificate terminals are contained in A or U_n, and for the player that owns s, all the certificate terminals are contained in A (not U_n), as desired in Lemma 3.3.5. The only unclear case if for a player i in Case 3. If the last terminal u' of i is matched to a Q-set and is therefore in A, then we are done. Otherwise, it must be that i has no components of $P\backslash S_i$ which are uncoupled and have no terminals of i, since there is only one Q-set paid for by i contained in each $Q(u)$ for u a terminal of i. Therefore, the right-most component of $P\backslash S_i$ is the only one that may be uncoupled and not contain a terminal of i from A, as desired.

We have now shown that to prove the lemma, it is enough to form a matching between Q-sets and terminals $u \notin U_n$, covering the Q-sets, so that if L is matched to u, then $L \subseteq Q(u)$, and so that only one terminal for each U_k is matched to a Q-set. We do this by constructing the following bipartite graph (Y, Z).

Let Y have a node for each Q-set in P, and let Z be the nodes of P. Form an edge between a node $v_k \in Z$ and node $L \in Y$ if there exists some terminal $u \in U_k$ such that $L \subseteq Q(u)$. For $X \subseteq Y$, define $\partial(X)$ to be the set of nodes in Z which X has edges to. According to Hall's Matching Theorem, there exists a matching in this bipartite graph with all nodes of Y incident to an edge of the matching if for each set $X \subseteq Y$, $|\partial(X)| \geq |X|$. Arrange the edges of the Q-sets of X in the order they appear in P. We want to show that between every Q-set of X, there appears a node belonging to $\partial(X)$.

Consider some edge e of X that is not the first one in P. Suppose this edge belongs to Q-set L, and the previous edge e' in X to some Q-set L'. Since these are different Q-sets, there must be some path $Q(u)$, with an endpoint at node v_j between e' and e, and $u \in U_k$ and $v_k \in \partial(X)$ (since $Q(u)$ contains either e or e'). If u belongs to the same player as s, let $Q(u)$ be the left-most such path that contains e'. This implies that $v_j = v_k$, so $v_j \in \partial(X)$. If u belongs to a player i not owning s, let $Q(u)$ be the right-most such path that contains e, if it exists. Once again, this implies that $v_j = v_k$. If such a path does not exist, this implies that the right-most terminal u' of i is between e and e', and the left-most is not. By the definition of $Q(u')$, it must be that $e' \in Q(u')$, so we have a node of $\partial(X)$ between e' and e.

Now let L be the first Q-set of X that appears in P. By our assumption, $U_1 \neq \{s\}$. Therefore, there must be some $Q(u)$ containing L such that u belongs to a different player than s. By construction of $Q(u)$, $u \in U_k$ for some v_k that comes before L. This means that there is a node of $\partial(X)$ before L.

Therefore, $|X| \leq |\partial(X)|$, and so we can assign a terminal $s_k \in U_k$ to each node v_k, $k = 1 \ldots n-1$, such that these terminals are matched to all the Q-sets of P, with the Q-set matched to s_k contained in $Q(s_k)$. By the above argument, this proves the lemma. □

3.3.4 Extensions and Lower Bounds

We have now shown that in any game, we can find a 3-approximate Nash purchasing the optimal network. We proved this by constructing a payment scheme so that each player pays for at most 3 connection sets. This is in fact a tight bound. In the example shown in Figure 3.11, there must be players that pay for at least 3 connection sets. There are N players, with only two terminals (s_i and t_i) for each player i. Each player must pay for edges not used by anyone else, which is a single connection set. There are $2N-3$ other edges, and if a player i pays for any 2 of them, they are 2 separate connection sets, since the component between these 2 edges would be uncoupled and would not contain any terminals of i. Therefore, there must be at least one player that is paying for 3 connection sets.

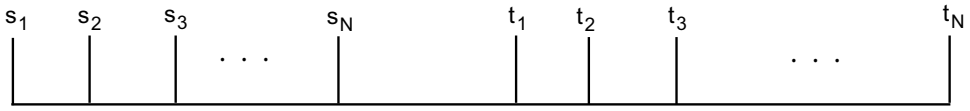

Figure 3.11: A graph where players must pay for at least 3 connection sets.

This example does not address the question of whether we can lower the approximation factor of our Nash equilibrium to something other than 3 by using a method other than connection sets. As a lower bound, in Section 3.4 we give a simple sequence of games such that in the limit, any Nash equilibrium purchasing the optimal network must be at least $(\frac{3}{2})$-approximate.

Polynomial-time algorithm Since the proof of Theorem 3.3.1 is constructive, it actually contains a polynomial-time algorithm for generating a 3-approximate Nash equilibrium on T^*. We can use the ideas from Theorem 3.2.7 to create an algorithm which, given an α-approximate Steiner forest T, finds a $(3+\varepsilon)$-approximate Nash equilibrium which pays for a Steiner forest T' with $c(T') \leq c(T)$, as follows. However, this algorithm requires a polynomial-time optimal Steiner tree finder as a subroutine. We can forego this requirement at the expense of a higher approximation factor.

We start by defining $\gamma = \frac{\varepsilon c(T)}{(1+\varepsilon)n\alpha}$, for ε small enough so that γ is smaller than the minimum edge cost. The algorithm of Theorem 3.3.1 generates at most 3 connection sets for each player i, even if the forest of bought edges is not optimal. We use this algorithm to pay for all but γ of each edge in T. We can check if each connection set is actually cheaper than the cheapest deviation of player i, which is found by the cheapest Steiner tree algorithm. If it is not, we can replace this connection set with the cheapest deviation tree and run this algorithm over again. The fact that we are replacing a connection set means that all the terminals are still connected in the new tree. If we modify T in this manner, it is easy to see that we have decreased its cost by at least γ.

We can now use the arguments from Theorem 3.2.7 to prove that this algorithm produces a $(3+\varepsilon)$-approximate Nash equilibrium, and runs in time polynomial in n, α, and ε^{-1}. It requires a poly-time Steiner tree subroutine, however. If each player only has two terminals, finding the cheapest Steiner tree is the same as finding the cheapest path, so this is possible, and we can indeed find a cheap $(3+\varepsilon)$-approximate Nash equilibrium in polynomial time.

For the case where players may have more than two terminals, we can easily modify the above algorithm to use polynomial time approximations for the optimal Steiner tree, at the expense of a higher approximation factor. If we use a 2-approximate Steiner forest T, and an optimal Steiner tree 1.55-approximation algorithm from [70] as our subroutine, then the above algorithm actually gives a $(4.65 + \varepsilon)$-approximate Nash equilibrium on T' with $c(T') \leq 2 \cdot OPT$, in time polynomial in n and ε^{-1}.

3.4 Lower bounds and NP-Hardness

Lower bounds for approximate Nash on the optimal network

Claim 3.4.1 *For any $\epsilon > 0$, there is a game such that any equilibrium which purchases the optimal network is at least a $(\frac{3}{2} - \epsilon)$-approximate Nash equilibrium.*

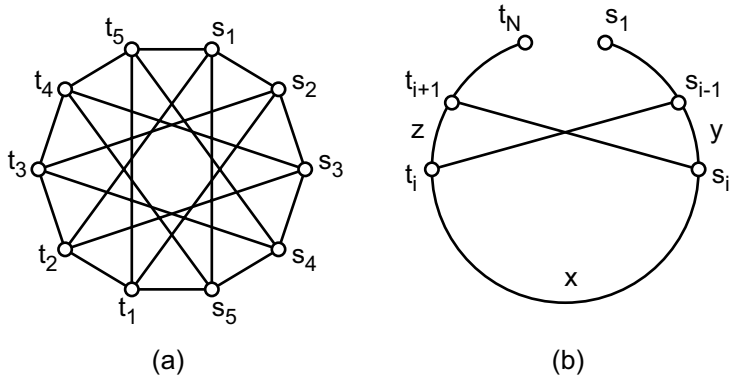

Figure 3.12: A game with best Nash equilibrium on OPT tending to at least a $\frac{3}{2}$-approximation.

Proof. Construct the graph H_N on $2N$ vertices as follows. Begin with a cycle on $2N$ vertices, and number the vertices 1 through $2N$ in a clockwise fashion. For vertex i, add an edge to vertices $i + N - 1 \mod (2N)$ and $i + N + 1 \mod (2N)$. Let all edges have cost 1. Finally, we will add N players with 2 terminals, s_i and t_i, for each player i. At node j, add the label s_j if $j \leq N$ and t_{j-N} otherwise. Figure 3.12(a) shows such a game with $N = 5$.

Consider the optimal network T^* consisting of all edges in the outer cycle except (s_1, t_N). We would like to show that any Nash which purchases this solution must be at least $(\frac{6N-21}{4N-11})$-approximate. This clearly would prove our claim.

First we show that players 1 and N are not willing to contribute too much to any solution that is better than $(\frac{3}{2})$-approximate. Suppose we have such a solution. Define x to be player 1's contribution to his connecting path in T^*, and define y to be his contribution to the remainder of T^*. Thus player 1 has a total payment of $x + y$. Player 1 can deviate to only pay for x. Furthermore, player 1 could deviate to purchase only y and the edge (s_1, t_N). If we have a solution that is at most $(\frac{3}{2})$-approximate, then we have that $\frac{x}{x+y} \geq \frac{2}{3}$ and similarly $\frac{y+1}{x+y} \geq \frac{2}{3}$. Taken together this implies that $\frac{1}{x+y} \geq \frac{1}{3}$, or $x + y \leq 3$. A symmetric argument shows that player N is also unwilling to contribute more than 3.

Thus we have that the remaining $N - 2$ players must together contribute at least $2N - 7$. Therefore there must be some player other than 1 or N who contributes $\frac{2N-7}{N-2}$. Suppose player i is such a player. Let x be the amount that player i contributes to his connecting path in T^*. Let y be his contribution to (s_{i-1}, s_i) and let z be his contribution to (t_i, t_{i+1}). See Figure 3.12(b).

Now consider three possible deviations available to player i. He could choose to contribute only x. He could contribute y and purchase edge (s_{i-1}, t_i) for an additional cost of 1. Or he could

contribute z and purchase edge (s_i, t_{i+1}), also for an additional cost of 1. We will only consider these possible deviations, although of course there are others. Note that if i was contributing to any other portion of T^*, then we could remove those contributions and increase x, y, and z, thereby strictly decreasing i's incentive to deviate. Thus we can safely assume that these are i's only payments, and hence

$$x + y + z \geq \frac{2N - 7}{N - 2}.$$

Since i is currently paying at least $x + y + z$, we know that his incentive to deviate is at least $\max(\frac{x+y+z}{x}, \frac{x+y+z}{y+1}, \frac{x+y+z}{z+1})$. This function is minimized when $x = y + 1 = z + 1$. Solving for x we find that

$$x \geq \frac{4N - 11}{3N - 6}.$$

Thus player i's incentive to deviate is at least

$$\frac{x + y + z}{x} \geq \frac{3x - 2}{x} = 3 - \frac{2}{x} \geq 3 - 2\frac{3N - 6}{4N - 11} = \frac{6N - 21}{4N - 11}.$$

Therefore as N grows, this lower bound on player i's incentive to deviate tends towards $\frac{3}{2}$. Note that in this proof, we only considered one optimal network, namely T^*. If we modify G by increasing the costs of all edges not in T^* by some small $\varepsilon > 0$, then T^* is the only optimal network. Repeating the above analysis under these new costs still yields a lower bound of $\frac{3}{2}$ for the best approximate Nash on T^* in the limit as N grows and ε tends to 0. □

NP Completeness

In this section, we present a brief proof that determining the existence of Nash equilibria in a given graph is NP-complete if the number of players is $O(n)$ (where n is the number of nodes in the graph). We present a reduction from 3-SAT to show that the problem is NP-hard. The graph constructed will have unit cost edges.

Consider an arbitrary instance of 3-SAT with clauses C_j and variables x_i. We will have a player for each variable x_i, and two players for each clause C_j. For each variable x_i construct the gadget shown in Figure 3.13a. The source and sink of the player x_i are the vertices s_i and t_i respectively. When player x_i buys the left path or right path, this corresponds to x_i being set to be true or false, respectively. For clarity, we will refer to this player as being the i^{th} variable player.

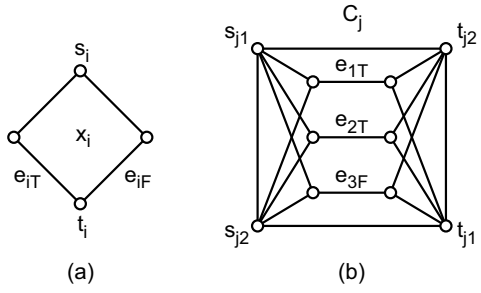

Figure 3.13: Gadgets for the NP-completeness reduction.

Next, we construct a gadget for each clause C_j. The construction is best explained through an example clause $C_j = (x_1 \vee x_2 \vee \overline{x}_3)$ whose gadget is given in Figure 3.13b. The two players for C_j have their source sink pairs as (s_{j1}, t_{j1}) and (s_{j2}, t_{j2}) respectively. We will call both players on this gadget clause players. The final graph is constructed by gluing these gadgets together at the appropriate labeled edges. Specifically, the edges in clause gadget C_j labeled e_{1T}, e_{2T}, and e_{3F} are the same edges that appear in the corresponding variable gadgets. In other words,

among all clauses and variable gadgets, there is only one edge labeled e_{iT} and only one labeled e_{iF}, and all the interior nodes in the gadget for each clause C_j are nodes in variable gadgets.

Suppose that there is a satisfying assignment A in our 3-SAT instance. Consider the strategy in which variable player i fully buys the left path if x_i is true in A and fully buys the right path otherwise. Since this is a satisfying assignment, by our construction each clause gadget has at least one interior edge fully paid for by a variable player. For each clause C_j, let e be one such edge, and let both players on this gadget buy the unique path of length 3 that connects their terminals which uses edge e. It is easy to see that the clause players are satisfied as the cost of this path to each clause player is 2, the minimum that he has to pay on any path from source to sink under the current payment scheme. The cheapest deviation for each variable player also costs 2, and therefore they do not have any incentive to move either. Thus, this forms a Nash equilibrium.

Suppose now that there is a Nash equilibrium. We will argue that this Nash equilibrium has to have a specific set of edges paid for. First, note that the contribution of each player is not more than 2, as the length of the shortest path is exactly 2.

Now suppose some perimeter edge of clause C_j is being bought. We know from the example in Figure 3.1 that perimeter edges cannot be bought by the clause players in C_j alone, for that would not constitute a Nash strategy. Therefore there must be some other player, variable or clause, contributing to the perimeter edge of C_j. Also, since this is a Nash strategy, any perimeter edge on which there is a positive contribution by any player must be fully bought. And once any perimeter edge of C_j has a positive contribution from a non-C_j player the payments of both the clause players of C_j will be strictly less than 2 in a Nash strategy.

Suppose one of the clauses, C_j, has some perimeter edge bought. Since at Nash equilibrium, the set of edges bought must form a Steiner forest, we look at the component of the Steiner forest that has the the clause C_j. We will show that the number of edges in this component is more than twice the number of players involved. Then, there must be a player who is paying more than 2, and hence this cannot be a Nash equilibrium.

Suppose there are x clause players and y variable players in the component of the forest containing C_j. We know from the example in Figure 3.1 that $x + y > 2$. Then, the total number of nodes in the Nash component containing C_j is $2x + 3y$ as we have to count the two source-sink nodes for each clause player and the three nodes on the path of each variable player. Since this is a connected tree, the total number of edges in this component is $2x + 3y - 1$. The average payment per player is then given by $\frac{2x+3y-1}{x+y} = 2 + \frac{y-1}{x+y}$. Now if $y > 1$, then the average payment per player is more than 2. Thus there must be some player who is paying more than 2, which is infeasible in a Nash. If $y = 1$, then the average payment per player is exactly 2. But again, since we know that the clause players of C_j pay strictly less than 2 each, there must be some player who pays strictly more than 2, which is again impossible. Lastly, we cannot have $y = 0$ as then whenever a clause player participates in paying for another clause, he must use a node in the path of a variable player, and thereby include this variable player in the component of C_j.

This implies that variable players only select paths within their gadget. Furthermore, it implies that variable players must pay fully for their entire path. Suppose i is a variable player who has selected the left (true) path, but has not paid fully for the second edge in that path. The remainder of this cost must be paid for by some clause player or players. But for such a clause player to use this edge, he must also buy two other edges, which are not used by any other player. Hence such a clause player must pay strictly more than 2. But there is always a path he could use to connect of cost exactly 2, so this can not happen in a Nash equilibrium. Thus we have established that variable players pay fully for their own paths.

Now consider any clause gadget. Since we have a Nash equilibrium, we know that only internal edges are used. But since each clause player can connect his terminals using perimeter edges for a cost of exactly 2, one of the interior variable edges must be bought by a variable player in each clause gadget. If we consider a truth assignment A in which x_i is true if and only if player i selects the left (true) path, then this obviously satisfies our 3-SAT instance, as every clause has at least one variable forcing it to evaluate to true.

Therefore, this game has a Nash equilibrium if and only if the corresponding formula is satisfiable, and since this problem is clearly in NP, determining whether a Nash equilibrium exists is NP-Complete.

Chapter 4
The Connection Game with Fair Sharing

In this chapter, we consider a version of the Connection Game where every player using an edge e must pay a fair share of its cost. Such cost-sharing mechanisms can be viewed as an underlying protocol that determines how much a network serving several participants will cost to each of them. Our general intent is to determine how this basic cost-sharing mechanism serves to influence the strategic behavior of the users, and what effect this has on the structure and overall cost of the network one obtains. Many of the results in this chapter initially appeared in [6].

To understand how much the fair sharing rule changes the Connection Game, see Table 4.1, which lists some results that hold in the Connection Game with Arbitrary Cost Sharing and considers what happens to them once we introduce the Shapley-value cost sharing rule into the game. We see a lot of improvements. Nash equilibria (NE) always exist, for example, and the P.o.S. (Price of Stability) in general is at most $O(\log N)$. There are also some nice results that no longer hold, however, such as our main price of stability result for single source games, and the fact that the centralized optimum (OPT) can be thought of as a β-approximate Nash equilibrium, with small β.

Arbitrary Sharing	Fair Sharing
NE may not exist	NE always exists
P.o.S. $O(N)$	P.o.S. $O(\log N)$
P.o.S. is 1 for Single Source	P.o.S is $\Omega(\log N)$ for Single Source
OPT is an approx NE	OPT may be far from a NE
NE are trees	NE can have cycles

Table 4.1: Comparison of some results for different versions of the Connection Game

4.1 The Model with Fair Sharing

In this section we consider the *Fair Connection Game* for N players as defined in the Introduction. Let a directed graph $G = (V, E)$ be given, with each edge having a nonnegative cost c_e. Each player i has a set of terminal nodes that it wants to connect. A strategy of a player is a set of edges $S_i \subset E$ such that S_i connects all of i's terminal nodes. We assume that we use the Shapley value to share the cost of the edges, i.e. all players using an edge split up the cost of the edge equally. Given a vector of players' strategies $S = (S_1, \ldots, S_N)$, let x_e be the number of players whose strategy contains edge e. Then the cost to player i is $C_i(S) = \sum_{e \in S_i}(c_e/x_e)$, and the goal of each player is to connect its terminals with minimum total cost.

Notice that the player strategies here are different from the strategies in the Connection Game with Arbitrary Cost Sharing. In the latter, the players had to specify not only the edges they wished to use, but also the payments that they would be willing to contribute to those edges. This choice is no longer in the hands of the players, since it is taken away from them by the fair Shapley-value cost sharing. Since a player has no choice in how much to pay for using an edge, it suffices for it to specify which edges it will use to connect, since the payments are implicitly defined by this choice. The Fair Connection Game, therefore, is the same as the Connection Game with Arbitrary Cost Sharing, with the player payments p_i constrained to be of the form $p_i(e) = c_e/x_e$.

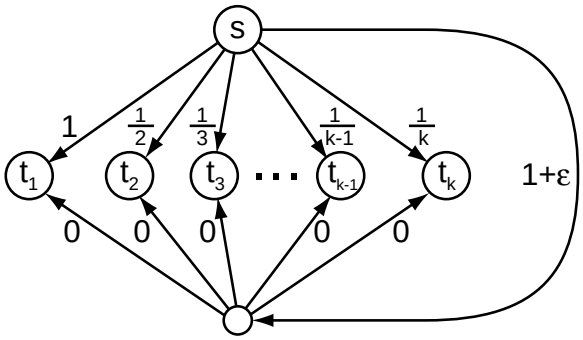

Figure 4.1: An instance in which the price of stability converges to $H(N) = \Theta(\log N)$ as $\varepsilon \to 0$.

4.2 Nash Equilibria of Network Design with Shapley Cost-Sharing

The goal of a network design protocol is to suggest for each user i a set of edges S_i so that the resulting set of built edges is in Nash equilibrium and its total cost exceeds that of an optimal network by as small a factor as possible; this factor is the *price of stability* of the instance. It is useful at this point to consider a simple example that illustrates how the price of stability can grow to a super-constant value (with N). Suppose N players wish to connect the common source s to their terminal t_i, assume player i has its own path of cost $1/i$, and all players can share a common path of cost $1 + \varepsilon$ for some small $\varepsilon > 0$ (see Figure 4.1). The optimal solution would connect all agents through the common path for a total cost of $1 + \varepsilon$. However, if this solution were offered to the users, they would defect from it one by one to their alternate paths. The unique Nash equilibrium has a cost of $\sum_{i=1}^{N} \frac{1}{i} = H(N)$.

While the price of stability in this instance grows with N, it only does so logarithmically. It is thus natural to ask how bad the price of stability can be for this network design problem. If we think about the example in Figure 4.1 further, it is also interesting to note that a good Nash equilibrium is reached by iterated greedy updating of players' solutions (in other words, best-response dynamics) starting from an optimal solution; it is natural to ask to what extent this holds in general.

In the worst case, Nash equilibria can be very expensive in this game, so that the price of anarchy becomes as large as N. To see this, consider N players with common source s and sink t, and two parallel edges of cost 1 and N. The worst equilibrium has all players selecting the more expensive edge, thereby paying N times the cost of the optimal network. However, we can bound the price of stability by $H(N)$, which is the harmonic sum $1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{N}$, as follows.

Theorem 4.2.1 *The price of stability of the fair connection game is at most $H(N)$.*

Proof. The fair connection game that we have defined falls into the class of congestion games as defined by Monderer and Shapley [62], as the cost of an edge e to a user i is $f_e(x) = c_e/x$, which depends only on edge e and the number of users x whose strategy contains e. In a *congestion game*, there is a set S of resources, and the possible strategies of players are certain subsets of S of resources. If the players select strategies $S = (S_1, \ldots, S_N)$, then for an element $s \in S$, let x_s denote the number of players whose strategy contains s. Now the cost of player i is $\sum_{s \in S_i} f_s(x_s)$, for some function s. The important point is that the cost player i pays for using a resource s depends only on the number of players using the resource. In the case of our fair connection game, S is the set of edges in G, and $f_e(x) = c_e/x$ for each edge e. Monderer and Shapley [62] show that all congestion games have deterministic Nash equilibria. They prove this using a

potential function Φ, defined as follows.

$$\Phi(S) = \sum_{e \in E} \sum_{x=1}^{x_e} f_e(x) \tag{4.1}$$

Monderer and Shapley show that for any strategy $S = (S_1, \ldots, S_N)$ if a single player i deviates to strategy S_i', then the change in the potential value $\Phi(S) - \Phi(S')$ of the new strategy set $S' = (S_1, \ldots, S_i', \ldots, S_N)$ is exactly the change in the cost to player i. Note that the change of player i's strategy affects the cost of many other players $j \neq i$, but the Φ value is not effected by the change in the cost of these players, it simply tracks the cost of the player who changes its strategy. They call a game in which such a function Φ exists a *potential game.* To show that such a potential game has a deterministic Nash equilibrium, start from any state $S = (S_1, \ldots, S_N)$ and consider a sequence of selfish moves (allowing players to change strategies to improve their costs). In a congestion game any sequence of such improving moves leads to a Nash equilibrium as each move decreases the potential function Φ, and hence must lead to a stable state.

Monderer and Shapley do not say anything about the quality of Nash equilibria with respect to the centralized optimum, but we can use their potential function to establish our bound. Let x_e be defined as above with respect to S. Now the potential function of Equation 4.1 in our case is $\Phi(S) = \sum_{e \in E} c_e H(x_e)$. According to the above argument, any improving deviation decreases $\Phi(S)$, and so a sequence of improving deviations by players must eventually result in a Nash equilibrium.

Consider the strategy $S^* = (S_1^*, \ldots, S_N^*)$ defining the optimal centralized solution. Let $OPT = \sum_{e \in S^*} c_e$ be the cost of this solution. Then, $\Phi(S^*) \leq \sum_{e \in S^*} (c_e \cdot H(N))$, which is exactly $H(N) \cdot OPT$. Now we start from strategy S^* and follow a sequence of improving self-interested moves. We know that this will result in a Nash equilibrium S with $\Phi(S) \leq \Phi(S^*)$.

Note that the potential value of any solution S is at least the total cost: $\Phi(S) \geq \sum_{e \in S} c_e = cost(S)$. Therefore, there exists a Nash equilibrium with cost at most $H(N) \cdot OPT$, as desired. □

Recall from above that this bound is tight as shown by the example in Figure 4.1. Unfortunately, even though Theorem 4.2.1 says that cheap Nash equilibria exist, finding them is NP-complete.

Theorem 4.2.2 *Given an instance of a fair connection game and a value C, it is NP-hard to determine if the game has a Nash equilibrium of cost at most C.*

Proof. The reduction is from 3D-Matching. Given an instance of 3D-Matching with node sets X, Y, Z, form a graph $G = (V, E)$ as follows. Form a node for each node in X, Y, and Z, and also a node $v_{i,j,k}$ for each 3D edge (x_i, y_j, z_k). Also add an additional node t. Form a directed edge from each node $v_{i,j,k}$ to t with cost function $c_e = 3$. Form a directed edge from each node v in X, Y, Z to all nodes representing 3D edges that contain v. Make these edges have a cost $c_e = 0$. Let $C = |X| + |Y| + |Z|$, and form a player for each node in v in $X \cup Y \cup Z$. This player has two terminals: v and t.

If there exists a 3D Matching in the 3D-Matching instance, then there exists a Nash equilibrium in the above fair connection game of cost C: Take the 3D Matching M, and let S_i for the player whose terminals are v and t be the edge from v to the unique node $v_{i,j,k}$ corresponding to the 3D edge in M, and the edge from this node to t. Since M is a matching, the cost of S is exactly $3C/3 = C$. S is a Nash equilibrium, since any deviation for a player involves paying for some edge of cost 3 by himself, while the current amount he is paying is 1.

If no 3D Matching exists, then any solution to the fair connection game must cost more than C. Therefore, no Nash equilibrium can exist of cost at most C. This finishes the proof. □
Notice that the same proof works to show that determining if there exists a Nash equilibrium that costs as little as OPT is NP-complete.

We can extend the results of Theorem 4.2.1 to concave cost functions. Consider the extended fair connection game where instead of a constant cost c_e, each edge has a cost which depends on

the number of players using that edge, $c_e(x)$. We assume that $c_e(x)$ is a nondecreasing, concave function, modeling the buy-at-bulk economy of scale of buying edges that can be used by more players. Notice that the cost of an edge $c_e(x)$ might increase with the number of players using it, but the cost per player $f_e(x) = c_e(x)/x$ decreases if $c_e(x)$ is concave.

Theorem 4.2.3 *Take a fair connection game with each edge having a nondecreasing concave cost function $c_e(x)$, where x is the number of players using edge e. Then the price of stability is at most $H(N)$.*

Proof. The proof is analogous to the proof of Theorem 4.2.1. We use the potential function $\Phi(S)$ defined by (4.1). As before, the change in potential if a player i deviates equals exactly to the change of that player's payments. We start with the strategy S^* with minimum total cost, and perform a series of improving deviations until we reach a Nash equilibrium S with $\Phi(S) \leq \Phi(S^*)$. To finish the proof all we need to show is that $cost(S) \leq \Phi(S) \leq H(N) \cdot cost(S)$ for all strategies S. The second inequality follows since $c_e(x)$ is nondecreasing and therefore $\sum_{x=1}^{x_e}(c_e(x)/x) \leq H(x_e) \cdot c_e(x_e)$. To see that $cost(S) \leq \Phi(S)$ notice that since $c_e(x)$ is concave, the cost per player must decrease with x, i.e. $c_e(x)/x$ is a nonincreasing function. Therefore, $cost(S) = \sum_{e \in S} c_e(x_e) = \sum_{e \in S} x_e \cdot (c_e(x_e)/x_e) \leq \Phi(S)$, which finishes the proof. □

Notice that the condition that cost functions be concave is general enough to encompass the utility function of a player being a combination of the cost he has to pay for his edges and the distance between his terminals in the network of bought edges. If c_e is the cost function of an edge, we simply set $c'_e(x) = c_e(x) + x$. The payment of each player i now becomes $|S_i| + \sum_{e \in S_i}(c_e(x_e)/x_e)$, and c'_e is still concave if c_e is concave.

Extensions The proof of Theorem 4.2.3 extends to a general congestion game, where players attempt to share a set of resources R they need. Instead of having an underlying graph structure, we now think of each $s \in R$ as a resource with a concave cost function $c_s(x)$ of the number of users selecting sets containing s. The possible strategies of each player i is a set $\mathcal{S}_i$ of subsets of R. Each player seeks to select a set $S_i \in \mathcal{S}_i$ so as to minimize his cost. Since the proofs above did not rely on the graph structure, they translate directly to this extension.

We can further extend the results to the case when the cost to a player is a combination of the cost $c_e(x)/x$, and a function of the selected set, such as the distance between terminals in the network design case. More precisely, the price of stability is still at most $H(N)$ if each player is trying to minimize the cost $\sum_{e \in S_i}(c_e(x_e)/x_e) + d_i(S_i)$ where c_e is monotone increasing and concave, and d_i is an arbitrary function specific to player i (e.g. a distance function, or diameter of S_i, etc.). The proof is analogous to Theorem 4.2.3, except with a new potential $\Phi(S) = \sum_i d_i(S_i) + \sum_{e \in S} \sum_{x=1}^{x=x_e} \frac{c_e(x)}{x}$. Notice that this is technically not a congestion game on the given graph G. Finally we note that all these results (as well as those subsequent) hold in the presence of capacities. Adding capacities u_e to each edge e and disallowing more than u_e players to use e at any time does not substantially alter any of our proofs.

4.3 The Case of Undirected Graphs

While the bound of $H(N)$ on the price of stability is tight for general directed graphs with costs, it is not tight for undirected graphs. Finding the correct bound is an interesting open problem. In the case of two players, our bound on the price of stability is $H(2) = 3/2$. We now show that this bound can be improved to $4/3$ in the case of two players and a single source.

Here is an example of an undirected two-player game with the price of stability approaching $4/3$. Let G have 3 nodes: s, t_1, and t_2. Player 1 wants to connect t_1 with s, and player 2 wants to connect t_2 with s. There are edges (s, t_1) and (s, t_2) with cost 2. There is an edge (t_1, t_2) with cost $1 + \varepsilon$. The optimal centralized solution has cost $3 + \varepsilon$. However, the cheapest Nash has cost 4. This example implies that the following claim is tight.

Claim 4.3.1 *The price of stability is at most 4/3 in a fair connection game with two players in an undirected graph, each having two terminals with one terminal in common.*

Proof. Let s be the common terminal, and let t_1 and t_2 be the personal terminals. Consider the optimal centralized solution (S_1, S_2). Let $X_1 = S_1 \backslash S_2$ be the edges only being used by player 1, $X_2 = S_2 \backslash S_1$ be the edge only used by player 2, and $X_3 = S_1 \cap S_2$ be the edges shared by the two players. Let (S_1', S_2') be a Nash equilibrium that a series of improving responses converges to starting with (S_1, S_2). Similarly, let $Y_1 = S_1' \backslash S_2'$, $Y_2 = S_2' \backslash S_1'$, and $Y_3 = S_1' \cap S_2'$. Finally, set $x_i = cost(X_i)$ and $y_i = cost(Y_i)$ for $1 \leq i \leq 3$. By the properties of $\Phi(S_1, S_2)$ from the proof of Theorem 4.2.1, we know that $\Phi(S_1', S_2') \leq \Phi(S_1, S_2)$. Substituting in the definition of Φ, we obtain that

$$y_1 + y_2 + \frac{3}{2}y_3 < x_1 + x_2 + \frac{3}{2}x_3. \tag{4.2}$$

Look at S_1' and S_2' as paths instead of sets of edges (there will be no cycles since then this would not be a Nash). We now show that in (S_1', S_2'), as in any Nash equilibrium, once the paths of the two players merge, they do not separate again. Suppose to the contrary that this happens. Let v be the first node that S_1' and S_2' have in common, and set P_1 and P_2 be the subpaths of S_1' and S_2' after v, respectively. We know that $cost(P_1 \backslash P_2) = cost(P_2 \backslash P_1)$, since if they were not equal, say $cost(P_1 \backslash P_2) > cost(P_2 \backslash P_1)$, then player 1 could deviate to P_2 instead and pay strictly less. However, even if they are equal, player 1 could deviate to use P_2 instead of P_1, and pay strictly less, since he will pay the same as before on edges in $P_1 \cap P_2$, and pay only $cost(P_1 \backslash P_2)/2$ in total on the other edges. Therefore, the only way this could be a Nash equilibrium is if $P_1 \cap P_2 = P_1 = P_2$, as desired.

Consider a deviation from (S_1', S_2') that player 1 could make. He could decide to use $X_1 \cup X_2 \cup Y_2 \cup Y_3$ instead of $S_1' = Y_1 \cup Y_3$. This is a valid deviation because player 1 still connects his terminals by following X_1 until X_1 meets with X_2, then following X_2 back to t_2, and then following S_2' to s. Since (S_1', S_2') is a Nash equilibrium, this deviation must cost more to player 1 than his current payments, and so $x_1 + x_2 + y_2/2 + y_3/2 \geq y_1 + y_3/2$. By symmetric reasoning, $x_1 + x_2 + y_1/2 + y_3/2 \geq y_2 + y_3/2$. If we add these inequalities together, we obtain that

$$y_1/2 + y_2/2 \leq 2x_1 + 2x_2. \tag{4.3}$$

To show that the price of stability is at most 4/3, it is enough to show that $cost(S_1', S_2') \leq \frac{4}{3} cost(S_1, S_2)$. Using the above notation, this is the same as showing $3y_1 + 3y_2 + 3y_3 \leq 4x_1 + 4x_2 + 4x_3$. We do this by using Inequalities 4.2 and 4.3 as follows:

$$\begin{aligned} 3y_1 + 3y_2 + 3y_3 &\leq 3y_1 + 3y_2 + 4y_3 \\ &= \frac{1}{3}(y_1 + y_2) + \frac{8}{3}(y_1 + y_2 + \frac{3}{2}y_3) \\ &\leq \frac{4}{3}(x_1 + x_2) + \frac{8}{3}(x_1 + x_2 + \frac{3}{2}x_3) \\ &= 4x_1 + 4x_2 + 4x_3 \end{aligned}$$

□

4.4 Convergence of Best Response

In this section, we address the convergence properties of best response dynamics in our game. Notice that even though we have shown above that best response dynamics always converge to a Nash equilibrium within a factor of log of the optimal centralized solution, there are cases where best response dynamics actually start out close to the optimum Nash equilibrium and yet end up in a costly one (although still within a log factor). For example, consider the same graph as in Figure 4.1, but with an extra path that can be shared by all players of cost 1. If the starting configuration is that all players are using the path of cost $1 + \varepsilon$, then there is a sequence of best

responses which ends up in a Nash equilibrium of cost $\log N$ (where N is the number of players), even though the starting configuration was of very similar cost to both the centralized optimum and the best Nash equilibrium.

Theorem 4.4.1 *In the two player fair connection game, best response dynamics starting from any configuration converges to a Nash equilibrium in polynomial time (even if edge costs are concave functions).*

Proof. Suppose we start from any configuration C_0. Suppose for $i \geq 1$, the configurations $\{C_i\}$ are obtained by alternating the best responses of the two players. $P_i(1,2)$ refers to the shared path of the two players.

We show inductively that for $i \geq 2$, $P_i(1,2)$ is a contiguous path and that $P_{i+1}(1,2) \supseteq P_i(1,2)$. The base case is showing that $P_2(1,2)$ is a contiguous path. Without loss of generality, assume that the sequence of best responses are as follows

$$C_0 \xrightarrow{2} C_1 \xrightarrow{1} C_2 \xrightarrow{2} 2 \ldots$$

Assume that $P_2(1,2)$ is not contiguous, and since player 1 was the last player to have done best response in reaching C_2 it follows that he did not choose a strategy which results in the shared segment being contiguous in C_2. But now, we use this fact to analyze the last response of player 2, who started from C_0. Since player 1 was able to take shortcuts across segments of player 2's path, we can construct a better response for player 2 starting from C_0, which is a contradiction.

In the inductive step, we have to show that for any configuration C_{i+1}, the edges $P_{i+1}(1,2)$ are contiguous and $P_{i+1}(1,2) \supseteq P_i(1,2)$. The fact that $P_{i+1}(1,2)$ is a contiguous path follows essentially from the same proof as in the base case. Given that, we now have to consider only the strategies in Figure 4.2, and show that these deviations will never occur.

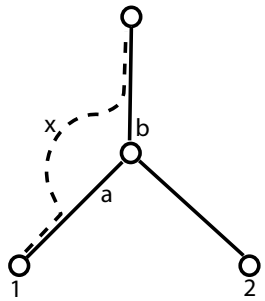

Figure 4.2: The only possible deviations for a two-player game.

Indeed if player 1 decides to take the path as in the figure, taking the shortcut through path x, then

$$c_x(1) < c_a(1) + c_b(2)/2.$$

By inductive hypothesis, the shared part only grew till now, so when player 1 had last done best response from configuration C_{i-1}, player 2 could not have been on any edges of a. So it must have been the case that

$$c_x(1) > c_a(1) + c_b(2)/2$$

which is a contradiction (we assume here that the costs are concave). Hence, this is not a valid deviation for player 1. Thus, either $P_{i+1}(1,2) = P_i(1,2)$ or $|P_{i+1}(1,2)| > |P_i(1,2)|$. But note that the two paths $P_i(1) - P_i(1,2)$ and $P_i(2) - P_i(1,2)$ are always shortest paths and so $P_{i+1}(1,2) = P_i(1,2)$ implies we have reached a Nash. Else $P_i(1,2)$ strictly increases by at least one edge. Hence, we reach a Nash in polynomial number of steps. □

The above proof shows that for any best response run, the number of edges shared by both players increases monotonically. For more players, however, the hope of any positive result about best response dynamics seems slim. In fact, we can show the following.

Theorem 4.4.2 *Best response dynamics for N players may run in time exponential in N.*

To prove this, we now construct an example (shown in Figure 4.3) in which by appropriate ordering of the best response of players, we can simulate a $O(N)$-bit counter.

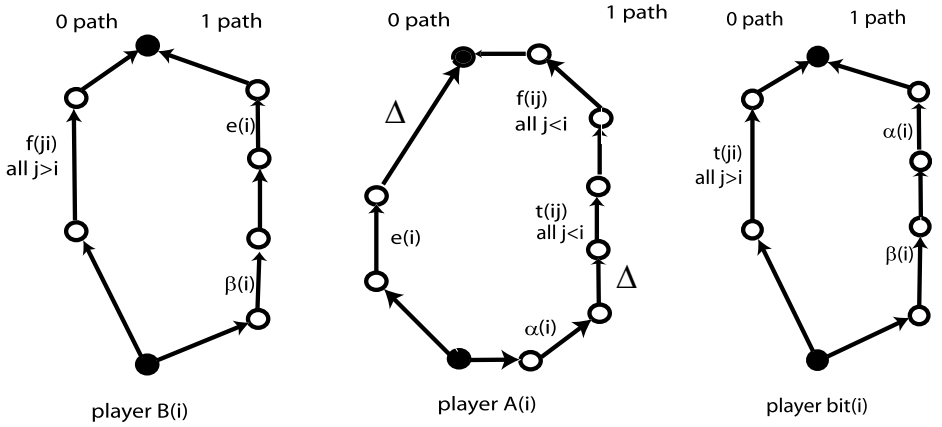

Figure 4.3: The construction of an exponential best response run.

The graph has $3n$ players, n "bit" players, each being assisted by two auxiliary players. The auxiliary players of the i^{th} player are denoted by $A(i)$ and $B(i)$. Each bit player and each auxiliary player has only two path options, we call these the 0 path and the 1 path. Note that the different players share edges, which are identified by the label that is on them. The only exception is for the edges that are labeled Δ, which is not the name but the cost of the edge. The same edge $f(j,i)$, for instance, is present in the 0-path of all $B(i)$ players when $j > i$ and also in the 1 path of all the $A(j)$ players such that $j > i$. The unnamed edges, on the other hand are specific to each gadget and are not shared. In addition to the unnamed edges drawn in the gadget, assume there are unnamed edges before and after every edge type of $f(i,j)$ or $t(i,j)$ and these unnamed edges are directed and specific to a particular gadget. All the unnamed edges are zero cost, except for the two edges, one in each path of $A(i)$ that each have cost Δ, which is large, say $3n$ times bigger than the sum of the cost of all the named edges. Thus player $A(i)$ always pays at least Δ in order to get to the sink, but never should agree to pay more than even $\Delta + \Delta/3n$.

In the above Figure, we have not specified the order in which the edges of, for example, $t(j,i)$ for all $j > i$ in the gadget of $bit(i)$ proceed. In all of these cases, there is one index that is fixed, and one that is variable, so we assume that these edges appear in increasing order of the variable index.

We also refer to the player going on the one path as the player being set and going on the zero path as the player being reset. Each player has one source and one sink and the paths of each player are as shown in the gadgets above. The costs of the paths of i^{th} bit player are referred to as $x_i^{(0)}$ and $x_i^{(1)}$, and those of the player $A(i)$ and $B(i)$ as $a_i^{(0)}$, $a_i^{(1)}$ and $b_i^{(0)}$, $b_i^{(1)}$ respectively. Here we describe how the counter works and the inequalities that should hold for it to work properly.

Start Step : All the players are reset.

General Step : At the start of this step, the bits from 1 to $l-1$ are all set. The bits from $l+1$ to n maybe at 0 or 1. The l^{th} bit is currently at 0 and has to be set at 1. Also, all the $A(j)$ players

are reset. The $B(j)$ players are set if and only if the j'th players are set.

- Now, the l^{th} bit sets. This triggers both $A(l)$ and $B(l)$.
  At this point, there are no other players using either of bit l player's paths, so for this to happen, we need only the following to be true. In writing the deviation requirements that we need, for clarity's sake, we first write it in a "short" and (arguably) more understandable form. In the next line we expand it in terms of the constituent edges.

$$\begin{aligned} x_l^{(1)} &< x_l^{(0)} \\ \alpha(l) + \beta(l) &< \sum_{j>l} t(j,l) \end{aligned}$$

- The cost of 1 path of $A(l)$ has now decreased by $\alpha(l)/2$ because of this bit and other players that were already there. $A(l)$ is triggered and is allowed to set. Since both the 0 and 1 path have the Δ cost edges that are to be paid by $A(i)$ alone, they do not matter. Thus,

$$\begin{aligned} a_l^{(1)} - \alpha(l)/2 &< a_l^{(0)} \\ \alpha(l)/2 + \sum_{j<l} f(l,j) + \sum_{j<l} t(l,j) &< e(l) \end{aligned}$$

- The setting of $A(l)$ triggers all the $B(i)$ for $i < l$ to be reset. Recall that the corresponding $A(i)$ are already reset. We allow these $B(i)$ to reset. Due to the triggering, the cost of 0 path has changed by $f_{li}/2$. We also need to take into account that the 1 path is possibly shared by others too.

$$\begin{aligned} b_i^{(0)} - f(l,i)/2 &< b_i^{(1)} - e(i)/2 - \beta(i)/2 \\ \sum_{j>i} f(j,i) - f(l,i)/2 &< e(i)/2 + \beta(i)/2 \end{aligned}$$

- $A(l)$ also triggers all the bits $i < l$ to reset by reducing the cost of the 0 path of bit-i by $t(l,i)/2$. We allow that too.

$$\begin{aligned} x_i^{(0)} - t(l,i)/2 &< x_i^{(1)} \\ \sum_{j>i} t(j,i) - t(l,i)/2 &< \alpha(i) + \beta(i) \end{aligned}$$

- Now because of the setting of the bit-l, the 1 path of B_l became cheaper by $\beta(l)/2$. $B(l)$ wants to set and is allowed to.

$$\begin{aligned} b_l^{(1)} - \beta_l/2 &< b_l^{(0)} \\ e(l) + \beta(l)/2 &< \sum_{j>l} f(j,l) \end{aligned}$$

- Lastly, as a result of the setting of $B(l)$, the 0 path of $A(l)$ became cheaper by $e(l)/2$. $A(l)$ now wants to reset. The 1 path of $A(l)$ is possibly shared by other players. Again we ignore the edges of cost Δ.

$$\begin{aligned} a_l^{(0)} - e(l)/2 &< a_l^{(1)} - \sum_{j<l} (f(l,j)/2 + t(l,j)/2) - \alpha(l)/2 \\ e(l)/2 &< \alpha(l)/2 + \sum_{j<l} (f(l,j)/2 + t(l,j)/2) \end{aligned}$$

- Now we have the subgame from 1 to $l-1$ being completely reset, and no other player from the top part influencing any of their paths. So we can play their complete game and come back to the configuration in the start of the recursion, except now we need to deal with the $(l+1)^{st}$ bit.

Proof of Theorem 4.4.2. We now prove that the above game has an exponential best response run under the above best response scheduling.

All we need to show is that the moves described in the scheduling are best responses. We first argue that each player has only two possible paths available to him, which we have described as the zero path and the one path. To complete the construction we next need to come up with a set of values for the links that satisfy the set of best response inequalities. Taken together, it follows that the moves are all best responses and simulate an exponential length counter.

First note that the 0 path and 1 path of any one particular player are vertex disjoint. The unnamed edges also impose the following property on the 0/1-paths of all players: the 0/1-path of any one player must either have at least one edge in common or must be vertex disjoint from the 0/1-path of any other player. We do not have to worry about the case of just having vertices in common.

We need to argue that each player has only the two paths available to him. Intuitively the argument is as follows. If one player deviates out of his gadget, he will never be able to come back to his own sink. Thus, the only available strategies to a player are the 0 and 1 paths.

Before going into the details, we first define a function to make the notation simpler. For any edge e, define the function $sink(e)$ to be the set of all possible sinks that paths from this edge can lead to. To compute the $sink(f(x,y))$ note that $f(x,y)$ can lead to the sink of $A(x)$ and $B(y)$ if $x > y$, as well as to $sink(f(x+1,y))$ and $sink(f(x,y+1))$. Inductively, if $x > y$

$$sink(f(x,y)) = \{A(x), \ldots, A(n), B(y), \ldots, B(n)\}$$

and $\emptyset$ otherwise.

To compute $sink(t(x,y))$, note that $t(x,y)$ can lead to $A(x)$, $bit(y)$, and $sink(f(x,1))$, as well as $sink(t(x+1,y))$ and $sink(t(x,y+1))$. Thus, if $x > y$

$$sink(t(x,y)) = \{bit(y), \ldots, bit(n), A(x), \ldots, A(n), B(1), \ldots, B(n)\}$$

and $\emptyset$ else.

We now flesh out the argument for each player for each of the two cases, the set-case and the reset-case. Consider player $B(i)$. If a strategy follows the first unnamed edge of 1-path, then it reaches $\beta(i)$. From the end vertex of $\beta(i)$ it has the option of choosing the next unnamed edge in 1-path of gadget $B(i)$, or in the 1-path of $bit(i)$. In the first case the strategy has to go through $e(i)$. But after it crosses $e(i)$ there only is a zero-cost edge left to be covered to the sink of $B(i)$, and so other deviations are useless. If the path instead had chosen to enter the gadget of $bit(i)$ after the edges $\beta(i)$, then it would have to travel through $\alpha(i)$ and then either get stuck at the sink of $bit(i)$ or travel through one or more edges of the form $\{t(i,y), y < i\}$. But in order to enter an edge of the form $t(i,y)$ this strategy would have to pay at least $\Delta/3n$ for the edge priced Δ, which is more than the budget of $B(i)$. So any strategy starting on the 1-path of $B(i)$ does not enter the $A(i)$ gadget.

Next consider a strategy starting out on the 0-path of $B(i)$ through the first unnamed edge. The edges then appearing on this path are of the form $\{f(x,i), x > i\}$. These edges also appear in the gadgets $A(x)$. If this strategy chooses to deviate to exit the $B(i)$ gadget after the edge $f(x,i)$ and enter the $A(x)$ one, the labeled edge that it meets next is $f(x,i+1)$. But no path from $f(x,i+1)$ leads to the sink of $B(i)$, as is verified from the sink-function above. So this deviation cannot happen. We have now shown that the best strategies of $B(i)$ must be the 0 and 1 paths only.

Now, take the $A(i)$ player and a strategy which starts out on the first unnamed edge of the 1-path. This strategy then has to go through $\alpha(i)$. From the endpoint of $\alpha(i)$ the only edges

are to the sink of $bit(i)$, or continue along the $A(i)$ gadget. The edges that appear next are of the form $t(i, x)$. If the path does not continue along the gadget of $A(i)$, and instead switches to the 0-path of $bit(x)$, then the next edge it encounters is $t(i + 1, x)$. But according to the sink function, there is no way to reach the sink of $A(i)$ from $t(i+1, x)$, so this is not a valid deviation. Next, the path cannot deviate out of any the $\{f(i, x), x < i\}$ edges since $A(i)$ does not lie in any $sink(f(i + 1, x))$. Thus strategies of $A(i)$ starting out on the 1-path of $A(i)$ stay on this path.

Next consider the strategy of $A(i)$ that starts out through the unnamed edge in the 0-path of $A(i)$. This path cannot deviate after $e(i)$ because it would just get stuck in the sink of $B(i)$. Thus, we have shown that the best strategies of $A(i)$ must be the 0 and 1 paths only.

Lastly consider the i-th bit player. Along $x_i^{(0)}$, the only shared nodes and edges of this player are the $t(x, i)$ edges that are each shared with $A(x)$ for all $x > i$. Suppose this bit player follows the $t(x, i)$ edge to the gadget of $A(x)$ with $x > i$. The next edge is then $t(x, i + 1)$ which does not lead to $bit(i)$-sink. Thus the 1-path strategies do not allow any deviations. A path starting with the first edge of the 0-path of this bit-player, can deviate after $\beta(i)$ and enter edge $e(i)$ of the $B(i)$ gadget. But from there all paths get stuck at sinks. Thus the 0-path deviations cannot happen after $\beta(i)$. But the deviations cannot happen after $\alpha(i)$ either, as the remaining edge is zero cost.

This concludes the proof that the best strategies of the players always correspond to the 0 and 1 paths in the gadgets.

For the last part of the construction, we show that it is possible to come up with a set of values for the links such that the best response inequalities are satisfied. Let the edge costs be as follows. In all the remaining formulae let c be any constant greater than 10. Let $\alpha(i) = 1$, and $\beta(i) = 2c^i(n - i) - c^i/2 - 3e(i)/2$. Also, $e(i) = \sum_{j<i} f(i, j) + \sum_{j<i} t(i, j) + 3/4$ for all i. Finally, for all pairs i, j, $f(i, j) = c^j$ and $t(i, j) = 2c^j$. Given these values, we can check that the inequalities above are satisfied for all $i < n$, and thereby we can have a run of best responses of length exponential in the number of players. □

4.5 Weighted Players

So far we have assumed that players sharing an edge e pay equal fractions of e's cost. We now consider a game with fixed edge costs where players have weights $w_i \geq 1$, and players' payments are proportional to their weight. More precisely, given a strategy $S = (S_1, \ldots, S_N)$, define W to be the total weight of all players, and let W_e be the sum of the weights of players using e. Then player i's payment for edge e will be $\frac{w_i}{W_e} c_e$.

Note that the potential function $\Phi(S)$ used for the unweighted version of the game is not a potential function once weights are added. In particular, in a weighted game, improving moves can increase the value of $\Phi(S)$, as this is no longer a congestion game. The following theorem uses a new potential function for a special class of weighted games.

Theorem 4.5.1 *In a weighted game where each edge e is in the strategy spaces of at most two players, there exists a potential function for this game, and hence a Nash equilibrium exists.*

Proof. Consider the following potential function. For each edge e used by players i and j, define

$$\Phi_e(S) = \begin{cases} c_e w_i & \text{if player } i \text{ uses } e \text{ in } S \\ c_e w_j & \text{if player } j \text{ uses } e \text{ in } S \\ c_e \theta_{ij} & \text{if both players } i \text{ and } j \text{ use } e \text{ in } S \\ 0 & \text{otherwise} \end{cases}$$

where $\theta_{ij} = (w_i + w_j - \frac{w_i w_j}{w_i + w_j})$. For any edge e with only one player i, simply set $\Phi_e(S) = w_i c_e$ if i uses e and 0 otherwise. Define $\Phi(S) = \sum_e \Phi_e(S)$. We now simply need to argue that if a player makes an improving move, then $\Phi(S)$ decreases. Consider a player i and an edge e that player i

joins. If the edge already supported another player j, then i's cost for using e is $c_e \frac{w_i}{w_i+w_j}$, while the change in $\Phi_e(S)$ is

$$c_e(w_i - \frac{w_i w_j}{w_i + w_j}) = c_e \frac{{w_i}^2}{w_i + w_j}.$$

Thus the change in potential when i joins e equals the cost i incurs, scaled up by a factor of w_i. In fact, it is easy to show the more general fact that when player i moves, the change in $\Phi(S)$ is equal to the change in player i's payments scaled up by w_i. This means that improving moves always decrease $\Phi(S)$, thus proving the theorem. □

Note that this applies not only to paths, but also to the generalized model in which players select subsets from some ground set. The analogous condition is that no ground element appears in the strategy spaces of more than two players.

Corollary 4.5.2 *Any two-player weighted game has a Nash equilibrium.*

While the above potential function also implies a bound on the price of stability, even with only two players this bound is very weak. However, if there are only two players with weights 1 and $w \geq 1$, then we can show that the price of stability is at most $1 + \frac{1}{1+w}$, and this is tight for all w.

The following result shows the existence of Nash equilibria in weighted single commodity games.

Theorem 4.5.3 *For any weighted game in which all players have the same source s and sink t, best response dynamics converges to a Nash equilibrium, and hence Nash equilibria exist.*

Proof. Start with any initial set of strategies S. For every $s-t$ path P define the marginal cost of P to be $c(P) = \sum_{e \in P} \frac{c_e}{W_e}$ where W_e depends on S. Observe that if player i currently uses path P, then i's payment is $w_i c(P)$. Define $P(S)$ to be a tuple of the values $c(P)$ over all paths P, sorted in increasing order. We want to show that the cheapest improving deviation of any player causes $P(S)$ to strictly decrease lexicographically.

Suppose that one of the best moves for player i is to switch paths from P_1 to P_2. Let $\mathcal{P}$ denote the set of paths that intersect $P_1 \cup P_2$. For any pair of paths P and Q, let $c_P(Q)$ denote the new value of $c(Q)$ after player i has switched to path P. To show that $P(S)$ strictly decreases lexicographically, it suffices to show that

$$\min_{P \in \mathcal{P}} c_{P_2}(P) < \min_{P \in \mathcal{P}} c(P). \tag{4.4}$$

Define $P' = \arg\min_{P \in \mathcal{P}} c(P)$. Since P_2 was i's best response, $c_{P_2}(P_2) \leq c_P(P)$ for all paths P. In particular, $c_{P_2}(P_2) \leq c_{P'}(P')$. We also know that $c_{P'}(P') \leq c(P')$, since in deviating to P', player i adds itself to some edges of P'. In fact, $c_{P'}(P') < c(P')$ unless $P' = P_1$. Assuming $P' \neq P_1$, we now have that $c_{P_2}(P_2) < c(P')$, which proves inequality 4.4. If $P' = P_1$, then since player i decided to deviate, $c_{P_2}(P_2) < c(P_1)$. Therefore, we once again have that $c_{P_2}(P_2) < c(P')$, as desired. □

In the case where the graph consists of only 2 nodes s and t joined by parallel links, we can similarly show that any sequence of improving responses converge to a Nash equilibrium.

With arbitrarily increasing cost functions, [55] gives an example demonstrating that a weighted game may not have any pure Nash equilibria. Recently, Chen and Roughgarden [18] proved general results on the price of stability and the existence of approximate Nash equilibria in the weighted game we present. They show that there exist weighted games with fixed edge costs and no pure Nash equilibria, settling an open question from [6]. They also prove that any weighted game has a $O(\log w_{\max})$-approximate Nash equilibrium that costs at most $O(\log W)$ times the centralized optimum (where $w_{\max}$ is the maximum weight). Their results, together with the following theorem, show that this upper bound is essentially tight.

Theorem 4.5.4 *There are weighted games for which the price of stability is $\Theta(\log W)$ and $\Theta(N)$.*

An example exhibiting this is a modified version of the graph in Figure 4.1. Change the edge with cost $1+\varepsilon$ to cost 1, and for all other edges with positive cost, set the new cost to be $\frac{1}{2}$. For $1 \leq i \leq N$ let player i have weight $w_i = 2^{i-1}$. Since each player has a greater weight than all smaller weight players combined, the only Nash equilibrium has cost $\frac{N}{2} = \Theta(\log W)$, while the optimal solution has cost 1.

Chapter 5
Self-Interested Routing with Atomic Demands

In all of the previous chapters, we assumed that the utility of a player depends only on the cost of the edges he uses. What changes if we introduce latency into the picture? We have extended our resutls to the case when the players' cost is a combination of "design" cost and the length of the path selected. More generally, delay on an edge does not have to be simply the "hop-count", but can also depend on congestion, i.e., on the number of players using the edge. In this chapter we will consider such a model with fair cost sharing, as in the previous chapter. Many of the results in the chapter initially appeared in [6].

Assume that each edge has both a cost function $c_e(x)$ and a latency function $d_e(x)$, where $c_e(x)$ is the cost of building the edge e for x users and the users will share this cost equally, while $d_e(x)$ is the delay suffered by users on edge e if x users are sharing the edge. The goal of each user will be to minimize the sum of his cost and his latency. If we assume that both the cost and latency for each edge depend only on the number of players using that edge, then this fits directly into our model of a congestion game above: the total cost felt by each user on the edge is $f_e(x) = c_e(x)/x + d_e(x)$. If the function $xf_e(x)$ is concave then Theorem 4.2.3 applies. But while concave functions are natural for modeling cost, latency tends to be convex.

5.1 Combining Costs and Delays

First, we extend the argument in the proof of Theorem 4.2.3 to general functions f_e. The most general version of this argument is expressed in the following theorem.

Theorem 5.1.1 *Consider a fair connection game with arbitrary edge-cost functions f_e. Suppose that $\Phi(S)$ is as in Equation 4.1, with $cost(S) \leq A \cdot \Phi(S)$, and $\Phi(S) \leq B \cdot cost(S)$ for all S. Then, the price of stability is at most $A \cdot B$.*

Proof. Let S^* be a strategy such that S_i^* is the set of edges i uses in the centralized optimal solution. We know from above that if we perform a series of improving deviations on it, we must converge to a Nash equilibrium S' with potential value at most $\Phi(S^*)$. By our assumptions, $cost(S') \leq A \cdot \Phi(S') \leq A \cdot \Phi(S^*) \leq AB \cdot cost(S^*) = AB \cdot OPT$. □

Our main interest in this section are functions $f_e(x)$ that are the sums of the fair share of a cost and a delay, i.e., $f_e(x) = c_e(x)/x + d_e(x)$. We will assume that $d_e(x)$ is monotone increasing, while $c_e(x)$ is monotone increasing and concave.

Corollary 5.1.2 *If $c_e(x)$ is concave and nondecreasing, $d_e(x)$ is nondecreasing for all e, and $x_e d_e(x_e) \leq A \sum_{x=1}^{x_e} d_e(x)$ for all e and x_e, then the price of stability is at most $A \cdot H(N)$. In particular, if $d_e(x)$ is a polynomial with degree at most l and nonnegative coefficients, then the price of stability is at most $(l+1) \cdot H(N)$.*

Proof. For functions $f_e(x) = c_e(x)/x + d_e(x)$, both the cost and potential of a solution come in two parts corresponding to the cost c and delay d.

For the part corresponding to cost the potential over-estimates the cost by at most a factor of $H(N)$ as proved in Theorem 4.2.3. If on the delay, the potential underestimates the cost by at most a factor of A, then we get the bound of $A \cdot H(N)$ for the price of stability by Theorem 5.1.1. □

Therefore, for reasonable delay functions, the price of stability cannot be too large. In particular, if the utility function of each player depends on a concave cost and delay that is independent of the number of users on the edge, then we get that the price of stability is at most $H(N)$ as we have shown at the end of Section 4.2. If the delay grows linearly with the number of users, then the price of stability is at most $2H(N)$.

5.2 Games with Only Delays

In this section we consider games with only delay. We assume that the cost of a player for using an edge e used by x players is $f_e(x) = d_e(x)$, and d_e is a monotone increasing function of x. This cost function models delays that are increasing with congestion.

We will mostly consider the special case when there is a common source s. Each player i has one additional terminal t_i, and the player wants to connect s to t_i via a directed path. Fabrikant, Papadimitriou, and Talwar [30] showed that in this case, one can compute the Nash equilibrium minimizing the potential function Φ via a minimum cost flow computation. For each edge e they introduce many parallel copies, each with capacity 1, and cost $d_e(x)$ for integers $x > 0$. We will use properties of a minimum cost flow for establishing our results.

5.2.1 A Bicriteria Result

First we show a bicriteria bound, and compare the cost of the cheapest Nash equilibrium to that of the optimum design with twice as many players.

Theorem 5.2.1 *Consider the single source case of a congestion game with only delays. Let S be the minimum cost Nash equilibrium and S^* be the minimum cost solution for the problem where each player i is replaced by two players. Then $cost(S) \leq cost(S^*)$.*

Proof. Consider the Nash equilibrium obtained by Fabrikant et al [30] via a minimum cost flow computation. Assume that x_e is the number of users using edge e at this equilibrium. By assumption, all users share a common source s. Let $D(v)$ denote the cost of the minimum cost path in the residual graph from s to v. The length of the path of user i is at most $D(t_i)$ (as otherwise the residual graph would have a negative cycle) and hence we get that $cost(S) \leq \sum_i D(t_i)$.

Now consider a modified delay function $\hat{d}_e$ for each edge $e = (u, v)$. Define $\hat{d}_e(x) = d_e(x)$ if $x > x_e$, and $\hat{d}_e(x) = D(v) - D(u)$ if $x \leq x_e$. Note that for any edge e we have $D(v) - D(u) \leq d_e(x_e + 1)$ as the edge $e = (u, v)$ is in the residual graph with cost $d_e(x_e + 1)$. This implies that the modified delay $\hat{d}$ is monotone. For edges with $x_e \neq 0$ we also have that $d_e(x_e) \leq D(v) - D(u)$ as the reverse edge (v, u) is in the residual graph with cost $-d_e(x_e)$, so the delay of an edge is not decreased.

Now observe that, subject to the new delay $\hat{d}$, the shortest path from s to t_i is length $D(t_i)$ even in an empty network. The minimum possible cost of two paths from s to t_i for the two users corresponding to user i is then at least $2D(t_i)$ for each player i. Therefore the minimum cost of a solution with delays $\hat{d}$ is at least $2\sum_i D(t_i)$.

To bound $cost(S^*)$ we need to bound the difference in cost of a solution when measured with delays $\hat{d}$ and d. Note that for any edge $e = (u, v)$ and any number x we have that $x\hat{d}_e(x) - xd_e(x) \leq x_e(D(v) - D(u))$, and hence the difference in total cost is at most $\sum_{e=(u,v)} x_e(D(v) - D(u)) = \sum_i D(t_i)$. Using this, we get that $cost(S^*) \geq \sum_i D(t_i) \geq cost(S)$. □

Note that a similar bound is not possible for a model with both costs and delays, when additional users compensate to some extent for the price of stability. Consider a problem with two parallel links e and e' and N users. Assume on link e the cost is all design cost $c_e(x) = 1 + \varepsilon$ for a small $\varepsilon > 0$. On the other link e' the cost is all delay, and the delay with x users is $d_{e'}(x) = 1/(N - x + 1)$. The optimum solution is to use the first edge e, and it costs $1 + \varepsilon$. Note that the optimum with any number of extra users costs the same, as this is all design cost. On the other hand, the only Nash is to have all users on e', incurring delay 1, for a total cost of N.

5.2.2 Bounding the Price of Stability of Atomic Routing

Note that the $H(N)$ term in Corollary 5.1.2 comes from the concave cost c, and so the bound obtained there improves by an $H(N)$ factor when the cost consists of only delay. The results from

Corollary 5.1.2 already tell us that if the delay functions are such that $x_e d_e(x_e) \leq A \sum_{x=1}^{x_e} d_e(x)$, the the price of stability is at most A. Specifically, we know that if the delays are polynomial of degree l, then the price of stability is at most $l+1$, and therefore with linear delays the price of stability is at most 2.

Roughgarden [72] showed a tighter bound for non-atomic games. He assumed that the delay is monotone increasing, and the total cost of an edge $xd_e(x)$ is a convex function of the traffic x. He showed that for any class of such functions $\mathcal{D}$ containing all constant functions, the price of anarchy is always obtained on a two node, two link network. Let us call $\alpha(\mathcal{D})$ the price of anarchy for non-atomic games with delays from the class $\mathcal{D}$ (which is also the price of stability, since the Nash equilibrium is unique in that context). For example, Roughgarden [72] showed that for polynomials of degree at most l this bound is $O(l/\log l)$, and for linear delays it is 4/3. Here we extend this result to a single source atomic game, and thereby show tighter bounds than in Corollary 5.1.2 for the singe source case.

Theorem 5.2.2 *If in a single source fair connection game all costs are delays, and all delays are from a set $\mathcal{D}$ satisfying the above condition, then the price of stability is at most $\alpha(D)$.*

Proof. As in the proof of Theorem 5.2.1 consider the Nash equilibrium obtained via a minimum cost flow computation, and let $D(v)$ be the length of the shortest path from s to v in the residual graph. As before we have that $cost(S) \leq \sum_i D(t_i)$. Further, for each edge $e = (u, v)$ we have that $D(v) - D(u) \leq d_e(x_e + 1)$, and for edges with $x_e \neq 0$, we also have that $d_e(x_e) \leq D(v) - D(u)$.

Add to each edge $e = (u, v)$ a capacity of x_e, and augment our network by adding a parallel edge e' with constant delay $D(v) - D(u)$. Let $\hat{G}$ denote the resulting network flow problem. Note that the new capacity and the added links do not effect the equilibrium, as $d_e(x_e) \leq D(v) - D(u)$. For each edge e, the two parallel copies: edge e with new capacity x_e and edge e', can carry any number of paths at least as cheaply as the original edge e could since $D(v) - D(u) \leq d_e(x_e + 1)$, hence this change in the network can only improve the minimum possible cost. We will prove the bound in this new network by comparing the cost of the Nash equilibrium with the minimum possible cost of a (possibly fractional) flow carrying one unit of flow from s to each of the terminals t_i.

The nice property of $\hat{G}$ is that the optimum fractional flow $\hat{x}$ in $\hat{G}$ is easy to determine. Consider an edge $e = (u, v)$ that is used by $x_e \neq 0$ paths in the equilibrium. We will obtain a fractional flow $\hat{x}_e$ by splitting the corresponding x_e amount of flow between the two edges e and e'. For an edge e let $\ell_e(x) = d_e(x) + xd'_e(x)$. By assumption, $d_e(x) \leq \ell_e(x)$ for all x. For an edge e such that $\ell_e(x_e) \leq D(v) - D(u)$, we set $\hat{x}_e = x_e$, and $\hat{x}_{e'} = 0$. Otherwise, let $\hat{x}_e$ be such that $\ell_e(\hat{x}_e) = D(v) - D(u)$, and let $\hat{x}_{e'} = x_e - \hat{x}_e$.

First, we claim that $\hat{x}$ is the minimum cost fractional solution in $\hat{G}$. For all edges $e = (u, v)$ such that $\hat{x}_e \neq x_e$, we have that $\ell_e(\hat{x}_e) = D(v) - D(u)$. When $\hat{x}_e = x_e$, then we have that flow $\hat{x}_e$ is equal the capacity of the edge, and $\ell_e(\hat{x}_e) \leq D(v) - D(u)$. Therefore, if there is a negative cycle in the residual graph of $\hat{x}_e$ with constant edge costs $\ell_e(x_e)$ for e and costs $D(v) - D(u)$ for e', then this is also a negative cost cycle in G with constant edge costs $D(v) - D(u)$. This contradicts x_e being a min-cost flow with those costs, however. We can now use Lemma 5.2.3 to see that $\hat{x}_e$ is also a min-cost flow for edge costs $xd_e(x)$.

The theorem then follows, as on each original edge $e \in E$ the cost $x_e d_e(x_e)$ is at most $\alpha(\mathcal{D})$ times the cost of the corresponding two edges e and e' in $\hat{G}$ by Lemma 5.2.4. □

To finish the proof of the Theorem, we require the following lemmas.

Lemma 5.2.3 *Let G be a network, and x_e be a fractional flow sending one unit of flow from the source s to each sink t_i. Let ℓ denote the gradient of the total cost $xd_e(x)$, that is, let $\ell_e(x) = d_e(x) + xd'_e(x)$ for each edge e. The flow x_e is minimum cost subject to the cost $\sum_e xd_e(x)$ if and only if it is a minimum cost flow subject to the constant cost function $c_e = \ell_e(x_e)$.*

Proof. If the flow x_e is not of minimum cost subject to costs c_e, then the residual graph has a negative cycle, and moving a small amount of flow along the cycle decreases the cost $\sum_e xd_e(x)$,

as the cost c_e is exactly the gradient of this objective function. To see the other direction, we use the fact that the cost function is convex by assumption, and hence all local optima are also global optima. □

Next, it is useful to recall from [72] what is $\alpha(\mathcal{D})$. Consider an edge e, with delay $d(x)$ from class $\mathcal{D}$. Now consider a graph with two parallel links, an edge e, which has delay $d(x)$, that will carry some r units of flow, and a parallel link e' with constant delay $d(r)$ independent of the traffic. Now the unique Nash equilibrium is to route all r units of flow on e, while we get the optimum by setting x such that the gradient $c(x) = d(x) + xd'(x)$ is equal to $d(r)$, and sending x units of flow along e, and the remainder $r - x$ along edge e'. This is because of the following Lemma from [72].

Lemma 5.2.4 (Roughgarden) *If a set $\mathcal{D}$ of delay functions satisfies the above condition, then the price of anarchy (as well as stability) in the above 2-link network is at most $\alpha(D) = \max_{r,x,d\in\mathcal{D}} rd(r)/(xd(x) + (r - x)d(r))$, and the maximum is achieved by setting x such that $d(x) + xd'(x) = d(r)$.*

Chapter 6
Simple Agents in Dynamic Adversarial Load Balancing

In this chapter we look at simple agents interacting in load balancing and packet routing settings. We use the fact that a decentralized algorithm with local knowledge is essentially a set of agents performing local optimizations. The questions about price of stability now become concerns about the algorithm's performance compared to a centralized algorithm with global knowledge. Many of the results in this chapter initially appeared in [8].

6.1 The Model

We consider a basic model of load balancing in a distributed network, which has formed the basis of a number of earlier studies [2, 41, 58, 64, 69]. A network of identical processors is represented by an undirected graph $G = (V, E)$ with $n = |V|$ nodes. There are a number of *jobs* to be processed in the system, abstractly represented by unit-size *tokens*. Time progresses in discrete steps called *rounds*; in a given round, each token is held by one of the nodes, which is viewed as processing the associated job, and the *load* on a node is defined to be the number of tokens it holds. The goal is to *balance* the loads, so that no single node has too many tokens; this can be accomplished by transmitting tokens between neighboring nodes of the graph, at a rate of one token per edge per round. We are particularly interested in *local* algorithms for this problem: rather than using a centralized approach to coordinate the movement of tokens, each node will simply compare its load to those of its neighbors, and decide whether to move a token across an edge based on this information.

Jobs enter and leave this system via an *adversary* that is allowed at the beginning of each round to introduce tokens at some nodes (corresponding to new jobs) and remove tokens from others (corresponding to jobs that have finished). In each round, the adversary first adds or removes tokens (this is called the *Adversary step*). Subsequently, in the *Redistribution step*, up to one token can be moved along each edge $e \in E$ by the algorithm (or up to c_e tokens in the case of networks with edge capacities). The algorithm moves these tokens with the goal of maintaining balanced loads. This alternation of moves by the adversary and algorithm continues for an arbitrary number of rounds. Note that by allowing the adversary to control the removal of tokens as well as their arrival, one is modeling a worst-case assumption that jobs may have arbitrary duration, and the algorithm does not know how much processing time a job has remaining until the moment it ends.

Nodes have queues associated with them, in which they store their tokens. The *height* $h_t(v)$ of a node v is the number of tokens in v's queue at the beginning of round t. Let a_t denote the average number of tokens per node in the system at the beginning of round t. For a set $S \subseteq V$, let $e(S)$ denote the set of edges with exactly one end in S, and $\delta_t(S)$ the net increase in tokens in set S due to the addition and removal of tokens in round t (note that $\delta_t(S)$ could be negative). Then, the adversary is limited by the following *cut condition*:

$$|\delta_t(S) - |S|(a_{t+1} - a_t)| \quad \leq \quad |e(S)|. \tag{6.1}$$

If the heights of nodes in S were to change precisely according to average, then the net change in tokens in S would be $|S| \cdot (a_{t+1} - a_t)$. This condition ensures that the difference between these two quantities is "accounted for" by the edges in $e(S)$.

It is the decision of the algorithm along which edges to send tokens. The goal of a dynamic load balancing algorithm in this model is to keep $h_t(v)$ close to a_t for all nodes v and rounds t. Formally, let the *imbalance* be defined as $b_t(v) = h_t(v) - a_t$, i.e. the number of excess (or missing) tokens at node v with respect to the average over the entire network. $\overline{h}_t(v)$ and $\overline{b}_t(v)$ denote the same quantities after the Adversary step of round t. Then, we say that an algorithm

is *safe* against a given adversary if there is a constant B such that $|b_t(v)| \leq B$ for all nodes v and rounds t. We will next show that a very simple family of local-control algorithms is safe against any adversary respecting the cut condition.

6.2 Single-commodity load balancing

In this section, we will study the load balancing problem for a single commodity, and in particular prove that the natural balancing algorithm is safe for any adversary respecting the cut condition, thus settling an open question from [64]. This algorithm has a *threshold parameter* $\theta \geq 1$, which determines how aggressively the algorithm balances.

```
Algorithm SCLB_θ
At each time t, for each edge e = (u, v):
   If h_t(u) ≥ h_t(v) + θ,
      then send a token from u to v.
   If h_t(v) ≥ h_t(u) + θ,
      then send a token from v to u.
```

This algorithm does not specify whether tokens are sent along an edge (u, v) when $|h_t(u) - h_t(v)| < \theta$. All of our subsequent statements will remain true independently of what the algorithm does in this case. Notice that this algorithm only requires local information, and can therefore be executed in a distributed fashion in a network.

6.2.1 Safety of the algorithm

Our main theorem in this section is that the algorithm $\texttt{SCLB}_\theta$ is safe against an adversary respecting the cut condition of Inequality 6.1. We allow for tokens to be in the system at time 1, and let $H := \max_{v \in V} h_1(v)$.

Theorem 6.2.1 *For any adversary respecting the cut condition, and any $\theta \geq 1$, the algorithm $\texttt{SCLB}_\theta$ is safe, i.e. there is a constant B (depending on H, θ and G), such that $|b_t(v)| \leq B$ for all nodes v at all times t.*

The intuition behind our proof is based on the (incorrect) observation that the algorithm seems to ensure that the height difference between adjacent nodes cannot grow beyond θ. Hence, the largest difference between the heights of any two nodes should be achieved when G is a simple path, and the two nodes are the endpoints of the path — having a height difference of about $n\theta$.

It is, however, easy to see that the height difference between two adjacent nodes can become more than θ, because the adversary can "rearrange" the heights within sets to a certain extent. On the path, for instance, it would be possible to rearrange the tokens on the nodes previously having heights $n\theta$, $(n-1)\theta$ and $(n-2)\theta$ so that they each have $(n-1)\theta$ tokens. The adversary can do this by adding two tokens to the node with height $(n-2)\theta$ and subtracting one from the node with height $n\theta$ for θ successive rounds. This process obeys the cut condition, but afterwards there are two adjacent nodes with almost 2θ height difference. Fortunately, although there are now more nodes with large heights, the adversary had to pay for this rearrangement by making the highest queue smaller. In an amortized sense, the situation has not become worse.

These observations suggest maintaining height bounds for each subset of the nodes, and showing that these bounds form an invariant. For convenience, we will write $b_t(S) = \sum_{v \in S} b_t(v)$ for any set $S \subseteq V$ of vertices (and similarly for other quantities like $h_t(S)$ and $\delta_t(S)$). With Δ denoting the maximum degree of any vertex, we write $\gamma = 2\Delta + \theta$. The key invariant is the following:

$$|b_t(S)| \leq \sum_{j=n-|S|+1}^{n} (H + \gamma \cdot j) \quad \text{for all } S \subseteq V \tag{6.2}$$

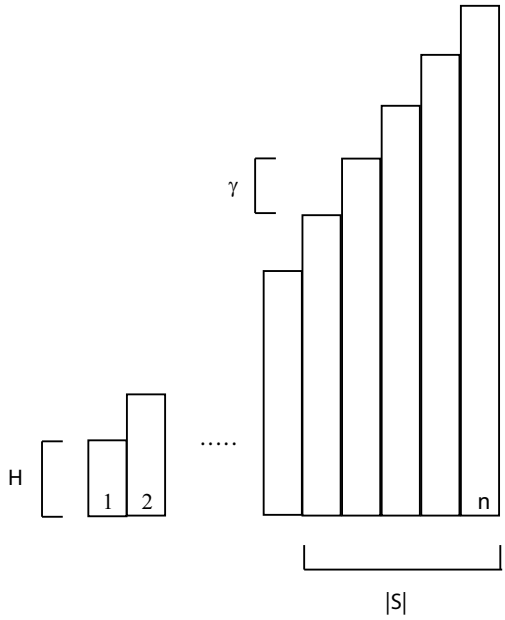

Figure 6.1: An illustration of Invariant 6.2

Figure 6.1 illustrates the upper bound of this invariant pictorially as the sum of the column heights of the right-most $|S|$ columns.

Below, we prove Lemma 6.2.2, showing that Inequality (6.2) is indeed an invariant over time for the algorithm $\texttt{SCLB}_\theta$, against an adversary respecting the cut condition.

Lemma 6.2.2 *If the adversary respects the cut condition, and the invariant (6.2) holds at the beginning of round t, then it holds at the beginning of round $t+1$.*

Using this lemma, the proof of Theorem 6.2.1 is straightforward.

Proof of Theorem 6.2.1. We prove by induction that (6.2) holds at every time t. At time 1, $h_1(v) \leq H$ for all v by definition, so

$$|b_1(S)| \quad \leq \quad \textstyle\sum_{v \in S} H \quad \leq \quad \sum_{j=n-|S|+1}^{n} (H + \gamma \cdot j)$$

for all sets $S \subseteq V$. The induction step from t to $t+1$ follows from Lemma 6.2.2, and we can apply the resulting guarantee to the singleton sets $\{v\}$, yielding a bound of $B = H + \gamma \cdot n$. □

Proof of Lemma 6.2.2. The proof is by contradiction. Assume that the invariant (6.2) holds at the beginning of round t, but not at the beginning of round $t+1$. Let S be a set maximizing

$$\Phi(S) \quad := \quad |b_{t+1}(S)| - \sum_{j=n-|S|+1}^{n} (H + \gamma \cdot j).$$

If several sets achieve the maximum value, let S have minimal size among all these sets. First off, notice that the choice of S guarantees that either all $u \in S$ have positive $b_{t+1}(u)$, or they all have negative $b_{t+1}(u)$, and hence $|b_{t+1}(S)| = \sum_{u \in S} |b_{t+1}(u)|$.

Since (6.2) was assumed to hold at the beginning of round t, and fails at the beginning of round $t+1$, we know that $|b_{t+1}(S)| > |b_t(S)|$. How can the values $h_t(u)$ for nodes $u \in S$ change?

Adversary step: Substituting the definitions of $\overline{b}$ and b, we obtain that for any set S,

$$\begin{aligned} |\overline{b}_t(S)| &= |\overline{h}_t(S) - |S| \cdot a_{t+1}| \\ &= |h_t(S) + \delta_t(S) - |S| \cdot a_{t+1}| \\ &= |b_t(S) + \delta_t(S) - |S| \cdot (a_{t+1} - a_t)| \\ &\leq |b_t(S)| + |\delta_t(S) - |S| \cdot (a_{t+1} - a_t)| \\ &\leq |b_t(S)| + |e(S)|. \end{aligned}$$

The two inequalities hold because of the Triangle Inequality and the cut condition on the adversary.

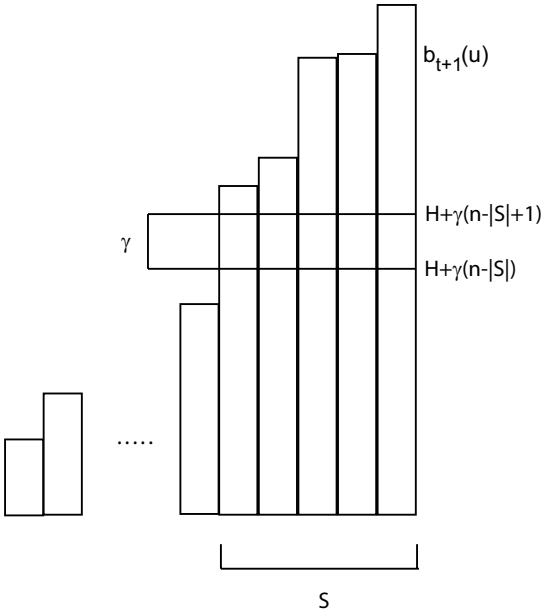

Figure 6.2: The gap of γ between the queue heights of nodes in S and outside S in the case that $b_{t+1}(S)$ is positive.

Redistribution step: Fix an edge $e = (u, v)$ with $u \in S$ and $v \notin S$. Because S maximizes Φ and has minimal size, u has imbalance $|b_{t+1}(u)| > H + \gamma \cdot (n - |S| + 1)$, since otherwise $\Phi(S - u) \geq \Phi(S)$. In particular, $|b_{t+1}(u)| > \gamma$.

Because v was not included in S, its imbalance $b_{t+1}(v)$ must either have sign opposite to the sign of $b_{t+1}(S)$, or have absolute value $|b_{t+1}(v)| \leq H + \gamma \cdot (n - |S|)$. In either case, $|b_{t+1}(u) - b_{t+1}(v)| > \gamma$, and simply substituting the definition of γ, we also obtain $|h_{t+1}(u) - h_{t+1}(v)| > 2\Delta + \theta$. During the Redistribution step of round t, at most Δ tokens can have moved to or from nodes u and v, so their heights can have changed by at most Δ each, and therefore their height difference is still $|\overline{h}_t(u) - \overline{h}_t(v)| > \theta$. Figure 6.2 illustrates the gap of at least γ between the queue heights of nodes in S and outside S.

If $b_{t+1}(u) \geq 0$, then $\overline{h}_t(u) - \overline{h}_t(v) > \theta$, so the algorithm $\texttt{SCLB}_\theta$ moves a token from u to v along e, and no tokens from v to u, thereby decreasing $\overline{b}_t(u)$ by 1. On the other hand, if $b_{t+1}(u) < 0$, then $\overline{h}_t(v) - \overline{h}_t(u) > \theta$, and a token must be moved from v to u along e, increasing the (negative) imbalance $\overline{b}_t(u)$ by 1. Because $|b_{t+1}(u)| > \gamma = \theta + 2\Delta$, and therefore $|\overline{b}_t(u)| > \theta + \Delta$, the sign of the imbalance does not change during the Redistribution step, even if Δ tokens were moved to or from u, and hence, $|\overline{b}_t(u)|$ decreased by 1 as a result of edge e.

This holds for every edge $e \in e(S)$, and using the fact that the average a does not change during the Redistribution step, we obtain that

$$\begin{aligned} |b_{t+1}(S)| &= \textstyle\sum_{u \in S} |b_{t+1}(u)| \\ &\leq (\textstyle\sum_{u \in S} |\overline{b}_t(u)|) - |e(S)| \\ &= |\overline{b}_t(S)| - |e(S)|. \end{aligned}$$

Putting the arguments for the two steps together, we obtain that $|b_{t+1}(S)| \leq |\overline{b}_t(S)| - |e(S)| \leq |b_t(S)|$. This contradicts our assumption that $|b_{t+1}(S)| > |b_t(S)|$, and thus completes the proof. □

Notice that our bound $B = H + \gamma \cdot n$ is asymptotically tight. To see this, consider a simple path of length n. It is certainly legal for the adversary to insert one token at node n in every round, and never remove tokens. After about $\theta \cdot \frac{n^2}{2}$ rounds, each node k will contain about $k\theta$ tokens, and hence the imbalance of node n is about $\theta \cdot \frac{n}{2}$.

6.2.2 Capacities, dynamic networks, and time windows

The result of Theorem 6.2.1 can be easily extended to the case that the edges have (integer) capacities c_e associated with them, and up to c_e tokens can be sent along e in every round. We assume that whenever the algorithm decides to send tokens from u to v along e, it sends as many as possible, i.e. bounded only by the capacity c_e and the number of tokens at u. The cut condition now requires that the imbalance created by the adversary be restricted by the total capacity of the cut.

It is then straightforward to see that both the algorithm and the adversary behave exactly like in the original model when edge e is replaced by c_e parallel edges. (Notice that the constant γ and hence the bound B now depend on the maximum capacity.) If the capacities are not integral, some slight alterations are sufficient to adapt the proof to the new setting.

Another easy extension concerns dynamically changing networks. That is, the set of available edges may change over time, and we assume that it is also controlled by the adversary. For each time t, we have a set E_t of available edges. The cut condition on the adversary must be satisfied at the specific time when the imbalance is created, i.e. $|\delta_t(S) - |S|(a_{t+1} - a_t)| \leq |e_t(S)|$. Here, $e_t(S)$ are the edges from E_t that have exactly one endpoint in S. By syntactically replacing all terms $e(S)$ with $e_t(S)$ in the proof of Lemma 6.2.2, we obtain a proof for the model of dynamically changing networks.

An extension often considered in the contexts of load balancing or packet routing is to relax the restriction on the adversary by allowing it to violate the cut condition for a certain time, provided that it holds "in the long run". Specifically, a *window size* W is specified, and it is required that for any set S and any time window $[t, t + W)$, the imbalance created on set S over that time window is bounded by the total capacity, i.e. $|(\sum_{r=t}^{t+W-1} \delta_r(S)) - |S|(a_{t+W} - a_t)| \leq W \cdot |e(S)|$.

It is not difficult to see that by allowing B to depend on W, we can also extend the safety result to that model — once the imbalance grows too large on a set S, all edges $e \in e(S)$ will be moving tokens so as to reduce the imbalance for every single round of an entire window, so that the imbalance cannot grow further.

6.3 Multi-commodity load balancing

In the previous section, we considered the problem of balancing loads on processors where the loads were interchangeable. However, we are also interested in the case of different kinds of loads that are to be balanced simultaneously. For instance, think of jobs that have an emphasis on different resources of the machine they are running on. Balancing the different classes of jobs independently could be desirable in order to avoid processing time becoming a bottleneck on one machine, and memory size an issue on another.

In the general multi-commodity load balancing problem, we have k different kinds of jobs (or tokens), which are stored in separate queues at the nodes. Our goal is to ensure an absolute bound on the deviation of any queue height from the average queue height for that commodity. Each round t is divided into the same two steps as before, the Adversary step and the Redistribution step.

In analogy to the single-commodity case, we use the following notation: For a node v and commodity i, let $h_t^{(i)}(v)$ be the number of tokens of commodity i on node v at the beginning of round t. Similarly, $a_t^{(i)}$, $b_t^{(i)}(v)$, $\delta_t^{(i)}(v)$, $\overline{h}_t^{(i)}(v)$, and $\overline{b}_t^{(i)}(v)$ are all defined for commodity i exactly as their single-commodity equivalents.

The algorithm will now not only have to choose when to send a token across an edge, but also which of several available (and conflicting) kinds of tokens to send. Our class of algorithms is practically identical to the one from [3] and [10], and can be formalized as follows:

```
Algorithm MCLB_θ
At each time t, for each edge e = (u, v):
   Choose i to maximize |h_t^(i)(u) − h_t^(i)(v)|.
   If h_t^(i)(u) ≥ h_t^(i)(v) + θ, then send a token
      of commodity i from u to v.
   If h_t^(i)(v) ≥ h_t^(i)(u) + θ, then send a token
      of commodity i from v to u.
```

In the case of a single commodity, this algorithm specializes to $\texttt{SCLB}_\theta$.

The natural analogue of the cut condition for a single commodity is to require that the adversary satisfy

$$\sum_i |\delta_t^{(i)}(S) - |S|(a_{t+1}^{(i)} - a_t^{(i)})| \quad \leq \quad |e(S)| \tag{6.3}$$

for all node sets $S \subseteq V$ and times t. This would require that the total imbalance for set S created by the adversary could be "balanced" along edges leaving S.

Unfortunately, Inequality (6.3) is too weak a restriction — it allows the adversary to create patterns of addition and removal that cannot be balanced by any algorithm, whether offline or online. At the end of this section, we show how to use a reduction from the multi-commodity flow problem to create such an adversary with $k \geq 3$ commodities.

For the special case $k = 2$, however, the Max-Flow Min-Cut Theorem still holds, and in fact, we can show that the cut condition is sufficient to ensure that algorithm $\texttt{MCLB}_\theta$ is safe.

6.3.1 Safety for k=2

We let $H = \max_{v \in V}\{h_1^{(1)}(v) + h_1^{(2)}(v)\}$ be the maximum height of the queues at any node during the start of the execution, and Δ the maximum degree of any vertex. This time, we define γ' slightly differently, namely $\gamma' = 2\Delta + 2\theta$.

Theorem 6.3.1 *For any adversary respecting the cut condition, there is a constant B (depending on H, θ and G), such that $\texttt{MCLB}_\theta$ ensures $|b_t^{(i)}(v)| \leq B$ at all times t, for all vertices v, and commodities $i = 1, 2$.*

Proof. At the start of the execution, $|b_1^{(1)}(S)| + |b_1^{(2)}(S)| \leq \sum_{v \in S} H$, by definition of H. The key Lemma 6.3.2 establishes that for all $S \subseteq V$, times t, and commodities $i = 1, 2$,

$$|b_t^{(1)}(S)| + |b_t^{(2)}(S)| \quad \leq \quad \sum_{j=n-|S|+1}^{n} (H + \gamma' \cdot j) \tag{6.4}$$

We can then apply the result to all singleton sets $\{v\}$, proving the theorem. □

Lemma 6.3.2 *If (6.4) holds at the beginning of round t, it holds at the beginning of round $t+1$.*

Proof. The proof is by contradiction. Let S be a set (of minimum size in case of ties) maximizing

$$\Phi(S) \quad := \quad |b_t^{(1)}(S)| + |b_t^{(2)}(S)| - \sum_{j=n-|S|+1}^{n} (H + \gamma' \cdot j).$$

In the case of two commodities, a node might be included in S because it contributes a lot to the imbalance in one of the commodities, although its contribution to the other commodity might actually be negative. To capture the imbalance contribution of a node to each commodity, we define the *signed imbalance* $\beta_t^{(i)}(v) := \text{sgn}(b_{t+1}^{(i)}(S)) \cdot b_t^{(i)}(v)$, $\beta_{t+1}^{(i)}(v) := \text{sgn}(b_{t+1}^{(i)}(S)) \cdot b_{t+1}^{(i)}(v)$, $\overline{\beta}_t^{(i)}(v) := \text{sgn}(b_{t+1}^{(i)}(S)) \cdot \overline{b}_t^{(i)}(v)$ for every node $v \in V$. Here, sgn denotes the sign of a term. Notice that we always use the sign of $b_{t+1}^{(i)}(S)$ at time $t+1$, even when defining the signed imbalance at time t. Then, we can rewrite the total imbalance over the set S at time $t+1$ as

$$|b_{t+1}^{(1)}(S)| + |b_{t+1}^{(2)}(S)| \;=\; \sum_{u \in S} (\beta_{t+1}^{(1)}(u) + \beta_{t+1}^{(2)}(u)). \tag{6.5}$$

Again, we show that the change in the imbalance for set S cannot be positive, by comparing the increase in the imbalance of S during the Adversary step with the decrease during the Redistribution step, and thus obtain a contradiction.

Adversary step: We know that for each commodity $i = 1, 2$, the imbalance on set S after the Adversary step is at most $|\overline{b}_t^{(i)}(S)| \leq |b_t^{(i)}(S)| + |\delta_t^{(i)}(S) - |S| \cdot (a_{t+1}^{(i)} - a_t^{(i)})|$ by the Triangle Inequality. Now, summing over $i = 1, 2$ and applying the cut condition on the adversary yields that

$$|\overline{b}_t^{(1)}(S)| + |\overline{b}_t^{(2)}(S)| \quad \leq \quad |b_t^{(1)}(S)| + |b_t^{(2)}(S)| + |e(S)|.$$

Redistribution step: Fix a node $u \in S$, and an edge $e = (u, v)$ of G with $v \notin S$. As in the proof of Lemma 6.2.2, we can use the definition of S (as maximizing Φ and being of minimal size) to obtain that the signed imbalances at nodes u and v satisfy $\beta_{t+1}^{(1)}(u) + \beta_{t+1}^{(2)}(u) > \beta_{t+1}^{(1)}(v) + \beta_{t+1}^{(2)}(v) + \gamma'$. As u and v can lose and gain at most Δ tokens each during the Redistribution step,

$$\overline{\beta}_t^{(1)}(u) + \overline{\beta}_t^{(2)}(u) \quad > \quad \overline{\beta}_t^{(1)}(v) + \overline{\beta}_t^{(2)}(v) + 2\theta. \tag{6.6}$$

In particular, there must be a commodity i such that $\beta_t^{(i)}(u) > \beta_t^{(i)}(v) + \theta$, and thus also $|\overline{h}_t^{(i)}(u) - \overline{h}_t^{(i)}(v)| > \theta$. Hence, $\texttt{MCLB}_\theta$ moved a token along edge e during the Redistribution step (w.l.o.g., it was a token of commodity 1). We want to show that this token actually decreased the signed imbalance of node u. Assume that it did not. This means that if the token moved from u to v, then $\text{sgn}(b_{t+1}^{(1)}(S))$ is negative, and if the token moved from v to u, then $\text{sgn}(b_{t+1}^{(1)}(S))$ is positive. In either case, the signed imbalance for commodity 1 at node v must be higher than at node u, and so

$$\overline{\beta}_t^{(1)}(v) - \overline{\beta}_t^{(1)}(u) = |\overline{b}_t^{(1)}(v) - \overline{b}_t^{(1)}(u)| = |\overline{h}_t^{(1)}(v) - \overline{h}_t^{(1)}(u)|.$$

Because $\texttt{MCLB}_\theta$ maximizes the difference in its choice of commodity, we obtain that

$$\begin{aligned} \overline{\beta}_t^{(1)}(v) - \overline{\beta}_t^{(1)}(u) &= |\overline{h}_t^{(1)}(v) - \overline{h}_t^{(1)}(u)| \\ &\geq |\overline{h}_t^{(2)}(v) - \overline{h}_t^{(2)}(u)| \\ &= |\overline{\beta}_t^{(2)}(v) - \overline{\beta}_t^{(2)}(u)| \\ &\geq \overline{\beta}_t^{(2)}(u) - \overline{\beta}_t^{(2)}(v). \end{aligned}$$

Rearranging this inequality yields that

$$\overline{\beta}_t^{(1)}(v) + \overline{\beta}_t^{(2)}(v) \geq \overline{\beta}_t^{(1)}(u) + \overline{\beta}_t^{(2)}(u),$$

and thus a contradiction with Inequality (6.6). Therefore, every edge (u, v) with $v \notin S$ decreases the signed imbalance of u by 1. Summing over all edges and all nodes $u \in S$, gives us $\beta_{t+1}^{(1)}(S) + \beta_{t+1}^{(2)}(S) \leq \overline{\beta}_t^{(1)}(S) + \overline{\beta}_t^{(2)}(S) - |e(S)|$. Using Equation (6.5) and the fact that $\overline{\beta}_t^{(1)}(S) + \overline{\beta}_t^{(2)}(S) \leq |\overline{b}_t^{(1)}(S)| + |\overline{b}_t^{(2)}(S)|$, we obtain $|b_{t+1}^{(1)}(S)| + |b_{t+1}^{(2)}(S)| \leq |\overline{b}_t^{(1)}(S)| + |\overline{b}_t^{(2)}(S)| - |e(S)|$.

Combining the arguments for the two steps, $|b_t^{(1)}(S)| + |b_t^{(2)}(S)|$ increases by at most $|e(S)|$ during the Adversary step, and decreases by at least $|e(S)|$ during the Redistribution step. Therefore, in total $|b_{t+1}^{(1)}(S)| + |b_{t+1}^{(2)}(S)| \leq |b_t^{(1)}(S)| + |b_t^{(2)}(S)|$, a contradiction. □

The result for two commodities can be extended to networks with edge capacities, adversarially changing edge sets, and adversaries with restrictions only for larger window sizes, just like for the single commodity case.

6.3.2 Load balancing and flows

By omitting the adversarial and dynamic nature in the load balancing problem, and forcing the adversary to repeat the same pattern of token additions in every round, we can infer from the safety of the load balancing algorithm the existence of a multi-commodity flow. Suppose that we are given a multi-commodity flow instance with edge capacities c_e, source-sink pairs (s_i, t_i), and demands d_i. Let $D = \max_i d_i$, and let $\mathcal{A}$ be the adversary inserting, at every round for each commodity i, $D + d_i$ tokens at node s_i, $D - d_i$ tokens at node t_i, and D tokens everywhere else.

Lemma 6.3.3 *If any load-balancing algorithm is safe against the adversary $\mathcal{A}$, then there is a (fractional) multi-commodity flow f with source-sink pairs (s_i, t_i) and demands d_i.*

Proof. Because we assumed the algorithm to be safe, all imbalances $b_t^{(i)}(v)$ are always bounded in absolute value by some constant B. Therefore, there are at most $(2B+1)^{kn}$ different combinations of imbalances for the entire network, and there must be times $t < t'$ such that $b_t^{(i)}(v) = b_{t'}^{(i)}(v)$ for all nodes v and commodities i.

For each edge $e = (u, v)$, let $\sigma_t^{(i)}(u, v)$ denote the number of tokens of commodity i sent from u to v in round t, and define a flow f by

$$f_{(u,v)}^{(i)} \quad := \quad \frac{1}{t' - t} \cdot \sum_{r=t}^{t'-1} (\sigma_r^{(i)}(u, v) - \sigma_r^{(i)}(v, u)).$$

Notice that we define negative flows, but only for symmetry and ease of notation. We want to verify that f is indeed a feasible multi-commodity flow for demands (s_i, t_i, d_i).

Capacity constraints: The total flow along any edge (u, v) is

$$\begin{aligned}
\sum_i |f_{(u,v)}^{(i)}| &\leq \frac{1}{t' - t} \cdot \sum_i \sum_{r=t}^{t'-1} |\sigma_r^{(i)}(u, v) - \sigma_r^{(i)}(v, u)| \\
&= \frac{1}{t' - t} \cdot \sum_{r=t}^{t'-1} \sum_i |\sigma_r^{(i)}(u, v) - \sigma_r^{(i)}(v, u)| \\
&\leq \frac{1}{t' - t} \cdot \sum_{r=t}^{t'-1} c_{(u,v)} \\
&= c_{(u,v)}.
\end{aligned}$$

The first inequality is simply the Triangle inequality, and the second inequality holds because the balancing algorithm never exceeds the capacity of any edge with any of its token moves, and therefore both $\sigma_r^{(i)}(u,v)$ and $\sigma_r^{(i)}(v,u)$ lie between 0 and $c_{(u,v)}$.

Flow Conservation: For any node v and commodity i, we can write

$$\begin{aligned}
&(t'-t)\sum_{(u,v)\in E} f^{(i)}_{(u,v)} \\
&= \sum_{(u,v)\in E}\sum_{r=t}^{t'-1}(\sigma_r^{(i)}(u,v)-\sigma_r^{(i)}(v,u)) \\
&= \sum_{r=t}^{t'-1}\sum_{(u,v)\in E}\sigma_r^{(i)}(u,v)-\sum_{r=t}^{t'-1}\sum_{(u,v)\in E}\sigma_r^{(i)}(v,u) \\
&= h_{t'}^{(i)}(v)-h_t^{(i)}(v)-\sum_{r=t}^{t'-1}\delta_r^{(i)}(v) \\
&= b_{t'}^{(i)}(v)+a_{t'}^{(i)}-b_t^{(i)}(v)-a_t^{(i)}-(t'-t)\cdot\delta_t^{(i)}(v) \\
&= (t'-t)(D-\delta_t^{(i)}(v)).
\end{aligned}$$

The third equality is because $h_{t'}^{(i)}(v)-h_t^{(i)}(v)$ is exactly the amount of tokens that entered v during the time period $[t,t'-1]$, minus the amount of tokens that left v during this period. In the last equality, we used that $b_{t'}^{(i)}(v)=b_t^{(i)}(v)$ for all i and v, and that $a_r^{(i)}=r\cdot D$ for all times r. Now, if node v is neither the source nor the sink for commodity i, then $\delta_t^{(i)}(v)=D$, so flow is conserved. If v is the source of commodity i, then $\delta_t^{(i)}(v)=D+d_i$, so the total flow entering node s_i is $-d_i$. If v is the sink for commodity i, then the total flow entering node t_i is d_i, because $\delta_t^{(i)}(v)=D-d_i$ by definition. Hence, f satisfies flow conservation and all demands.

f conserves flow, satisfies all demands, and does not exceed any edge capacities, so it is a feasible multi-commodity flow for the given demands. □

By combining Lemma 6.3.3 with the safety of $\texttt{MCLB}_\theta$ proved in Theorem 6.3.1, we obtain as a corollary an alternate proof of the two-commodity Max-Flow Min-Cut Theorem. In only remains to verify that the adversary $\mathcal{A}$ as defined in Lemma 6.3.3 indeed respects the cut-condition.

Corollary 6.3.4 (2-Commodity Max-Flow Min-Cut) *Let $G=(V,E)$ be a graph with edge capacities c_e, and two demand pairs (s_1,t_1), (s_2,t_2) with demands d_1, d_2 such that for any vertex set $S\subseteq V$, the total demand of commodities $i\in\{1,2\}$ with exactly one of $\{s_i,t_i\}$ in S is at most $\sum_{e\in e(S)}c_e$. Then, there exists a feasible two-commodity flow sending d_i units of flow from s_i to t_i for $i=1,2$.*

Proof. Define the adversary $\mathcal{A}$ as in Lemma 6.3.3. To show that the algorithm $\texttt{MCLB}_\theta$ is safe against $\mathcal{A}$, we merely have to verify that $\mathcal{A}$ satisfies the cut condition. Let $S\subseteq V$ be arbitrary. For convenience, we write $[u\in S]:=1$ if $u\in S$, and 0 otherwise. Then, for any time t

$$\begin{aligned}
&\sum_{i=1,2}|\delta_t^{(i)}(S)-|S|\cdot(a_{t+1}^{(i)}-a_t^{(i)})| \\
&= \sum_{i=1,2}||S|\cdot D+[s_i\in S]\cdot d_i-[t_i\in S]\cdot d_i-|S|\cdot D| \\
&= \sum_{i=1,2}d_i\cdot|[s_i\in S]-[t_i\in S]|.
\end{aligned}$$

In the first equality, we used the definition of the insertion pattern for $\mathcal{A}$. The contribution of commodity i to this sum is d_i if and only if exactly one of s_i,t_i lies in S — otherwise, it is 0.

Hence, the value of the sum is the total demand of commodities i with exactly one of $\{s_i, t_i\}$ in S, which by assumption is bounded by $\sum_{e \in e(S)} c_e$. Hence, $\mathcal{A}$ satisfies the cut condition.

We can therefore apply Theorem 6.3.1 to obtain that $\texttt{MCLB}_\theta$ is safe against $\mathcal{A}$, which in turn implies the existence of a feasible multi-commodity flow f for the given instance via Lemma 6.3.3. □

The 2-Commodity Max-Flow Min-Cut Theorem was first proved by Hu [45], essentially repeating Ford and Fulkerson's original [35] augmenting paths argument for two commodities. Seymour [77] showed a short and simple explicit reduction to the single-commodity case. Subsequently, Linial, London and Rabinovich [56] gave a novel proof using geometric embeddings and linear programming duality. Our proof uses yet different (and much more elementary) techniques, and does not rely on the single-commodity Max-Flow Min-Cut Theorem.

Another Corollary we obtain from Lemma 6.3.3 is the existence of an adversary respecting the cut condition for $k = 3$ commodities, such that no algorithm (offline or online) can balance the insertion pattern. To prove this, we simply take a 3-commodity instance with a graph $G = (V, E)$ and demand pairs $(s_i, t_i), i \in \{1, 2, 3\}$ (with demands d_i) such that for all cuts $(S, V \setminus S)$, the total demand across the cut is at most the capacity of the edges crossing the cut, yet there is no (fractional) multi-commodity flow satisfying all demands. The first such example for $k = 3$ was given in [45]. (It is well-known that with appropriate demand pairs, the complete bipartite graph $K_{2,3}$ can be used to obtain an example for $k = 4$ with unit demands.) Let $\mathcal{A}$ be the adversary defined from this instance as in Lemma 6.3.3. If any load balancing algorithm were safe against $\mathcal{A}$, Lemma 6.3.3 would guarantee a feasible multi-commodity flow, a contradiction. In Section 7.1.3 we will discuss an alternative approach for a restriction on a multi-commodity adversary.

6.4 Packet Routing

There is a natural connection between the load balancing problem studied in the previous sections, and the problem of routing packets in an adversarial network. It has been observed previously [3, 10] that the natural balancing algorithm $\texttt{SCLB}_\theta$ is also safe for packet routing.

The model for packet routing differs from the load balancing one in that after the Redistribution step, there is an additional *Removal step*, during which all packets that have reached their destination are removed from the network. Safety of an algorithm is now defined as meaning that there is an absolute bound on all queue heights at all times, i.e. $h_t^{(i)}(v) \leq B$ for some constant B.

In the single-commodity packet routing problem, we can again restrict the adversary by a cut condition: the total number $\delta_t(S)$ of packets inserted into a set S must be at most $|e(S)|$ for any set S not containing the sink of the packets. If S does contain the sink, then there is no restriction. In the multi-commodity case, the adversary specifies a source s_i and a sink t_i for each packet inserted, and must guarantee that there is a set of edge-disjoint paths connecting all s_i-t_i pairs.

In [10], it was shown that for the single-commodity case, the algorithm $\texttt{SCLB}_\theta$ is safe against an adversary guaranteeing the existence of a path for all packets inserted, even when edges dynamically appear and disappear. Aiello et. al. [3] proved that an algorithm essentially equivalent to $\texttt{MCLB}_\theta$ is safe for the multi-commodity packet routing problem if the paths specified by the adversary are not only disjoint, but leave an ε fraction of capacity for every edge unused over a given window length W.

Our techniques from Section 6.2 can be used to obtain an alternate (and simpler) proof for the safety of algorithm $\texttt{SCLB}_\theta$ in the packet routing model. Our proof also works for the case of adversarially appearing and disappearing edges, although the restriction on the adversary is different from (and essentially less general than) the one in [10].

We define Δ and H as before, and let $\gamma = 2\Delta + \theta$. Then, the safety of $\texttt{SCLB}_\theta$ against an adversary respecting the cut condition is guaranteed by the following theorem.

Theorem 6.4.1 *For any time t and set $S \subseteq V$,*

$$h_t(S) \quad \leq \quad \sum_{j=n-|S|+1}^{n} (H + \gamma \cdot j). \tag{6.7}$$

Proof. By definition of H, Invariant (6.7) certainly holds at time 1. Assume that the theorem is wrong, and let t be the earliest time such that there is a set S violating (6.7) at time $t+1$. Among all such sets, let S be the one maximizing $h_{t+1}(S) - \sum_{j=n-|S|+1}^{n}(H + \gamma \cdot j)$, and break ties for minimal size. Then, we can show as in the proof of Lemma 6.2.2 that for all nodes $u \in S, v \notin S$, $h_{t+1}(u) > h_{t+1}(v) + \gamma$, and $h_{t+1}(u) > H$. In particular, the set S cannot contain the sink (because the sink contains no tokens after the Removal step).

Therefore, the adversary can have inserted at most $|e(S)|$ tokens into S in the Adversary step. During the Redistribution step of round t, a token leaves the set S along each edge $e = (u, v) \in e(S)$, because even if u had lost Δ tokens and v gained Δ tokens during the Redistribution step, $\overline{h}_t(u)$ would still exceed $\overline{h}_t(v)$ by at least θ. Taken together, this shows that the number of tokens in S cannot have increased, contradicting the choice of t and S. □

Extensions: Directed Graphs and More

Like the proofs in the previous sections, the proof of Theorem 6.4.1 can be easily extended to deal with dynamically changing networks, edge capacities, and time windows in the cut condition. In addition, this proof extends to directed graphs G, if we redefine $e(S)$ in the cut condition to be the edges coming out of S. We can also show safety of a wider class of single-commodity balancing algorithms. Specifically, let $g : \mathbb{N} \to \mathbb{N}$ be a function with $g(x) \geq x$ for all x. The balancing algorithm $\texttt{SCLB}_g$ always sends a token from u to v (and never sends a token from v to u) if there is an edge $e = (u, v) \in E$ and $h_t(u) > g(h_t(v))$. It does not matter what the algorithm does if neither $h_t(u) > g(h_t(v))$ nor $h_t(v) > g(h_t(u))$. Extending the above proof only slightly, we obtain that $\texttt{SCLB}_g$ is safe for all such functions g. Of course, the bound B now depends on the rate of growth of g.

Our packet routing results also imply the safety of an interesting load balancing scenario. Suppose each node processes one job per round, so that one token is removed every round from each node with positive queue height. If the adversary $\mathcal{A}$ satisfies the constraint that the number of tokens it adds to S is at most $e(S) + |S|$, then we can use Theorem 6.4.1 to show that the queue heights are bounded. Consider adding a sink node v to our graph, and add an edge from every node to v. We can now think of $\mathcal{A}$ as adding packets to the new graph, with v being the packets' destination. The number of edges coming out of a set S in the new graph is exactly $e'(S) = e(S) + |S|$, so $\mathcal{A}$ satisfies the condition needed for Theorem 6.4.1. It is easy to see that if the queue heights are bounded in this new packet routing scenario, then they are also bounded in the original load balancing one.

The proofs for Theorem 6.4.1 and Theorem 6.2.1 (and the proofs in [3] and [64]) are so similar in nature that one suspects a formal reduction from the packet routing problem to the load balancing problem (which seems more general). However, we have not yet been able to determine such a reduction. It would certainly be interesting, since it would allow us to focus on the load balancing problem in the future.

As with the load balancing problem, we can obtain a multi-commodity flow if $\texttt{MCLB}_\theta$ is safe against a suitably defined adversary $\mathcal{A}$. The proof is practically identical to the one for Lemma 6.3.3, and we therefore omit it.

Lemma 6.4.2 *Let $\mathcal{A}$ be an adversary inserting d_i tokens of commodity i (whose destination is t_i) into node s_i in every round. If any routing algorithm is safe against this adversary, then there is a (fractional) multi-commodity flow f with source-sink pairs (s_i, t_i) and demands d_i.*

Chapter 7
Conclusion and Further Directions

In this thesis we focused on the price of stability in networks designed by self-interested agents. We considered several versions of network creation games, and showed various bounds on the price of stability, as well as addressed the existence of Nash equilibria and the convergence of best-response dynamics. We also examined network routing with atomic demands, and dynamic distributed processes in load balancing and packet routing. Our results suggest many concrete open problems in these areas, as well as important fundamental directions for further research. In this chapter we describe the most important and promising of these.

7.1 Open Questions

7.1.1 Bicriteria Approximations

Most of our important results in Chapter 3 can be phrased as bicriteria approximations. We show different cases where there exists (or it is possible to find) a β-approximate Nash equilibrium that is only a factor of α more expensive than the centralized optimum. Specifically, for the single-source case of the Connection Game with Arbitrary Cost Sharing we showed that there exists an exact Nash equilibrium which costs no more than the centralized optimum, which is a $(1, 1)$ approximation. The results of this kind from Chapter 3 are summarized in Table 7.1. Notice that the numbers 1.55 and 4.65 rely on the best approximation factor to the Steiner tree problem, and will become lower as better approximations to that problem are found. The lower bound of $(1.5, 1)$ from Table 7.1 means that there cannot be a β-approximate Nash equilibrium that costs as much as the centrally optimal solution, with $\beta < 1.5$.

	Single-Source	Multi-Source
Exists Nash	(1,1)	(3,1)
Can find Nash in poly-time	$(1 + \varepsilon, 1.55)$	$(4.65 + \varepsilon, 2)$
Lower Bounds on Existence	(1,1)	(1.5,1)

Table 7.1: Bicriteria approximations, written as (β, α), meaning there exists (or it is possible to find) a β-approximate Nash equilibrium that is only a factor of α more expensive than the centralized optimum.

Looking at these results as a two-parameter optimization problem, we see that we have just begun exploring the space of possible approximations. In the case of the single-source problem, we are mostly done, since we cannot get a better result than a $(1, 1)$ approximation, and it is doubtful that we can do much better than $(1, 1.55)$ in polynomial time unless a better approximation for the Steiner tree problem is found. For the general multi-source problem, however, things are very different. While we have shown that we cannot hope to get a β-approximate Nash equilibrium with small β that is as good as the optimum, we do not have any results for $\alpha > 1$. In other words, it is still possible that there exists a $(1 + \varepsilon)$-approximate Nash equilibrium for the general case that is close to optimum in cost. One of the most important concrete open questions here is:

Open Question 7.1.1 *Does there always exist a β-approximate Nash equilibrium that costs at most α times the centralized optimum in the Connection Game with Arbitrary Cost Sharing, for α and β both being small?*

Our results even leave open the possibility of a $(1+\varepsilon, 1+\varepsilon)$ approximation in this context. Proving these sorts of results for $\alpha > 1$ may be far more difficult and require novel proof techniques, since we would not be able to use the fact that we are dealing with the centrally optimal solution, which

has certain nice properties. While it seems doubtful that a $(1+\varepsilon, 1+\varepsilon)$ approximation always holds, we believe it rather likely that an approximation like $(1+\varepsilon, 2)$ may hold. This would still be of interest, since it would correspond to a good network on which the incentive to deviate for any player is at most $1+\varepsilon$ times what they are paying now. If we were a central authority that wanted to build a good network, we would only need to pay tiny incentives to all the players to guarantee the existence of a cheap network built by the players themselves. Furthermore, this result would have the advantage that it may be possible to find this (approximate) Nash equilibrium in polynomial time, since it is 2 times more expensive than the optimal centralized solution, and 2 is the best known approximation factor for the generalized Steiner forest problem.

For the general Connection Game with Arbitrary Sharing, there are still many things we would like to say about the existence of approximate Nash equilibria on the optimal centralized solution. We currently have an upper bound of 3 and a lower bound of 1.5. While 3-approximate Nash equilibria do not pose too much interest, 1.5-approximate ones do, since there are many settings where players may not want to deviate if they increase their utility by a factor of 3/2, but it is hard to imagine such settings if the factor was instead 3. Fortunately, we believe that the actual approximation factor is closer to 1.5, and would like to narrow the gap:

Open Question 7.1.2 *Does a 1.5-approximate Nash equilibrium that is as cheap as the centralized optimum always exist in the Connection Game with Arbitrary Cost Sharing? What about a 2-approximate Nash equilibrium?*

As seen in Table 7.1, besides existence results we have also provided polynomial-time algorithms for finding good Nash equilibria. Unfortunately, it is an artifact of our proofs that an extra factor of ε is added to the approximation factor of the Nash equilibrium when we switch from existence to actually finding it. We do not feel that this factor has to be there, and would like to see it removed:

Open Question 7.1.3 *Can we find an* exact *Nash equilibrium in the single source case which is at most twice as expensive as the centralized optimum? In other words, can we find a* $(1, 2)$ *approximation in polynomial time?*

We have shown that our bicriteria results are quite versatile and admit a number of extensions. Unfortunately, there are a few notable extensions that we were not able to show. Whether the above results hold for those extensions remain open questions:

Open Question 7.1.4 *In the single-source version of the Connection Game with Arbitrary Cost Sharing, does there always exist an exact Nash equilibrium as cheap as the centralized optimum* if the players are trying to connect more than 2 terminals each? *In other words, is the price of stability still 1 in the case where players may be trying to connect many terminals, but share one in common?*

Open Question 7.1.5 *In the general (multi-source) version of the Connection Game with Arbitrary Sharing, does there exist a* β*-approximate Nash equilibrium that is as cheap as the centralized optimum* if the graph is directed*? (The proof for* $\beta = 3$ *only works for the undirected case so far.)*

7.1.2 Fair Connection Game and Extensions

For the Fair Connection Game, we showed that the price of stability is always at most $\log N$, and that this is tight for directed graphs. However, the case for undirected graphs remains largely unresolved. The results of Section 4.3 show that, at least for two players, the price of stability for undirected graphs can be strictly better than that for directed. 8/5 is the greatest lower bound we currently possess for the price of stability of undirected graphs. This logically leads to the following important open question.

Open Question 7.1.6 *What is the price of stability in the Fair Connection Game with undirected edges? In particular, is it a constant?*

If the price of stability in this case was 2 (which we consider quite possible), this would be a wonderful result. We have shown in Chapter 4 that the introduction of the Shapley-value cost sharing scheme decreases the price of stability from as much as $\Theta(N)$ to $\Theta(\log N)$ if the graph has directed edges. If the above question was answered positively, however, it would tell us that this same scheme makes an even greater improvement in the quality of Nash equilibria for the undirected Connection Game, since in the Connection Game with Arbitrary Cost Sharing the price of stability for undirected graphs can be as large as $\Omega(N)$.

We showed that the price of stability in the Fair Connection Game is $\log N$, but we have not actually shown how to find a good Nash equilibrium in polynomial time. Perhaps the most useful open question here is:

Open Question 7.1.7 *Can we find a cheap (possibly approximate) Nash equilibrium for the Fair Connection Game in polynomial time?*

We already attempted to answer this question by trying to analyze the convergence rates of best response dynamics, which we know converges to a Nash equilibrium in the Fair Connection Game. However, we have found that sometimes, if the choice of player moves is particularly bad, best response dynamics could take exponential time to converge to a Nash equilibrium. However, we have not resolved the following question:

Open Question 7.1.8 *For every instance of the Fair Connection Game, does there exist a polynomial-length sequence of best-response moves that ends up at a Nash equilibrium?*

If this were true, and we were able to find this sequence, then we would be able to find a cheap Nash equilibrium as well, since we have shown that any best response sequence starting with the centralized optimum ends up at a Nash equilibrium that is at most $\log N$ more expensive. While finding the centralized optimum itself is NP-Hard, we can find a constant approximation to it, perform best response starting from this configuration, and end up with a Nash equilibrium that is only $O(\log N)$ more expensive than the best centralized solution.

It may be tempting to form a polynomial-time algorithm finding a $(1+\varepsilon)$-approximate Nash equilibrium by using similar methods as we used in Chapter 3. For the case of the Connection Game with Arbitrary Cost Sharing, however, we had an algorithm that either yielded an approximate Nash equilibrium on the given set of edges, or a set of edges that is cheaper by a constant factor. We have no such algorithm for the case of the Fair Connection Game. It may seem at first that the following algorithm would be enough: "Look if any best-response move decreases the potential Φ by more than an ε factor. If it does, then take it." Unfortunately, this would give us a solution where no player could deviate and increase its utility by more than an ε factor *of the entire solution cost*. Since the cost to a particular player can be much smaller than the cost of the entire solution, this is not an ε-approximate Nash equilibrium. We can successfully use this idea to form a pseudo-polynomial time algorithm, however, whose running time depends on the maximum edge cost.

The most natural extension of the Fair Connection Game is the same game with weighted players addressed in Section 4.5. Indeed, why should all the players share the cost of an edge equally, instead of proportionally to the amount of traffic they are sending through? Unfortunately, everything becomes much harder with weighted players. We have produced some preliminary results, but have hardly scratched the surface. We have shown the existence of Nash equilibria only in special cases. Recently, Chen and Roughgarden [18] proved that there exist weighted games with fixed edge costs and no pure Nash equilibria, settling an open question from [6]. They also showed that any weighted game has a $O(\log w_{\max})$-approximate Nash equilibrium that costs at most $O(\log W)$ times the centralized optimum (where $w_{\max}$ is the maximum weight). This still leaves the natural open question:

Open Question 7.1.9 *What is the price of stability of the Weighted Fair Connection Game if all players share a terminal in common? Can the results of [18] be improved for this case?*

As for the version of the Fair Connection Game with latencies instead of edge costs, the main challenge is to provide price of stability results for the general scenario, not just for single-source.

7.1.3 Open Questions for Simple Agents in Load Balancing

In Chapter 6, we have shown that a simple local load-balancing algorithm is safe against dynamic adversarial addition and removal of jobs in a network, so long as the adversary is bounded by a natural extension of the cut condition in the sense defined in [64]. This settles an open question from [64]. Our proof techniques extend to the case of balancing two commodities at once, and to routing packets injected by an adversary. They yield easier proofs and essentially tight bounds for the general case. In addition, the safety of the load balancing algorithm for two commodities gives a new proof of the two-commodity Max-Flow Min-Cut Theorem.

This work leaves open a number of interesting questions. Most importantly, we would like to be able to show safety of the multi-commodity load balancing algorithm for an arbitrary number of commodities, both for the problem of routing packets and balancing loads. If we want to prove safety of load-balancing algorithms for more commodities, we will have to use a different condition on the adversary. As a consequence of Lemma 6.3.3, any reasonable restriction on the adversary will have to guarantee the existence of multi-commodity flows for all instances where we hope to prove safety. We therefore suggest the following restriction:

> We define demands $d_t^{(i)}(v)$ for commodity i, node v and time t by $d_t^{(i)}(v) := \delta_t^{(i)}(v) - (a_{t+1}^{(i)} - a_t^{(i)})$. Then, the adversary is restricted to moves that guarantee the existence of a (fractional) multi-commodity flow in G satisfying all these demands.

The disadvantage of this condition is that it bears no direct relation to the load balancing problem — it arises from observing the insufficiency of the more natural cut condition rather than from having an actual meaning for the problem of balancing loads. Nevertheless, this condition should be considered the right restriction on the adversary to measure the quality of $\texttt{MCLB}_\theta$ or other load balancing algorithms, which leads to the question:

Open Question 7.1.10 *Is* $\texttt{MCLB}_\theta$ *safe against adversaries obeying the above condition? What about other simple local algorithms?*

Alternatively, we might investigate whether the cut condition is sufficient for balancing multiple loads if we restrict our attention to specific networks. For example, it is well known that for trees or cycles, the cut condition implies the existence of multi-commodity flows, and we might hope that it would hence be sufficient to prove safety.

We would also be interested in algorithms that are allowed to discard some small amount (or fraction) of jobs. In many cases this can cause significant improvement in the ability of an algorithm to balance load. This makes even more sense in the scenario of packet routing. The main questions here would be:

Open Question 7.1.11 *If the algorithm were allowed to discard a small fraction of jobs (or packets), would some version of* $\texttt{MCLB}_\theta$ *be safe against an adversary obeying the cut condition?*

Open Question 7.1.12 *If the algorithm were allowed to discard a small fraction of jobs (or packets), would some version of* $\texttt{SCLB}_\theta$ *be safe even against much more powerful adversaries?*

Especially for packet routing, the above questions are still interesting even if we had to discard a *large* fraction of packets. If an algorithm were able to discard 90% of the packets and by doing so become safe against a powerful adversary, this result would imply that against such an adversary, we are able to achieve a 10% rate of throughput. Depending on the exact application, such results can be extremely important.

7.2 Further Directions

Our research suggests a number of important fundamental directions for further research, some of which we describe in this section.

The entity framework A lot of the questions we address in this thesis can be expressed coherently using what one could call *the entity framework* (see e.g., [65]). We are given a network of strategic agents. Suppose there also exists some single entity with an objective function. This entity is usually considered to be a centralized authority, but could also correspond to one particular agent in the network. The entity also has some limited power to affect the agents. The question is: how well can this entity do for itself? Given its limited control, how big of a personal objective can it achieve, and can it determine how to use its power to achieve it?

The questions we ask about the price of stability in the Connection Game can be stated simply using the entity framework. The entity is a centralized authority desiring maximum social welfare, and it achieves this by calculating the best possible Nash equilibrium and then suggesting it to the agents.

In general, it is important to study the behavior of strategic agents in various networks with the aim of finding mechanisms which entities (whether altruistic, self-interested, or malicious) could use to influence these agents. The goals should be to analyze how much influence entities with various power and desires have in various network scenarios. This power could be none at all, could be the ability to pay agents money from a limited budget, could be the ability to destroy edges, or an ability to take over a constant number of agents, etc. In many cases, this analysis should either yield a way for improving the global qualities of a network, or a way for preventing undesirable entities from gaining undue influence over it. Traditional optimization deals with questions of how to find, given a set of constraints, a solution which meets these constraints and is close to optimal. There is a need for a framework just as developed as traditional optimization which answers the same questions, but assumes that the agents concerned are self-interested.

A start on this framework would be to consider entities with such powers as the ability to destroy or create a few edges, the ability to take over a small number of players, or the ability to bribe a small number of players with a small (but larger than ε) budget. These entities should either be altruistic, self-interested, or malicious. A self-interested entity could be a single agent who has the ability to bribe other agents, for example. Note also that there are many notions of "social good", so there can also be many variants of an altruistic objective. Looking at such entities in the context of the Connection Game will provide a greater understanding of how an entity could affect the above network design process.

What is a good outcome? Ideally, the results of future research will yield a deeper understanding of how to get self-interested agents to behave. We must now address the question of what constitutes a "desirable" outcome. The goal of the entities in this framework is to form a good outcome with respect to their objective function. In this thesis, we usually assumed that any desirable outcome of the interactions between self-interested agents must be a stable outcome. The most popular notion of stability is a Nash equilibrium, which is a solution such that, once implemented, no single agent would want to deviate from it. However, this notion is rather limited. If two agents decided to join forces, for example, a Nash equilibrium would not be stable anymore, and there are examples of games where the actual resulting outcomes are not Nash equilibria (see e.g. [53]). There are other notions of stability that need to be explored in games similar to the Connection Game, such as stability in the face of limited numbers of agents acting cooperatively together [14]. If agents could form small *coalitions*, we may want to find a solution such that no coalition of small size would want to deviate from this solution.

Since we are interested in designing mechanisms to influence agents in real networks, where these agents may have only limited local information, another important solution concept needing study in this context is the Bayes-Nash equilibrium, which is a Nash equilibrium with agents

having certain beliefs formed from their knowledge about the game they are playing which may not correspond to reality.

Approximate Equilibria and Approximation Algorithms Techniques from approximation algorithms are desperately needed to address the types of questions which arise in this thesis. Besides having Nash equilibria which are approximately optimal with respect to the centrally optimal solution, there are also many notions here of approximately stable solutions. The definition of an ε-approximate Nash equilibrium that we used in this thesis was that of a solution where no agent can become more than a factor of $(1 + \varepsilon)$ better off by deviating. This solution could correspond to a stable solution in the scenario where there is an ε cost incurred by any agent for deviating. If an "entity" (e.g., a central authority with limited power) tried to implement such a solution (because there is no good exact Nash equilibrium, for example) it could suggest this solution and pay an ε-fraction to each agent for it not to deviate (assuming the entity has this power), thereby ending up with a good stable solution and spending only a small amount of money to get it. Most definitions of an approximate stable solution can be interpreted as an exact stable solution after some small manipulation by an entity. For example, if we defined an approximate Nash equilibrium as a solution that would be an exact Nash equilibrium if k edges were removed from the graph, then this solution could be interpreted as an exact stable solution induced by an entity that has the power to remove k edges. Since it is important to study entities with various powers to affect the network at their disposal, this puts the focus on various notions of approximation as well, and raises questions about the ability to computationally find and implement approximate Nash equilibria in games with strategic agents.

Cost Sharing Methods in the Fair Connection Game There is another interesting entity power, which we have already begun studying in this thesis. The Shapley-value sharing rule in the Fair Connection Game is really a cost sharing mechanism, which we assume is enforced by an underlying protocol, and therefore by some entity power. This mechanism can be used to guarantee some better global properties of the network. This cost-sharing mechanism, though, is only one of many possible. It would be wonderful to explore the existence of other such mechanisms which may guarantee even better global properties. These mechanisms must be judged not only on the basis of the global properties of the networks formed if all the agents obeyed the mechanism, but also on whether the agents would even desire to participate in the mechanism at all. These mechanisms also must be fully distributed, use only local information, and use only limited power, since while a mechanism could be designed which guaranteed centralized optimal behavior, the cost-sharing must be done quickly and locally, and therefore on an edge-by-edge basis. In essence, there is a tradeoff here between the power and knowledge available to the mechanism, and the quality of the solutions it can produce, and one of the major research goals should be to examine this tradeoff and find mechanisms with very small power and yet large effect.

Player Entities One important type of entity not mentioned yet is an entity that corresponds to a particular player/agent. In the network game we defined, as well as in the usual network games, there is an assumption that the players are myopic, i.e., they only do what is best for them given the current configuration, and do not think ahead in any way (otherwise Nash equilibria would not make any sense here, we would need a different solution concept). We are also assuming that each agent cannot affect the others except by changing his strategy (e.g., it cannot collude with others). However, what if one agent decided that it will try to do better than the rest, obtains more information, and gets additional power? The power it obtains may be anything we mentioned above, such as the ability to pay money to other players. Then we can think of this agent as an entity whose objective is the utility of this single player, and who has limited power as described above. The questions we would ask in this context would tell us how much this single player could affect the network if it had limited power.

Notice that in the above description we gave the entity a somewhat subtle power: the power of information. We assumed that the players only know local information and therefore behave myopically. However, an entity with more global information (whether it is a particular player with listening posts or a central authority) can compute a Nash advantageous to itself and then either suggest it to all the other players or use a tiny bit of influence to get them to agree to this particular Nash. For a central authority, it might make sense to have access to this global information, since it is possibly observing the network over a longer period of time.

Routing The questions described above do not apply only to network design, but to routing on an existing network as well, since the Fair Connection Game becomes Selfish Routing with atomic players if we change the edge costs to be convex functions.

Game Dynamics All of the questions addressed above focus on using incentives to push the strategic agents into a desirable stable solution. One reason for this is that the dynamic behavior of agents is difficult to analyze and predict. In Chapter 4, we showed that in the Fair Connection Game, any sequence of deviations by agents unhappy with the current configuration will converge to a Nash equilibrium. In real systems, however, multiple agents might change their strategy at the same time, and we saw many cases in this thesis where even best response dynamics may not converge to a Nash equilibrium. Studying convergence issues of various dynamics (especially if these dynamics are close to how actual players interact) is very important in these types of network games, since it is not always that a central authority can suggest a stable solution.

In addition, we do not have to insist on stable solutions, if the nature of the agent interaction is such that all the configurations formed by them have good global properties (and we do not insist on stability as one of these properties). For example, if best response dynamics quickly converge to a (possibly unstable) solution that is close to the centralized optimum, or if there exists a small cycle of best-responses such that all the solutions in this cycle are of high quality, this may be good enough. See [59] for some of these ideas.

Load Balancing and Distributed Computing In Chapter 6, we examined simple dynamics in a network where each agent owns a node corresponding to a processor, and an adversary is adding jobs into the network. We showed that a simple local protocol has good global properties, specifically that the load imbalance in the network never becomes unbounded. In Chapter 6, the agents were considered obedient, but if the agents were instead self-interested with local knowledge, this protocol with small incentives added would not force the agents to do anything against their will. With the advent of massively distributed computing, in the near future a large amount of computation will be done in a distributed fashion on many machines. These machines are likely to be independent instead of centrally controlled agents. In a scenario where every machine wants the other machines to do its computations, we have a need for distributed protocols which balance the load without forcing individual machines to perform actions they would regard as unfair. As before, we could also ask questions about different types of entities here, such as in what ways a malicious entity could sabotage the computational process, and in what ways a particular agent could manipulate this process to get others to perform its calculations without performing any himself.

BIBLIOGRAPHY

[1] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM Journal on Computing*, 24(3) 445-456 (1995).

[2] W. Aiello, B. Awerbuch, B. Maggs, and S. Rao. Approximate load balancing on dynamic and asynchronous networks. *Proc. ACM Symp. on Theory of Computing* 1993.

[3] W. Aiello, E. Kushilevitz, R. Ostrovsky, and A. Rosén. Adaptive packet routing for bursty adversarial traffic. *Proc. ACM Symp. on Theory of Computing* 1998.

[4] A. Akella, R. Karp, C. Papadimitrou, S. Seshan, and S. Shenker. Selfish Behavior and Stability of the Internet: A Game-Theoretic Analysis of TCP. In *Proceedings of SIGCOMM*, 2002.

[5] D. M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, F. T. Leighton, Z. Liu. Universal Stability Results for Greedy Contention–Resolution Protocols. *Proc. 37th IEEE Symp. on Foundations of Computer Science*, 1996.

[6] E. Anshelevich, A. Dasgupta, J. Kleinberg, É. Tardos, T. Wexler, T. Roughgarden. The Price of Stability for Network Design with Fair Cost Allocation. In *Proc. 45th Annual IEEE Symposium on Foundations of Computer Science*, 2004.

[7] E. Anshelevich, A. Dasgupta, É. Tardos, T. Wexler. Near-Optimal Network Design with Selfish Agents. In *Proc. 35th ACM Symposium on Theory of Computing*, 2003.

[8] E. Anshelevich, D. Kempe, J. Kleinberg. Stability of Load Balancing Algorithms in Dynamic Adversarial Systems. In *Proc. 34th ACM Symposium on Theory of Computing*, 2002.

[9] A. Archer and É. Tardos. Frugal Path Mechanisms. In *SODA* 2002.

[10] B. Awerbuch, P. Berenbrink, A. Brinkmann and C. Scheideler. Simple routing strategies for adversarial systems. *Proc. IEEE Symp. on Foundations of Computer Science* 2001.

[11] B. Awerbuch and F. T. Leighton. A simple local-control approximation algorithm for multicommodity flow. *Proc. IEEE Symp. on Foundations of Computer Science* 1993.

[12] B. Awerbuch and F. T. Leighton. Improved approximation algorithms for the multicommodity flow problem and local competitive routing in dynamic networks. *Proc. ACM Symp. on Theory of Computing* 1994.

[13] V. Bala and S. Goyal. A Non-Cooperative Model of Network Formation. In *Econometrica* 68(2000), pages 1181–1229.

[14] B. D. Bernheim, B. Peleg, and M. Whinston. Coalition proof Nash equilibrium: Concepts. *Journal of Economic Theory*, 42(1):1–12, 1989.

[15] M. Beckmann, C. B. McGuire, and C. B. Winsten. *Studies in the Economics of Transportation.* Yale University Press, 1956.

[16] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, D. Williamson, "Adversarial Queueing Theory," *Proc. 28th ACM Symp. on Theory of Computing*, 1996.

[17] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha. Approximation Algorithms for Directed Steiner Problems. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[18] H. Chen, T. Roughgarden. Unpublished manuscript, 2005.

[19] G. Christodoulou and E. Koutsoupias. On the price of anarchy and stability of correlated equilibria of linear congestion games. Unpublished manuscript, 2005.

[20] J.R. Correa, A. Schulz, and N. Stier Moses. Selfish Routing in Capacitated Networks. In *Mathematics of Operations Research*, 29(4), pages 961–976, 2004.

[21] M. Crovella, M. Harchol-Balter, C. Murta, "Task assignment in a distributed system: Improving performance by unbalancing load," *Proc. ACM Sigmetrics'98 Conf. on Measurement and Modeling of Computer Systems*, 1998.

[22] A. Czumaj, P. Krysta, and B. Vöcking. Selfish Traffic Allocation for Server Farms. In *Proceedings of the 34th Annual ACM Symposium on the Theory of Computing*, pages 287–296, 2002.

[23] A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. In *Proceedings of the 13th Annual Symposium on Discrete Algorithms*, pages 413–420, 2002.

[24] X. Deng, C. Papadimitriou, and S. Safra. On the Compexity of Equilibria. In *STOC* 2002.

[25] N. Devanur, C. Papadimitriou, A. Saberi, and V. Vazirani. Market equilibrium via a primal-dual-type algorithm. In *FOCS* 2002.

[26] D. Dutta, A. Goel, and J. Heidemann. Oblivious AQM and Nash Equilibrium. In *Proceeding of the IEEE Infocom*, 2003.

[27] D. Eager, E. Lazowska, J. Zahorjan, Adaptive Load Sharing in Homogeneous Distributed Systems. *IEEE Transactions on Software Engineering* 12(1986).

[28] E. Even-Dar, A. Kesselman, and Y. Mansour. Convergence Time to Nash Equilibria. In *Proc. of the 30th International Colloquium on Automata, Languages and Programming* (ICALP '03), pages 502–513. Springer-Verlag, 2003.

[29] A. Fabrikant, A. Luthra, E. Maneva, S. Papadimitriou, and S. Shenker. On a Network Creation Game. In *PODC*, 2003.

[30] A. Fabrikant, C. Papadimitriou, K. Talwar. The complexity of pure Nash equilibria. In *STOC*, 2004.

[31] U. Feige. A treshold of $\lg n$ for approximating set-cover. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 314-318, 1996.

[32] J. Feigenbaum, C. Papadimitriou, S. Shenker. Sharing the Cost of Multicast Transmissions. *Journal of Computer and System Sciences* 63 (2001), 21–41.

[33] J. Feigenbaum, C. Papadimitriou, R. Sami, S. Shenker. A BGP-based Mechanism for Lowest-Cost Routing. In *PODC* 2002.

[34] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13, 2002.

[35] L. Ford, D. Fulkerson. Maximal flow through a network. *Can. J. Math* 8(1956).

[36] E. Friedman, A. Greenwald, and S. Shenker. Learning in Network Context: Experimental Results from Simulations. *Games and Economic Behavior,* 35:80-123, 2001.

[37] E. Friedman, M. Shor, S. Shenker, and B. Sopher. Experiment on Learning with Limited Information: Non-convergence, Experimentation Cascades, and the Advantage of Being Slow. *Games and Economic Behavior,* 47:2 (2004), pages 325–352.

[38] D. Fudenberg and J. Tirole. *Game Theory.* MIT Press, 1991.

[39] D. Gamarnik. Stability of adaptive and non-adaptive packet routing policies in adversarial queueing networks. *Proc. ACM Symp. on Theory of Computing* 1999.

[40] J. Hershberger and S. Suri. Vickerey Prices and Shortest Paths: What is an edge worth? In *FOCS* 2001.

[41] B. Ghosh, F. T. Leighton, B. Maggs, S. Muthukrishnan, C. Plaxton, R. Rajaraman, A. Richa, R. Tarjan and D. Zuckerman. Tight analyses of two local load balancing algorithms. *Proc. ACM Symp. on Theory of Computing* 1995.

[42] M. Goemans, and D. Williamson. A General Approximation Technique for Constrained Forest Problems. *SIAM Journal on Computing*, 24:296-317, 1995.

[43] H. Heller and S. Sarangi. Nash Networks with Heterogeneous Agents. Working Paper Series 2001, E-2001-1, Virginia Tech.

[44] Shai Herzog, Scott Shenker, Deborah Estrin. Sharing the "Cost" of Multicast Trees: An Axiomatic Analysis. IEEE/ACM Transactions on Networking, Dec. 1997.

[45] T. C. Hu. Multi-commodity network flows. *Operations Research* 11(1963).

[46] K. Jain and V. Vazirani. Applications of Approximation Algorithms to Cooperative Games. In *Proceedings in the Annual ACM Symposium on Theory of Computing*, 2001.

[47] R. Johari and J. Tsitsiklis. Efficiency loss in a network resource allocation game. *Mathematics of Operations Research* 29 (3): 407–435.

[48] S. Kapoor and R. Garg. Auction Algorithms for Market Equilibrium. In *STOC*, 2004.

[49] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, volume 8 (1997), 33–37.

[50] K. Kent and D. Skorin-Kapov. Population monotone cost allocation on mst's. In *Operations Research Proceedings* KOI, 43–48, 1996.

[51] A. Kesselman, D. Kowalski, and M. Segal. Energy Efficient Connectivity in Ad Hoc Networks from User's and Designer's Perspective. In *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(1), pp. 15-26, 2005.

[52] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413, 1999.

[53] D.M. Kreps. *Game Theory and Economic Modeling.* Oxford University Press, Oxford, UK, 1990.

[54] F. T. Leighton, S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM* 46(1999).

[55] L. Libman, A. Orda. Atomic Resource Sharing in Noncooperative Networks. In *Telecommunication Systems*, 17:4, 385–409, 2001.

[56] N. Linial, E. London, Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica* 15(1995).

[57] M. Mavronicolas and P. Spirakis. The price of selfish routing. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 510–519, 2001.

[58] F. Meyer auf der Heide, B. Oesterdiekhoff and R. Wanka. Strongly adaptive token distribution. *Algorithmica* 15(1996).

[59] V. Mirrokni and A. Vetta. Convergence issues in competitive games. In *Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization* (APPROX), pages 183–194, 2004.

[60] M. Mitzenmacher, On the Analysis of Randomized Load Balancing Schemes. In *Proc. ACM Symp. on Parallel Algorithms and Architectures*, 1997.

[61] D. Monderer, D. Samet. Variations of the Shapley Value. In *Handbook of Game Theory Vol. III* (Ed. R.J. Aumann and S. Hart), Elsevier Science, 2002.

[62] D. Monderer, L. Shapley. Potential Games. *Games and Economic Behavior* 14(1996), 124–143.

[63] H. Moulin, S. Shenker. Strategyproof Sharing of Submodular Costs: Budget Balance Versus Efficiency. Duke Economics Working Paper 96-31, 1996.

[64] S. Muthukrishnan and R. Rajaraman. An adversarial model for distributed dynamic load balancing. *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures* 1998.

[65] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1/2), 2001.

[66] A. Ronen. Algorithms for rational agents. In *Conference on Current Trends in Theory and Practice of Informatics*, pp. 56–70, 2000.

[67] C. Papadimitriou. Algorithms, Games, and the Internet. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 749–753, 2001.

[68] C. Papadimitriou and T. Roughgarden. Computing equilibria in multiplayer games. In *SODA* 2005.

[69] D. Peleg and E. Upfal. The token distribution problem. *SIAM Journal on Computing* 18(1989).

[70] G. Robins and A. Zelikovsky Improved Steiner Tree Approximation in Graphs. In *Proceedings 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–779, 2000.

[71] R. W. Rosenthal. The network equilibrium problem in integers. *Networks*, 3:53-59, 1973.

[72] T. Roughgarden. The price of anarchy is independent of the network topology. In *Proceedings of the 34th Annual ACM Symposium on the Theory of Computing*, 2002.

[73] T. Roughgarden. *Selfish Routing.* PhD thesis, Cornell University, May 2002.

[74] T. Roughgarden. Stackelberg scheduling strategies. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 104–113, 2001.

[75] T. Roughgarden and É. Tardos. How bad is selfish routing? In *Journal of the ACM*, 2002.

[76] A. Schulz and N. Stier Moses. On the Performance of User Equilibria in Traffic Networks. In *ACM/SIAM Symposium on Discrete Algorithms* (SODA), 2003, pages 86–87.

[77] P. Seymour. A short proof of the two-commodity flow theorem. *J. of Combinatorial Theory* 26(1979).

[78] S. Shenker. Making Greed Work in Networks: A Game-Theoretic Analysis of Switch Service Disciplines. In *IEEE/ACM Transactions on Networking*, pages 819–831, 1995.

[79] B. Shirazi, A. Hurson, K. Kavi. *Scheduling and Load Balancing in Parallel and Distributed Systems*, IEEE Computer Society Press, 1995.

[80] É. Tardos. *Network Games.* In *Proceedings of the 36th Annual ACM Symposium on the Theory of Computing*, 2004.

[81] A. Vetta. Nash equilibria in competitive societies with applications to facility location, traffic routing and auctions. In *Proceedings of the Annual IEEE Symposium on the Foundations of Computer Science*, 2002.

[82] M. Wooldridge. *An Introduction to Multiagent Systems.* John Wiley & Sons, 2002.