Network Flow Problems and Congestion Games: Complexity and Approximation Results

by

Carol Meyers

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2006

Author .....
Sloan School of Management
May 18, 2006

Certified by .....
Andreas S. Schulz
Class of 1958 Associate Professor of Operations Research, MIT
Thesis Supervisor

Accepted by .....
James B. Orlin
Edward Pennell Brooks Professor of Operations Research
Co-Director, Operations Research Center, MIT

Network Flow Problems and Congestion Games: Complexity and Approximation Results

by

Carol Meyers

Submitted to the Sloan School of Management
on May 18, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

In this thesis we examine four network flow problems arising in the study of transportation, communication, and water networks. The first of these problems is the Integer Equal Flow problem, a network flow variant in which some arcs are restricted to carry equal amounts of flow. Our main contribution is that this problem is not approximable within a factor of $2^{n(1-\epsilon)}$, for any fixed $\epsilon > 0$, where n is the number of nodes in the graph. We extend this result to a number of variants on the size and structure of the arc sets.

We next study the Pup Matching problem, a truck routing problem where two commodities ('pups') traversing an arc together in the network incur the arc cost only once. We propose a tighter integer programming formulation for this problem, and we address practical problems that arise with implementing such integer programming solutions. Additionally, we provide approximation and exact algorithms for special cases of the problem where the number of pups is fixed or the total cost in the network is bounded.

Our final two problems are on the topic of congestion games, which were introduced in the area of communications networks. We first address the complexity of finding an optimal minimum cost solution to a congestion game. We consider both network and general congestion games, and we examine several variants of the problem concerning the structure of the game and its associated cost functions. Many of the problem variants are NP-hard, though we do identify several versions of the games that are solvable in polynomial time.

We then investigate existence and the price of anarchy of pure Nash equilibria in k-splittable congestion games with linear costs. A k-splittable congestion game is one in which each player may split its flow on at most k different paths. We identify conditions for the existence of equilibria by providing a series of potential functions. For the price of anarchy, we show an asymptotic lower bound of 2.4 for unweighted k-splittable congestion games and 2.401 for weighted k-splittable congestion games, and an upper bound of 2.618 in both cases.

Thesis Supervisor: Andreas S. Schulz
Title: Class of 1958 Associate Professor of Operations Research, MIT

Acknowledgments

Where would I be without all of the cool people in my life. (And how could I have made it through five years at MIT otherwise?) My thanks go first to my advisor, Andreas Schulz, for the support, inspiration, and helpfulness he showed me over the course of my graduate education. I have never met anyone with a keener eye for detail, and all of my research has benefitted greatly from his amazing ability to read not only for content but also for interesting new research directions. I can never thank him enough for all of the care and attention he has shown to me at MIT, for which I am truly grateful.

I would also like to thank Jim Orlin, for not only serving on my thesis committee, but also acting as my 'surrogate advisor' this past year while Andreas was on sabbatical in Zurich. Our weekly meetings were invaluable in helping me stay on track with my thesis and progress toward graduation. In these meetings, I learned that not only is Jim a top-notch researcher, but he is also a top-notch person, and I value our friendship especially highly.

Much appreciation goes to Tom Magnanti, for making time in his terribly busy schedule as the Dean of Engineering to serve on my thesis committee. My thesis improved because of his helpful comments, and my defense benefitted from using his beautiful conference room!

An enormous debt of gratitude is owed to Juliane Dunkel, for reading this entire thesis and providing incredibly helpful comments. Her ability to read for style, clarity, correctness, and organization is truly outstanding. I am incredibly appreciative of her spending so much time to make this thesis better. She definitely went above and beyond the call of duty.

I would also like to express how much I have enjoyed working with Birger Wernerfelt this past year, on his project in the marketing group. His abundant good cheer and positive outlook made the experience tremendously pleasurable, and his advice and understanding in the job search process was invaluable. What a great experience this has been for me.

My classmates at MIT have been an awesome source of helpfulness, support, and (most importantly!) good times these past five years. My ex-officemates Michele Aghassi and Susan Martonosi proved to be two of the best friends I could hope for. We cheered each other through the hurdles of quals, generals, and the occasionally agonizing research process,

and my time would have been far less enjoyable without their company. Thanks also to my current officemates, Guillaume Roels and Théo Weber, for being so much fun to be around.

I have learned so much from Andreas's other advisees, both in our group Mittagsseminar meetings and outside the classroom. Nico Stier-Moses and José Correa were wonderful role models for me as I advanced in the program, and I owe much to them. Special thanks go to Nico for all of his advice and computer help, and to José for his amazing book collection. I also learned a great deal from Pranava Goundan, Nelson Uhan, and Juliane Dunkel, for which I am very appreciative. Nelson has the most commanding grasp of LaTeX of anyone I know, and he is always quick to share his help and his jellybeans. Pranava had to endure my numerous complaints this past year as the ORC computer administrator, which he did with an admirable degree of good humor. Finally, though Dan Stratila is not one of Andreas's advisees, I owe him many thanks for his unflagging enthusiasm in combinatorial optimization, and his support (computer and otherwise) and friendship over the years.

There are too many others in the ORC to thank, but I want to say what a wonderful experience it has been knowing you all. You are are such great people, and I'm so happy to have had these five years with you. Special thanks also go to Paulette Mosley, Laura Rose, Andrew Carvalho, and Veronica Mignott for oustanding administrative support.

I would never have reached MIT without my college advisor, Tim Hsu, who taught me how to do research, and to my high school calculus teacher, Joanne Robison, who got me interested in math in the first place. They both have been such a wonderful influence in my life. Thanks also to Dana Tidwell, for being my best friend since we were in third grade.

Finally, and most importantly, I would like to thank my family, starting with my fiancé Mike, whom I started dating on the very first day of grad school and who defended his thesis the day after mine. He makes every day exciting, and I love him so much. Thanks also to my parents, Barbara and Tom, who have always been enormously supportive. I never had any doubt in their confidence in me. My grandfather, Harwood Kolsky, actually accessed this thesis online more than anyone else– it was great knowing that he was so interested! I can only hope to be as socially and intellectually active as he is when I am 85 years old.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this section we preview the problems and results that will be addressed in the thesis.

1.1 The Integer Equal Flow Problem

In Chapter 3, we examine a generalization of the integer network flow problem known as the Integer Equal Flow problem. The setup is the same as in a standard network flow problem, except in addition we are given sets $R_1, R_2, \ldots, R_\ell$ of disjoint groups of arcs, with the requirement that all arcs in the same set must carry the same amount of flow. Even, Itai, and Shamir [32] have shown that the maximum flow version of this problem is NP-hard, even when the capacity of each arc is 1. We show that this problem is not approximable within a factor of $2^{n(1-\epsilon)}$, for any fixed $\epsilon > 0$, where n is the number of nodes in the graph. This result holds even if the cardinality of each arc set is 2. For the variant where the number of arc sets is fixed, we show that the minimum cost flow version is solvable in polynomial time. We then extend these results to the factor-α flow problem, in which the flow on any two arcs in set R_k must be within a given factor of $\alpha \geq 1$.

1.2 The Pup Matching Problem

Chapter 4 addresses the Pup Matching problem, a variant of multicommodity flow where two commodities ('pups') traversing an arc together incur the arc cost only once. We assume

that the network is directed and that all arcs have infinite capacity. Bossert [16] has shown that this problem is NP-hard, even in the case of a single source and a single commodity.

We propose a new integer programming formulation for this problem and prove that the linear programming relaxation of this formulation provides a stronger lower bound than previous formulations. We show that if certain properties hold, the gap between the optimal IP and LP solutions is at most a factor of $\frac{4}{3}$. We conjecture that the bound of $\frac{4}{3}$ holds overall and offer intuition as to why this seems to be the case.

Next we discuss one of the practical problems with implementing a network flow solution, known as the 'waiting ring' phenomenon. Somewhat similar to the occurrence of deadlocks in databases, waiting rings occur when an integer programming solution does not translate to a feasible pup routing with the same objective function value. We discuss how this phenomenon arises and we provide an integer programming formulation of the problem for the case when waiting rings are not permitted.

We then examine two variants of the Pup Matching problem. The first variant we address is the K-Pup Problem, which is the Pup Matching problem restricted to a fixed number K of pups. The second variant is the C-Problem, which asks the question of whether a solution to the Pup Matching Problem exists of cost less than or equal to a fixed value of C. We show that both problems are solvable in polynomial time when waiting rings are forbidden. When waiting rings are allowed, we give approximation algorithms for the K-Pup problem and polynomial time algorithms for small values of C in the C-Problem.

Next we take a slightly different tactic, exploring the Pup Matching problem from the perspective of a *noncooperative game* rather than an optimization problem. Accordingly, we define and discuss properties of Nash equilibria of the Pup Matching Problem. We show that the cost of a user-optimal solution is always within a factor of 2 of the cost of an optimal solution. We also discuss variants of Nash equilibria, and show how their cost compares to that of an optimal solution.

Finally, we show that the Capacitated Pup Matching problem is not approximable within a constant factor. We conclude by interpreting all of these results and offering sug-

gestions for avenues of future research.

1.3 Complexity in Congestion Games

We investigate complexity issues related to congestion games in Chapter 5. In particular, we provide a full classification of complexity results for the problem of finding an optimal minimum cost solution to a congestion game, under the model of Rosenthal [73]. We consider both network and general congestion games, and we examine several variants of the problem based on the structure of the game and the properties of its associated cost functions. Many of these problem variants are NP-hard, and some are even hard to approximate within a finite factor. We also identify several versions of the problem that are solvable in polynomial time.

1.4 Equilibria in k-Splittable Congestion Games

In Chapter 6, we investigate problems of existence, computability, and the price of anarchy of pure Nash equilibria in k-splittable network congestion games with linear costs. A k-splittable network congestion game is a congestion game in which each player may split its flow along at most k paths, forming an intermediate problem to the splittable and unsplittable games previously studied in the literature.

We show that Nash equilibria always exist in weighted k-splittable games, with the added requirement that the flow on each arc must be a multiple of an arbitrarily small number. We also show that such equilibria may be computed in pseudopolynomial time, and we identify cases in which a solution can be verified to be a Nash equilibrium in polynomial time. With regards to the price of anarchy, we show an asymptotic lower bound of 2.4 for unweighted k-splittable network congestion games and 2.401 for weighted k-splittable network congestion games, and an upper bound of 2.618 in both cases. We finally prove that the price of anarchy for k-splittable flows in a given instance needs not be monotone with the value of k.

Chapter 2

Preliminaries

In this section, we define some of the terminology that will be used in subsequent chapters of the thesis.

2.1 Computational Complexity

We now provide a brief overview of complexity theory, highlighting some of the vocabulary to be found later in the thesis. This is by no means intended to be a thorough account; for a more complete treatment see [43] or [82].

The study of complexity issues in optimization problems was initiated in the early 1970's as a method of classifying the tractability of certain algorithmic problems. At the time, it was noticed that large instances of some problems were relatively easy to solve within a reasonable amount of time, while for others there was little to do short of exhaustive enumeration. Several classes of problems were proposed to quantify the observed differences in solution times, the most well-known of which are the classes P and NP. The class P contains all problems that are solvable in polynomial time using a deterministic Turing machine, or equivalently via an algorithm that runs in time polynomial in the size of the input. The class NP includes all problems that are solvable in polynomial time using a nondeterministic Turing machine. The question of whether P=NP is one of the greatest outstanding questions

in mathematics.

In 1971, Cook [24] made the startling discovery that if the problem SAT in NP could also be shown to be in P, then all problems in NP would be in P. This initiated the study of *NP-complete* problems, of which SAT was the first member. By NP-complete, we mean that the problem is in NP and that all other problems in NP polynomially reduce to it; equivalently, there exists one NP-complete problem that polynomially reduces to it. Karp [50] went on to show that many kinds of decision problems are NP-complete.

The study of NP-complete problems also spawned the study of *NP-hard* problems. An NP-hard problem is a problem that can be polynomially reduced to from an NP-complete problem. It is similar to the definition of NP-complete problems, but without the requirement that the problem be in NP. Many natural optimization problems can be shown to be NP-hard.

To formally show that an optimization problem is NP-hard, we must find a polynomial-time reduction to that problem from an NP-complete problem. In other words, starting with an instance of the NP-complete problem, we must show there is an algorithm that will compute a corresponding instance of the NP-hard problem in polynomial time, such that 'yes' instances of the NP-complete problem translate to instances with an optimal objective above (below) a certain value, and 'no' instances translate to instances with an optimal objective below (above) that value.

Finally, we say that a problem is *strongly NP-hard* if it can be shown to be NP-hard even if all of the numbers in the input are bounded by a polynomial in the length of the input. These problems are in some sense 'more difficult' than regular NP-hard problems, in that they require a more stringent criterion for inclusion. It should be noted that not all NP-hard problems are strongly NP-hard (though the reverse direction holds); see the Partition [43] problem for an example.

2.2 Approximation Algorithms

For problems that are NP-hard, one question that arises is whether approximately good solutions can be calculated in polynomial time. This has led to the study of *approximation*

algorithms, which we now define. This topic is covered in depth in the texts of Hochbaum [47] and Vazirani [86].

An α-approximation algorithm for an optimization problem P and a factor $\alpha > 1$ is defined to be a polynomial-time algorithm for P that returns a solution within a factor of α of the optimum. In other words, if $c(OPT)$ is the cost of an optimal solution and $c(SOL)$ is the cost of the solution returned by the algorithm, then

$$\frac{1}{\alpha} \cdot c(OPT) \leq c(SOL) \leq \alpha \cdot c(OPT).$$

The left-hand inequality applies to maximization problems and the right-hand inequality to minimization problems. We call α the *performance ratio* of the algorithm.

Much like the concept of NP-hard problems, we can also show that certain problems are hard to approximate, in the sense that obtaining an approximate solution within a certain bound would allow us to solve an NP-complete problem. One way of showing that a problem is *hard to approximate within a factor of* α is if there exists a polynomial transformation from an NP-complete problem to that problem, such that 'yes' instances of the NP-complete problem translate to instances with an optimal objective above (below) a certain value, and 'no' instances translate to instances with an optimal objective below (above) a factor of $\frac{1}{\alpha}$ (α) times that value. (Other techniques may be possible as well, but this is the technique most commonly used in this thesis.)

2.3 Congestion Games

The study of congestion games is currently a popular area in the academic literature, and multiple definitions exist as to how to formulate such games. The material we present is based on the model of Rosenthal [73], who was the first to investigate such games.

Congestion games are a form of *noncooperative games* [88], which are games where players cannot participate in cooperative behavior. In other words, players do not have the option of planning together before choosing their actions. Many real-world problems can be

modeled as noncooperative games, and such games have been studied extensively since the groundbreaking work of Nash [68].

2.3.1 General Congestion Games

In a *general congestion game*, we are given a set of *resources* $A = \{a_1, \ldots, a_m\}$ and a set of *players* $P = \{1, \ldots, n\}$. Each player i possesses a set of *strategies* $\{s_{i1}, s_{i2}, \ldots, s_{ik_i}\}$, where each strategy $s_{ij} \subseteq A$ consists of a subset of the resources. Each player wishes to select and play exactly one strategy. A *solution* $s = (s_1, \ldots, s_n)$ consists of the chosen strategies for each player.

The *cost of a resource* $a \in A$ is given by a function $c_a(j)$ that computes the per-unit cost of j players using a. The cost function may be arbitrary in general, but it is restricted to being solely a function of the number of players using the resource. The *cost of a strategy* s_i is the sum of the costs of the resources associated with that strategy. The *cost of a solution* s is equal to

$$\sum_{a \in A} x_a c_a(x_a),$$

where $x_a = |i : a \in s_i|$ is the total number of players using resource a in the solution, and s_i is the strategy chosen by player i in s.

In such games, a *Nash equilibrium* (see Section 2.4) arises when no player can deviate in a given solution and improve their overall cost. A *system optimal* solution occurs when we instead choose to minimize the *total* cost, disregarding the individual preferences of the players.

2.3.2 Network Congestion Games

A *network congestion game* is a special case of a general congestion game in which resources are associated with *arcs*, strategies are associated with *simple paths*, and players are associated with *units of demand* in a network. This is a special type of minimum cost integer multicommodity flow problem where the cost per unit flow on each arc differs based on how much flow is traversing the arc.

More formally, in a network congestion game we are given a graph $G = (N, A)$ and a set of players $P = \{1, \ldots, n\}$. Each player i is associated with a pair of nodes $s_i \in N$ and $t_i \in N$, with the understanding that player i wishes to send 1 unit of flow from node s_i to node t_i. If an arc $a = (u, v)$ is labeled as:

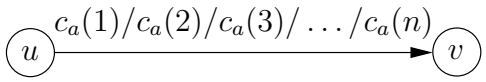

Figure 2-1: A typical arc labeling in a network congestion game

then the cost of sending 1 unit of flow along the arc is $c_a(1)$, the cost of sending 2 units of flow is $c_a(2)$ per unit (for a total cost of $2c_a(2)$), and the cost of sending k units of flow is $c_a(k)$ per unit (for a total cost of $kc_a(k)$). The goal is to route each player on a single path from its source to its sink in a minimum cost manner.

2.3.3 Weighted Network Congestion Games

One generalization of network congestion games occurs when each player i controls a positive integral number w_i units of flow. We call such a game a *weighted network congestion game*, as in [39]. The unweighted case corresponds to $w_i = 1$ for all i. Different versions of the problem may be defined (see Chapter 6) according to whether the flow is splittable or unsplittable.

If player i sends f_a^i units of flow along arc $a \in A$ in solution s, then the cost $C_i(s)$ associated with player i is given by

$$C_i(s) = \sum_{a \in A} f_a^i c_a(x_a),$$

where x_a is the total amount of flow on arc a in solution s. This is analogous to the previous definition, but now each player's cost is proportionate to the amount of flow that they use on the arc. Again, the total cost $C(s)$ of the solution is the sum of the costs associated with each of the players.

2.4 Nash Equilibria

Often in problems involving multiple players, it may be useful not only to know the optimal solution to the problem, but also to know the optimal 'stable' solution. For instance, in traffic guidance systems users are less likely to take the routes proposed to them if they perceive that they could travel faster by taking an alternate route.

The concept of *Nash equilibria* was proposed by Nash [68] as a way to quantify stable solutions. In a Nash equilibrium, no player has an incentive to improve its overall cost by switching strategies. More formally, let $1, \ldots, n$ be players, and s a solution to the game such that s_i is the strategy followed by player i. We say s is a Nash equilibrium if for all alternate strategies s_i' we have

$$C_i(s_i', s_{-i}) \leq C_i(s),$$

where the right hand side indicates the cost of the strategy to player i under s, and the left hand side indicates the cost to player i if the player switches to strategy s_i', with all other players remaining the same. (We refer here to *costs* rather than *utilities* for each player, as can be found in some of the literature.) Nash equilibria are also known as *user optimal* solutions, since they are optimal with the perspective of each of the users, and overall optimal solutions are known as *system optimal* solutions.

In the literature, distinctions are made between *pure Nash equilibria*, in which each player follows a single given strategy, and *mixed Nash equilibria*, in which each of the possible strategies are chosen stochastically with a particular fixed frequency. In this thesis we examine only pure Nash equilibria.

A natural question that arises is how far the objective value of a Nash equilibrium can be from that of a system optimal solution. The *price of anarchy* of a strategic game is defined as the largest ratio in the objective value of a Nash equilibrium to that of an optimal solution. The larger the price of anarchy is, the farther Nash equilibria may be from optimal; conversely, if the price of anarchy is 1 then all Nash equilibria are optimal solutions. This topic has received much attention in recent years; for a thorough treatment, see [27] or [75].

Chapter 3

The Integer Equal Flow Problem

In this chapter, we examine the approximability of the integer equal flow problem. Most of the approximation results we obtain are for the maximum flow version of this problem; however, standard network transformation techniques (see [5]) allow us to convert the maximum flow problem into a minimum cost flow problem and thus the same results hold for this problem as well.

3.1 Introduction

The equal flow problem was first studied by Sahni [78] as a generalization of the traditional network flow problem. Its setup is similar to a standard maximum flow problem: we are given a graph $G = (N, A)$ with capacities u_a for all $a \in A$, and a designated source node s and sink node t. However, in addition we are also given sets $R_1, R_2, \ldots, R_\ell$ of disjoint groups of arcs, with the requirement that all arcs in the same set must carry the same amount of flow. We wish to send the maximum amount of flow from s to t subject to these constraints. The special case where $\ell = 1$ is known as the simple equal flow problem. (We can also define a minimum cost flow version of this problem, by assigning costs to each of the arcs and a set demand from s to t.)

Ahuja, Orlin, Sechi, and Zuddas [6] studied the minimum cost simple equal flow

problem as a means of modeling a water resource system in Sardinia, Italy. They detailed several different methods of solving the problem, including a version of the network simplex algorithm and a parametric simplex method. They noted that the integer version of the problem is solvable optimally by using the parametric simplex method in conjunction with two minimum cost flow problems.

Recently, Calvete [17] demonstrated a version of the network simplex algorithm for solving the general minimum cost equal flow problem. While it is possible to solve the general problem in polynomial time using the simplex method, her algorithm exploits the network structure of the problem and improves upon the running time. Her key insight is in characterizing the bases of the problem, which allows her to adapt the well-known network simplex algorithm to this problem with only slight modifications.

The special case where all of the arc flows must be integral is known as the integer equal flow problem. Sahni [78] proved that the maximum flow version of this problem is NP-hard with a reduction from Non-Tautology. Later, Even, Itai, and Shamir [32] showed via a reduction from Satisfiability that the problem remains NP-hard even if the capacity of each arc is 1. Srinathan et al. [83] showed by a reduction from Exact Cover by 3-Sets that this problem also remains NP-hard if we further require that all arcs in a set R_i must originate from the same node.

Ali, Kennington, and Shetty [7] examined a special case of the integer equal flow problem where each arc set has cardinality 2. We refer to this as the paired integer equal flow problem. They developed a heuristic for solving the problem using Lagrangian relaxation and decomposition techniques. Computational experience indicated that in 'balanced' problems, where the number of equal flow pairs was low, near-feasible solutions within 1% of the optimum could be found in $1\% - 65\%$ of the time of that taken by a leading mixed integer programming solver.

Larsson and Liu [59] also used a Lagrangian dualization approach to address the paired integer equal flow problem, building on the work of Ali et al. [7]. They proposed a heuristic algorithm based on Lagrangean relaxation, where the subproblem solutions are used

to find feasible solutions to the original problem. Their method was simpler to implement and provided comparable computational results to that of Ali et al.

Goldberg, Feldman, and Stein [46] studied the maximum integer equal flow problem where each arc has a unit capacity. They showed that the paired version of this problem is solvable in polynomial time if there are at most $O(\log(|X|))$ arc pairs, where $|X|$ is the input size. They also showed that the paired version with $\Theta(|X|)$ arc pairs is NP-complete, even if the underlying graph is acyclic and of degree $\Theta(|X|)$.

The integer equal flow problem finds applications in several areas, including airline parts manufacturing [87] and crew scheduling [18, 81]. Feldman and Karger [35] show how the optimal decoding of certain Turbo codes can be accomplished using an integer equal flow problem. Srinathan et al. [83] describe a special case of the problem arising from supply chain management, where the flow on all arcs exiting a node other than the source is required to be the same. They give an approximation algorithm for the maximum flow version of this problem, which has a performance guarantee that is proportional to the degree of the source node.

Other problems that may be modeled as special cases of the integer equal flow problem include balanced network flow problems (see [41, 51]) and certain network flow problems in constraint programming [15]. Glockner and Nemhauser [44] describe a dynamic network flow problem with random arc capacities that may be considered as a special case of the equal flow problem.

In what follows, we address the approximability of the integer equal flow problem. We begin in Section 3.2 by presenting an LP formulation of the maximum equal flow problem, along with an example showing that the integrality gap can be very large. We review an NP-hardness construction due to Srinathan et al. [83] in Section 3.3. We observe that the problem of determining whether a nontrivial feasible solution exists to the maximum integer equal flow problem is strongly NP-complete. This motivates our main result in Section 3.4, which is that no $2^{n(1-\epsilon)}$-approximation algorithm exists for the maximum integer equal flow problem for any fixed $\epsilon > 0$. We then extend this argument to show that this result also

holds for two related problems, the maximum paired integer equal flow problem (Section 3.5) and the uncapacitated minimum cost equal flow problem (Section 3.6).

For a special case where the number of sets that must have equal flow is fixed, we observe in Section 3.7 that this problem is solvable in polynomial time. Finally in Section 3.8 we show that the same results hold for a generalization of the equal flow problem known as the factor-α flow problem, in which the flow on any two arcs in a given set must be within a fixed factor of α. We conclude by offering interpretations of these results.

3.2 Problem Definition

An instance of the *maximum equal flow problem* is defined as follows. We are given a directed graph $G = (N, A)$ with designated nodes s and t, and capacities u_a for all $a \in A$. In addition, we are given sets $R_1, R_2, \ldots, R_\ell \subseteq A$ of disjoint groups of arcs, with the requirement that all arcs in the same set must carry the same amount of flow. We wish to send the maximum amount of flow from s to t subject to these conditions. This problem can be formulated mathematically as:

$$
\begin{aligned}
\max \quad & v \\
\text{s.t.} \quad & \sum_{j:(s,j)\in A} x_{sj} - \sum_{j:(j,s)\in A} x_{js} = v & \\
& \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0 & \text{for all } i \in N \setminus \{s,t\} \\
& x_{i_1 j_1} = x_{i_2 j_2} & \text{for every pair } (i_1,j_1),(i_2,j_2) \in R_k,\ k = 1,\ldots,\ell \\
& 0 \le x_{ij} \le u_{ij} & \text{for all } (i,j) \in A
\end{aligned}
$$

The *maximum integer equal flow problem* is the same as above, except we constrain $x_{ij} \in \mathbb{N}$ for all arcs (i, j). This is also known as the *integral flow with homologous arcs* problem [43]. The integrality gap between optimal LP and IP solutions can be very large, as is the case in the following example:

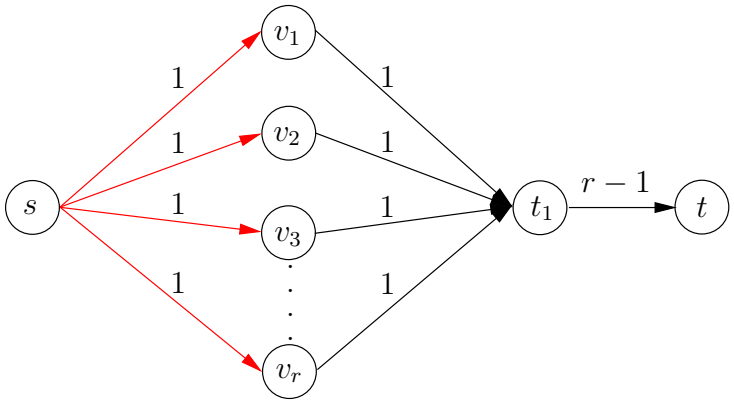

Figure 3-1: Example of a large gap in optimal LP and IP solutions

Here, there is one set of homologous arcs $\{(s, v_i)|i = 1, \ldots, r\}$; these arcs are shown colored red. The number on each arc represents its capacity. The optimal LP solution has a value of $r - 1$, which is achieved by sending $\frac{r-1}{r}$ units of flow along each of the arcs (s, v_i) and (v_i, t_1). The optimal IP solution has a value of 0, since there is no way of sending any integral amount of flow along this network. Hence the gap can be made arbitrarily large.

In fact, the LP gap can be arbitrarily large even if the cardinality of each homologous arc set is 2, as is shown in the following example:

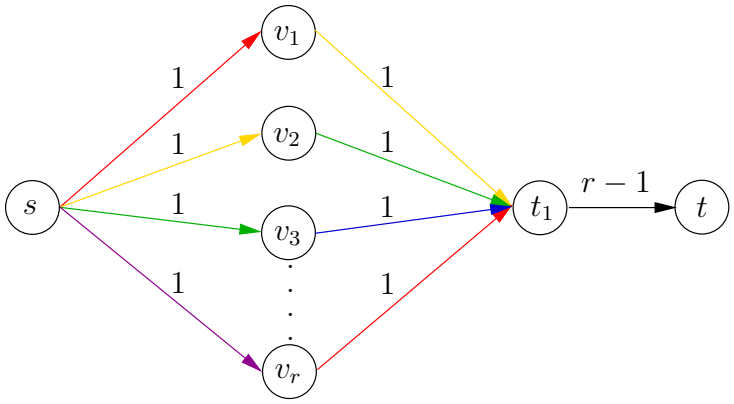

Figure 3-2: Example of a large gap where all homologous sets have size 2

Here the homologous arc sets are $\{(s, v_{i+1}), (v_i, t_1)\}$ for $i = 1, \ldots, r-1$, and $\{(s, v_1), (v_r, t_1)\}$. Note that by the way the homologous sets are constructed, all arcs (s, v_i) are 'forced' to

have equal flow, for all $i = 1, \ldots, r$. The optimal LP and IP solutions are the same as in the previous example. Hence again the gap can be made arbitrarily large.

We can equivalently define the *minimum cost integer equal flow problem*, which has the same setup as a traditional minimum cost flow problem, but additionally contains sets $R_1, R_2, \ldots, R_\ell \subseteq A$ of arcs that must have equal flow. We further address this version of the problem in Sections 3.6 and 3.7.

Sahni [78] showed that the maximum integer equal flow problem is NP-hard, as mentioned in Section 3.1. Later, Garey and Johnson [43] observed that the modification of an Even et al. [32] construction shows that the problem is NP-hard even if the capacity of every arc is 1. Srinathan et al. [83] furthered this, showing that the problem remains NP-hard even if all capacities are 1 and all arcs in a homologous set originate from the same node. We shall build on the Srinathan et al. construction, so we review their proof in the next section.

3.3 NP-hardness

Theorem 3.1 (Srinathan et al.) *The maximum integer equal flow problem is strongly NP-hard.*

Proof: We reduce from the Exact Cover by 3-Sets problem, which is strongly NP-complete [43]. This problem is:

> **Instance:** A set $\mathcal{A} = \{a_1, \ldots, a_q\}$, such that q is divisible by 3, and a collection $S = \{S_1, \ldots, S_r\}$ of 3-element subsets of $\mathcal{A}$. (Without loss of generality, we assume that the sizes of $\mathcal{A}$ and S are the same, a fact that we will use later.)
>
> **Question:** Does there exist a subcollection $S' \subseteq S$ such that each element of $\mathcal{A}$ occurs in exactly one member of S'?

Assume we are given an instance of the Exact Cover by 3-Sets problem, consisting of $\mathcal{A}$ and S. Construct an instance of the maximum integer equal flow problem as follows:

1. Create a source node s and a sink node t.

   Create q nodes $S_1, S_2, \ldots, S_q$, corresponding to elements of S.

   Create q nodes $a_1, a_2, \ldots, a_q$, corresponding to elements of $\mathcal{A}$.

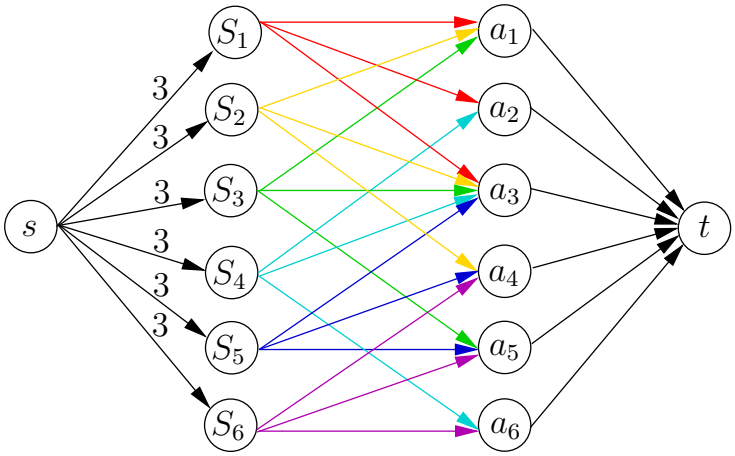

Figure 3-3: Constructed instance of the maximum integer equal flow problem

2. Add arcs: (s, S_i) for all i, of capacity 3.

   (S_i, a_j) if element a_j is contained in set S_i, of capacity 1.

   (a_j, t) for all j, of capacity 1.

3. Add homologous sets $\{(S_i, a_{i_1}), (S_i, a_{i_2}), (S_i, a_{i_3})\}$ for all i, where $S_i = \{a_{i_1}, a_{i_2}, a_{i_3}\}$.

For instance $\mathcal{A} = \{a_1, a_2, a_3, a_4, a_5, a_6\}$

$S = \{\{a_1, a_2, a_3\}, \{a_1, a_3, a_4\}, \{a_1, a_3, a_5\}, \{a_2, a_3, a_6\}, \{a_3, a_4, a_5\}, \{a_4, a_5, a_6\}\}$

the constructed graph is shown in Figure 3-3. Homologous arcs are colored the same, and all unlabeled arcs have capacity 1.

We claim that the answer to the Exact Cover by 3-Sets problem is 'yes' if and only if the value of the maximum integer equal flow on the constructed instance is equal to q.

To see this, first note that by inspection any flow on this graph must have value at most q, since this is the sum of the capacities of the arcs incident to t. If there exists an exact cover $\{S_1', S_2', \ldots, S_{\frac{q}{3}}'\}$, we can achieve a flow of value q by sending 3 units along each of the arcs (s, S_i'), and from there on through each of the nodes $a_1, \ldots, a_q$ to t. Since each element a_j appears in exactly one of the sets S_i', none of the capacities will be violated.

Conversely, if there exists a flow of value q, then we claim there exists an exact cover by 3-sets. This is since by the homologous conditions, any flow of value q must send flow

through exactly $\frac{q}{3}$ of the nodes S_i, and from there on through each of the nodes $a_1, \ldots, a_q$. By construction, this means that the set of nodes $\{S_1', S_2', \ldots, S_{\frac{q}{3}}'\}$ receiving positive flow must correspond to an exact cover by 3-sets.

Hence the maximum integer equal flow problem is strongly NP-hard. □

Corollary 3.2 *The maximum integer equal flow problem is strongly NP-hard even if all arc capacities are 1.*

Proof: In the previous construction, replace each arc (s, S_i) of capacity 3 with a copy of the graph in Figure 3-4.

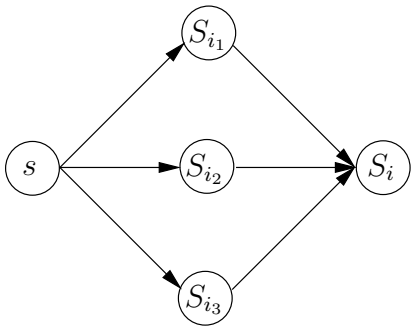

Figure 3-4: Replacement construction for each arc of capacity 3

Here each arc has capacity 1, and we have introduced the 3 new nodes S_{i_1}, S_{i_2}, and S_{i_3}. The result immediately follows. □

3.4 Hardness of Approximation

Our hardness results are motivated by the following theorem. By 'nontrivial', we mean in the following that some arc in the solution has positive flow (since the zero vector is always feasible).

Theorem 3.3 *The problem of determining whether an instance of the integer equal flow problem has a nontrivial feasible solution is strongly NP-complete.*

Proof: First notice that this problem is in NP, since any nontrivial feasible solution can be taken as a certificate.

We reduce Exact Cover by 3-Sets to the nontrivial feasibility problem. We use the same construction as Srinathan et al., with one minor alteration: we require arcs $(a_1, t), (a_2, t), \ldots, (a_q, t)$ to have equal flow.

If there is an exact cover by 3-sets, then there exists a maximum flow of value q in the original construction. Moreover, this flow must route 1 unit of flow through each of the arcs $(a_1, t), (a_2, t), \ldots, (a_q, t)$. Hence this is a nontrivial feasible solution, since these arcs all satisfy the new homologous requirement.

If there is no exact cover by 3-sets, then the maximum flow in the original construction has value less than q. This means that there is no way to route 1 unit of flow along all of $(a_1, t), (a_2, t), \ldots, (a_q, t)$ such that the homologous condition is satisfied. Thus in the new construction, the flow along all of these arcs must be 0, implying that the only feasible solution is the trivial solution.

Hence there exists an exact cover by 3-sets if and only if the corresponding equal flow instance has a nontrivial feasible solution. This implies that the problem of determining whether a nontrivial feasible solution exists is strongly NP-complete. □

A simple extension of this argument provides us with our first hardness result. In essence, we translate the problem of determining whether a nontrivial feasible solution exists into a problem of determining whether a solution of a certain cost exists. We then use the hardness of the first problem to induce a gap in the approximability of the second problem. In what follows, we use n to denote the number of nodes in the graph.

Theorem 3.4 *There is no approximation algorithm for the maximum integer equal flow problem with performance ratio less than $\frac{n}{2}$, unless P=NP.*

Proof: We again reduce from Exact Cover by 3-Sets. We use the same construction as in the previous proof, with one further modification: we add a new arc (s, t) of capacity

1. This modification ensures that the constructed instance has a nontrivial feasible solution.

By the same argument as in the previous proof, we see that if there is an exact cover by 3-sets, then the value of the maximum integer equal flow is greater than q; if there is no exact cover, then the value of the maximum integer equal flow is equal to 1.

We now express q in terms of n. Observe that our instance of the maximum integer equal flow problem has $2q + 2$ nodes: $S_1, \ldots, S_q, a_1, \ldots, a_q$, and the special nodes s and t. Hence

$$q = \frac{n-2}{2}.$$

Thus:

There is an exact cover by 3-sets	$\Rightarrow$	value of max integer equal flow is $\geq \frac{n}{2}$.
There is no exact cover by 3-sets	$\Rightarrow$	value of max integer equal flow is 1.

Thus there is no approximation algorithm for the maximum integer equal flow problem with performance ratio less than $\frac{n}{2}$, unless P=NP. □

An extension of this argument yields the main result.

Theorem 3.5 *There is no $2^{n(1-\epsilon)}$-approximation algorithm for the maximum integer equal flow problem for any fixed $\epsilon > 0$, unless P=NP.*

Proof: Let $\epsilon > 0$ be given. We again reduce from Exact Cover by 3-Sets. Create the same instance of maximum integer equal flow as in the previous proof, and modify it as follows:

1. Delete arc (s, t) and let $k = \frac{(2q+2)}{\epsilon}$.
2. Create new nodes $t_1, t_2, \ldots, t_k$.
3. Add new arcs: (s, t_i) of capacity $2^{i-1}q$, for $i = 1, \ldots, k$.
   (s, t_k) of capacity 1.
   (t, t_1) of capacity q.
   (t_{i-1}, t_i) of capacity $2^{i-1}q$, for $i = 2, \ldots, k$.

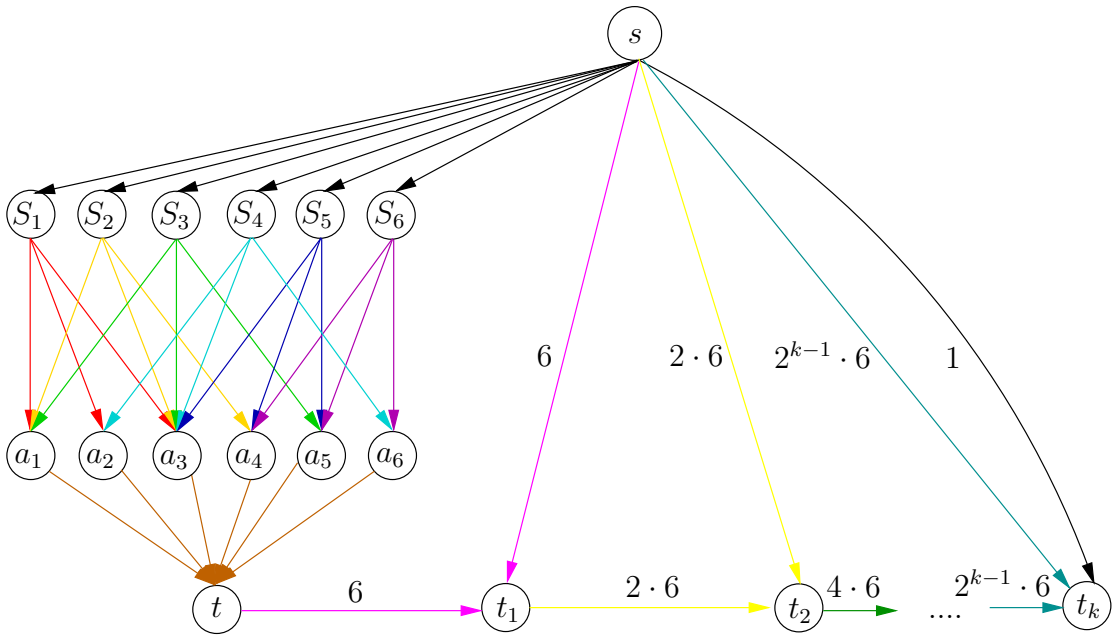

Figure 3-5: Extended construction of the maximum integer equal flow instance

4. Add homologous sets $\{(s, t_1), (t, t_1)\}$ and $\{(t_{i-1}, t_i), (s, t_i)\}$ for $i = 2, \ldots, k$.

5. Redefine the problem so that instead of a maximal $s - t$ flow, we would now like a maximal $s - t_k$ flow.

For our previous instance

$A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$

$S = \{\{a_1, a_2, a_3\}, \{a_1, a_3, a_4\}, \{a_1, a_3, a_5\}, \{a_2, a_3, a_6\}, \{a_3, a_4, a_5\}, \{a_4, a_5, a_6\}\}$

the graph is as shown in Figure 3-5 (here, $q = 6$). Homologous arcs are colored the same, and the capacities in the original portion of the graph are unchanged.

We can see by inspection that if the maximum integer equal flow is greater than q in the original graph, then it is greater than $2^k \cdot q$ in the new graph. Similarly, if the maximum flow was 1 in the original graph, then here it is also 1. Thus:

There is an exact cover by 3-sets $\Rightarrow$ value of max integer equal flow is $> 2^k$

There is no exact cover by 3-sets $\Rightarrow$ value of max integer equal flow is 1.

Now, we know

$$\begin{aligned} n &= 2q + 2 + k \\ &= (2q+2)\tfrac{1+\epsilon}{\epsilon} \end{aligned}$$

which implies that

$$\begin{aligned} k &= n \cdot \tfrac{1}{1+\epsilon} \\ &> n(1-\epsilon), \end{aligned}$$

as $(1+\epsilon)(1-\epsilon) < 1$ for all $\epsilon < 1$. Hence:

There is an exact cover by 3-sets	$\Rightarrow$	value of max integer equal flow is $> 2^{n(1-\epsilon)}$
There is no exact cover by 3-sets	$\Rightarrow$	value of max integer equal flow is 1.

Thus no $2^{n(1-\epsilon)}$ approximation algorithm exists, unless P=NP. □

3.5 The Paired Integer Equal Flow Problem

We now examine a special case of the integer equal flow problem known as the paired integer equal flow problem. This problem requires that all homologous arc sets have cardinality 2 (hence, they are pairs of arcs).

Lemma 3.6 *Any of the homologous sets of size $\ell > 3$ used in the proof of Theorem 3.5 can be converted into collections of homologous sets of size 2, such that the equal flow conditions are still enforced and ℓ new nodes are introduced.*

Proof: The only homologous sets of size greater than 2 used in the proof of Theorem 3.5 are those used in the original Exact Cover by 3-Sets gadget introduced in Theorem 3.1. These have the special structure that all arcs in a homologous set either originate from or end at a common node. A nearly identical construction may be used for both cases, so

without loss of generality we consider the case where all homologous arcs end at the same node.

We replace homologous sets of the form:

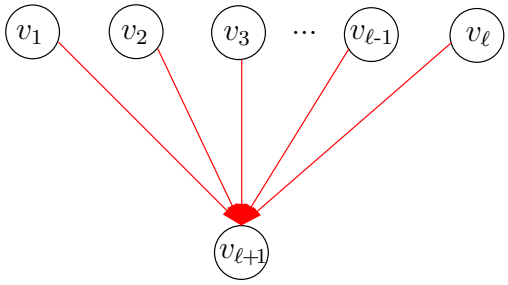

Figure 3-6: A homologous set of size ℓ

with the following collection of sets of size 2:

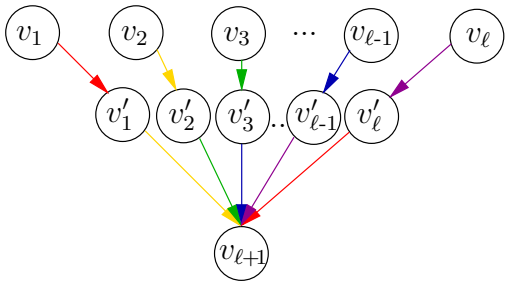

Figure 3-7: Transformed instance with ℓ sets of size 2

Here we have introduced ℓ nodes. The homologous pairs are $\{(v'_i, v_{\ell+1}), (v_{i+1}, v'_{i+1})\}$ for all $i = 1, \ldots, \ell - 1$, and $\{(v'_\ell, v_{\ell+1}), (v_1, v'_1)\}$.

We can verify by inspection that the $v_i - v_{\ell+1}$ flow must be the same for all i in both cases, by the way the homologous pairs are defined. Hence the two constructions are equivalent. □

We are now ready for the main result of this section.

Theorem 3.7 *There is no $2^{n(1-\epsilon)}$-approximation algorithm for the maximum paired integer equal flow problem for any fixed $\epsilon > 0$, unless P=NP.*

Proof: Use the same construction as in Theorem 3.5, and convert all of the homologous arc sets into sets of size 2 using the procedure in Lemma 3.6. The result follows by applying similar arguments as in Theorem 3.5, though the value of k must be (slightly) increased to compensate for a greater number of nodes in the original graph. □

3.6 Uncapacitated Minimum Cost Integer Equal Flow

In the previous sections of this chapter, we have addressed the complexity of the maximum integer equal flow problem. By extension, the hardness results we presented for this problem also apply to the (capacitated) minimum cost integer equal flow problem. This is since using standard network techniques, we can transform an instance of the maximum integer equal flow problem into an instance of the (capacitated) minimum cost integer equal flow problem.

In this section, we examine the complexity of the minimum cost integer equal flow problem with *no capacities*. Using a standard network transformation to remove arc capacities, we can show that the previous hardness results apply to the uncapacitated problem as well; however, this requires the introduction of multiple sources. By a slight manipulation in our original construction, we can show that the same arguments also hold in the single source case.

Theorem 3.8 *The uncapacitated single source minimum cost integer equal flow problem is NP-hard, and no $2^{n(1-\epsilon)}$-approximation algorithm exists for any fixed $\epsilon > 0$, unless P=NP.*

Proof: We use the same construction as in Theorem 3.5, with the following modifications:

1. Remove the capacities on all of the arcs.

2. Assign the cost of arcs (s, t_k) and (a_q, t) to be 1. Give all other arcs zero cost.

3. Assign a supply of $2^k q$ units to s, and a demand of $2^k q$ units at t. Give all other nodes zero supply and demand.

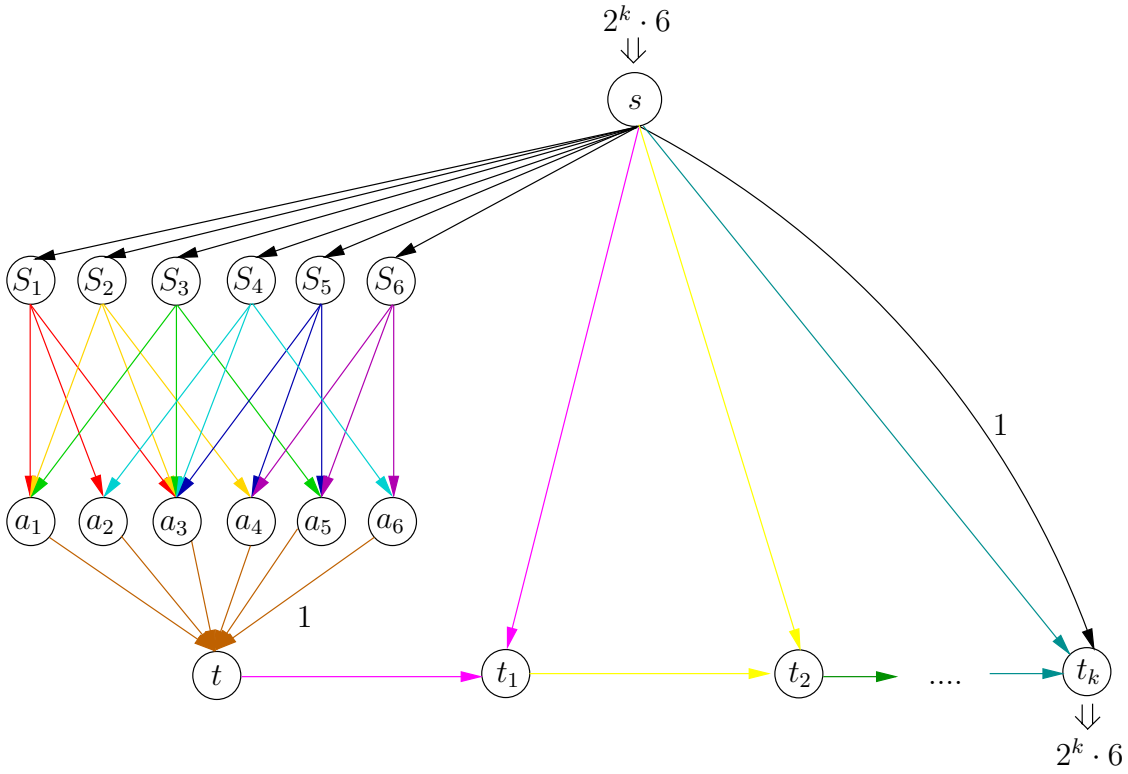

Figure 3-8: Constructed instance of the uncapacitated min cost integer equal flow problem

For the example

$A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$

$S = \{\{a_1, a_2, a_3\}, \{a_1, a_3, a_4\}, \{a_1, a_3, a_5\}, \{a_2, a_3, a_6\}, \{a_3, a_4, a_5\}, \{a_4, a_5, a_6\}\}$

from Theorem 3.5, the construction is as shown in Figure 3-8. All arcs are uncapacitated; unlabeled arcs have zero cost.

We claim that if there is an exact cover by 3-sets, then the cost of the minimum cost integer equal flow is 1; if there is no exact cover, then the cost of the minimum cost integer equal flow is greater than 2^k.

To see this, first observe if there is an exact cover by 3-sets, then we can route the flow as in Theorem 3.5 and achieve a cost of 1. This is since 1 unit of flow will traverse arc (a_q, t), and all other flow will have a cost of zero.

If there is no exact cover by 3-sets, then there is no way that all of the arcs (a_1, t), (a_2, t), $\ldots, (a_q, t)$ can simultaneously contain one unit of flow, by the way the graph is

constructed. Moreover, there is no way that these arcs can simultaneously carry *more* than one unit of flow either; as the amount of flow on arc (t_i, t_{i+1}) must be double that of arc (t_{i-1}, t_i), the total amount of required flow would then exceed the available demand in the network. Thus arc (t, t_1) must have zero flow, and by extension all of the arcs (t_i, t_{i+1}) must also have zero flow. The only feasible flow is to send all $2^k q$ units of flow along the arc (s, t_k), which gives a cost of $2^k q > 2^k$.

Using a very similar anlaysis to that in Theorem 3.5, this implies that the problem is NP-hard and no $2^{n(1-\epsilon)}$-approximation algorithm exists for any $\epsilon > 0$, unless P=NP. □

Using the same techniques as in Lemma 3.6 and Theorem 3.7, we also obtain the following result.

Theorem 3.9 *The uncapacitated minimum cost paired integer equal flow problem is NP-hard, and there is no $2^{n(1-\epsilon)}$-approximation algorithm for any fixed $\epsilon > 0$, unless P=NP.*

3.7 Integer Equal Flow with Fixed Number of Arc Sets

We now address the problem where the number ℓ of homologous arc sets is fixed. Ahuja et al. [6] have shown that this problem is solvable in polynomial time when $\ell = 1$, but they did not address the complexity for greater values of ℓ. Our results are for the minimum cost flow version of the problem, though they also hold for the maximum flow version since maximum flow problems can be formulated as a special case of minimum cost flow problems [5].

Theorem 3.10 *The minimum cost integer equal flow problem is solvable in polynomial time for any fixed number of homologous arc sets.*

Proof: Suppose the amount of the supply at node i is $b(i)$, and conversely the amount of demand at node i is $-b(i)$. Let the value of ℓ be fixed, and let y_k be the (common) amount of flow on the arcs in homologous set R_k. Our primary observation is that we can obtain the optimal amount of flow on the arcs in each of the homologous arc sets by solving the

following mixed integer program:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} & & \\
\text{s.t.} \quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b(i) & & i \in N \\
& x_{ij} = y_k & & \text{for all } (i,j) \in R_k,\ k = 1, \ldots, \ell \\
& 0 \le x_{ij} \le u_{ij} & & \text{for all } (i,j) \in A \\
& y_k \in \mathbb{Z} & & \text{for all } k = 1, \ldots, \ell
\end{aligned}
$$

Given an optimal solution (x^*, y^*) to this problem, we can obtain an integral solution with the same objective function value as follows. First, we ensure that exactly y_k^* units of flow are sent along the arcs in R_k, using the following network transformation technique: if (i, j) is an arc in set R_k, we decrease the supply at node i by y_k^*, decrease the demand at node j by y_k^*, and set the new capacity of arc (i, j) to 0. Once these transformations have been performed, the resulting problem will be a minimum cost network flow problem on the remaining arcs, which we can then solve to give an integral optimal solution. This solution will have the same cost as the original, because network flow problems are always guaranteed to possess at least one integral optimal solution.

Hence if we can solve the above mixed integer program in polynomial time, we can solve the minimum cost integer equal flow problem in polynomial time. Since ℓ is fixed, this amounts to solving a mixed integer program with a fixed number of integer variables. Lenstra [60] has shown that such problems are solvable in polynomial time. □

3.8 The Factor-α Flow Problem

The factor-α flow problem is a generalization of the equal flow problem first proposed by Larsson and Liu [59]. We are given a graph $G = (N, A)$ and disjoint sets $R_1, \ldots, R_\ell$ of arcs.

We want to find a flow such that for all $(i_1, j_1), (i_2, j_2) \in R_k$,

$$\frac{1}{\alpha} x_{i_2 j_2} \leq x_{i_1 j_1} \leq \alpha x_{i_2 j_2}$$

for some given $\alpha \geq 1$. The equal flow problem corresponds to the case when $\alpha = 1$.

The arguments employed in the previous sections can be used to establish hardness results for the integer version of this problem.

Theorem 3.11 *There is no $2^{n(1-\epsilon)}$-approximation algorithm for the maximum integer factor- α flow problem for any fixed $\epsilon > 0$ and $\alpha \geq 1$, unless P=NP.*

Proof: Let $\epsilon > 0$ and $\alpha \geq 1$ be given. We use the same reduction and construction as in the proof of Theorem 3.5. We claim:

There is an exact cover by 3-sets	$\Rightarrow$	value of max integer factor-α flow is $> 2^k$;
There is no exact cover by 3-sets	$\Rightarrow$	value of max integer factor-α flow is 1.

The first implication follows since any integer equal flow is a factor-α flow.

To see the second implication, observe that if there is no exact cover by 3-sets, then in an interal flow one of the arcs $(a_1, t), (a_2, t), \ldots, (a_q, t)$ must have zero flow. By the factor-α flow conditions, all of these arcs must then have zero flow. This means in particular that (t, t_1) has zero flow. This in turn implies that arc (s, t_1) has zero flow, which gives that (t_1, t_2) has zero flow. Extending this argument, we see all of the arcs $(t_2, t_3), \ldots, (t_{k-1}, t_k)$ have zero flow. Hence the maximum flow is 1.

By the same analysis as in Theorem 3.5, this implies there is no $2^{n(1-\epsilon)}$-approximation algorithm for the maximum factor-α flow problem, unless P=NP. □

An analogous extension of Theorem 3.7 gives the following result.

Theorem 3.12 *There is no $2^{n(1-\epsilon)}$-approximation algorithm for the maximum paired integer factor-α flow problem for any fixed $\epsilon > 0$ and $\alpha \geq 1$, unless P=NP.*

When the number of arc sets is constant, we obtain a result similar to the previous section.

Theorem 3.13 *The integer factor-α equal flow problem is solvable in polynomial time for any fixed number of homologous arc sets.*

Proof: We claim we can solve a mixed-integer program to find integral upper and lower bounds on the optimal flow, extending the argument used in Theorem 3.10. In this program, each arc set R_k is given an integral lower bound ${y_k}^{lb}$ and upper bound ${y_k}^{ub}$ on the flow, and the upper bound is restricted to being within a factor of α of the lower bound:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} & & \\
\text{s.t.} \quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b(i) & & i \in N \\
& {y_k}^{lb} \le x_{ij} \le {y_k}^{ub} & & \text{for all } (i,j) \in R_k,\ k = 1,\ldots,\ell \\
& {y_k}^{ub} \le \alpha {y_k}^{lb} & & \\
& 0 \le x_{ij} \le u_{ij} & & \text{for all } (i,j) \in A \\
& 0 \le {y_k}^{lb};\quad {y_k}^{lb}, {y_k}^{ub} \in \mathbb{Z} & & \text{for all } k = 1,\ldots,\ell
\end{aligned}
$$

This is a mixed integer program with a fixed number $(2 \cdot \ell)$ of integer variables, so by a result of Lenstra [60] it is solvable in polynomial time.

Given an optimal solution (x^*, y^*) to the mixed integer program, we claim we can obtain an integral solution with the same objective function value as follows. First, we ensure that for every set R_k, the flow on each arc (i,j) in the set satisfies $y_k^{*lb} \le x_{ij} \le \min\{u_{ij}, y_k^{*ub}\}$. This can be accomplished using a standard network transformation technique to remove lower bounds on the flow. Note since the values of y_k^{*lb}, y_k^{*ub}, and u_{ij} are all integral, the resulting problem will be a minimum cost flow problem with integral capacities and supplies. This can be solved in polynomial time to obtain an integral optimal solution.

Finally, we observe that an integral optimal solution to the mixed integer program is an optimal solution for our problem, since it is integral and respects the factor-α condition. □

We conclude by noting that these results hold even if the value of α is allowed to vary between homologous arc sets. Such problems correspond to situations in which the flow in some arc sets is restricted to be closer together than the flow in other arc sets. The key issue in the hardness arguments is that as soon as the flow on one arc in a set is forced to be zero, then the flow on all other arcs in the same set must be zero as well.

3.9 Conclusions

We have shown in Theorems 3.5 and 3.7 that the integer equal flow problem is not approximable within a factor of $2^{n(1-\epsilon)}$, for any fixed $\epsilon > 0$, even if the cardinality of each arc set is 2. This result holds not only for the maximum flow version of the problem, but also for the uncapacitated minimum cost flow version (Theorem 3.8) as well. The result was motivated by the observation (Theorem 3.3) that determining whether an instance of the problem has a nontrivial solution is NP-complete.

For the case where the number of homologous arc sets is constant, we have shown (Theorem 3.7) that the problem is solvable in polynomial time. This is due to the fact that mixed integer programs with a fixed number of integer variables can be solved efficiently. It is interesting that the gap in solvability between the constant and non-constant versions of the problem is so large. As in much of integer programming, this shows that slight changes in the definition of a problem may have a huge impact on theoretical solvability.

We have also shown how to extend the results to two related problems, the paired integer equal flow problem (Section 3.6) and the factor-α flow problem (Section 3.8). In the first case, we were able to transform integer equal flow instances into paired integer equal flow instances; in the second case, we observed that the similar structure of the factor-α problem allowed us to apply the same constructions.

We have thus addressed nearly all issues pertaining to the approximability of the integer equal flow problem. It is an exciting problem in that the relatively simple structure of the problem can nevertheless be exploited to yield very strong results. It would be interesting to see whether further variants of this problem could produce similar findings.

Chapter 4

The Pup Matching Problem

4.1 Introduction and Literature Review

Trucking is an extensive and influential industry. Just within the United States, the trucking industry generates revenues of more than $606 billion annually [1]. This suggests that even a small improvement in operations can often translate to big savings. Most traditional tractor trailers consist of a cab with a single long trailer attached. However, some cabs permit the attachment of up to two shorter trailers, known as 'pups' [16].

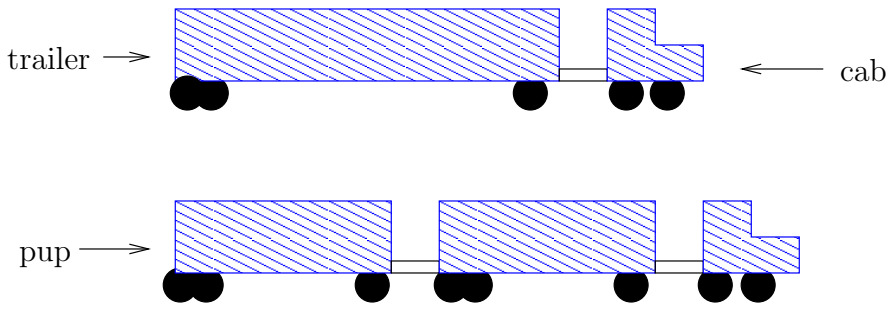

Figure 4-1: A conventional trailer and one with two tandem pups

In this situation, the cost of sending a cab along a network is the same whether it is towing one or two pups. The Pup Matching problem is the problem of pairing pups behind cabs in the most efficient manner, to minimize total travel costs.

To illustrate the problem, we consider the example below. In this problem there are two pups; we wish to send pup 1 from node s_1 to t_1 and pup 2 from node s_2 to t_2. The arc cost represents the cost of sending one cab (with one or two pups) over each link.

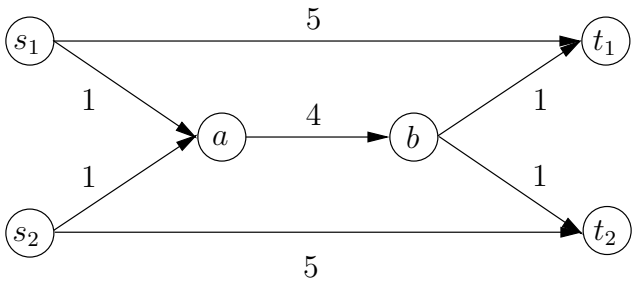

Figure 4-2: A simple pup matching example

If each cab could tow only one pup, the optimal solution would be to route both pups along their shortest paths, for a total cost of 10. However, we can do better if we take advantage of possible pup pairings. The optimal solution is to send each pup singly to node a, then have both pups share along arc (a, b), and then send each singly to its destination. The cost of this solution is 8. (Note that we assume that cabs are available at every node.)

Powell [71] characterizes the Pup Matching problem as one of the key physical operational decisions in truck routing. Along with network design, traffic assignment, and trailer management, he describes pup matching as an essential consideration in deciding how to manage flow in a trucking network. This motivates our own desire to investigate the problem and obtain stronger results.

Barnhart and Ratliff [13] consider a truck/rail intermodal trailer routing problem that can be seen as a special case of the Pup Matching problem. In this problem, the rail costs are allotted per flatcar, and each flatcar can accommodate either one or two trailers. The setting differs from the Pup Matching problem in that each origin/destination path contains at most one rail segment. Barnhart and Ratliff show that this problem can be solved in polynomial time via a weighted matching algorithm. Here, each of the pups represents a node in the matching; an edge connects two pups with its cost equal to the cost of the two pups sharing a trailer. Barnhart and Kim [12] study a class of problems they call Intra-Group

Line-Haul problems. These problems involve the determination of cyclic routes to service required pickups and deliveries at certain terminals. The cabs must be routed over circuits in the network such that each pickup trailer can advance from its origin to consolidation node and each delivery can advance from the consolidation node to its destination node. The objective is to minimize total costs, given that a trailer can carry at most two pups at a time. They present a three-step approach toward obtaining approximation bounds, which quickly produces near-optimal solutions.

Bossert [16] has conducted one of the most thorough studies of the Pup Matching problem. He proves that the problem is NP-hard, as well as observing several characteristics of an optimal solution. He further introduced four heuristic methods and a cutting plane branch-and-bound procedure for solving the problem.

Bossert's heuristics perform very well, solving tested problems to an average of within 1.3% of optimality. His branch-and-bound procedure meets with less success, due to the weakness of currently known linear programming bounds. Among the cutset inequalities that he derived, a set of "odd flow inequalities" provide the strongest results in practice. He concluded by suggesting methods for obtaining tighter bounds.

Li, McCormick, and Simchi-Levi [61] consider a related class of problems, which they call Point-to-Point Delivery problems. These problems involve shipping one item from each of p sources to p sinks. Up to K items at a time can share a truck on an arc, with costs linear in the number of trucks used. In this terminology our problem corresponds to the case when $K = 2$. The authors show that the Point-to-Point Delivery problem is NP-hard for $K \geq 2$, as is the variation of finding a minimum cost arc subset connecting sources with destinations. They describe polynomial-time algorithms for the special cases where p is fixed, or the underlying network is a grid with all sources on one side and all sinks on the other.

In what follows, we study the complexity, approximability, and solution properties of the Pup Matching problem and its variants. In Section 4.2, we formally define the problem as a network flow problem and give a simpler proof that it is NP-hard. We also review a 2-approximation algorithm for the problem due to Bossert [16]. In Section 4.3 we present two

integer programming formulations of the Pup Matching problem, a new one that we propose and one that is due to Bossert [16]. We show that the linear programming relaxation of the new formulation is stronger than that of Bossert's linear programming relaxation.

Section 4.4 further investigates the properties of the linear programming relaxation of our formulation. We show several different characteristics relating to the behavior of the linear program, and we make a conjecture as to its overall performance bound. In Section 4.5 we examine the phenomenon of *waiting rings*, which create difficulty in translating our integer programming solutions into feasible routings. Following this section, we consider two versions of the Pup Matching problem: one version in which waiting rings are allowed, and one in which they are forbidden.

In Section 4.6, we consider a variant of the Pup Matching problem that we call the K-Pup problem. This is the Pup Matching problem restricted to a fixed number K of pups. We show that in the case where waiting rings *are not* allowed, the problem is solvable in polynomial time; for the case where waiting rings *are* allowed, we propose a class of approximation algorithms. In Section 4.7, we consider another variant called the C-Problem. This is the question of whether a solution to the Pup Matching problem exists with cost less than or equal to a fixed value of C. We show that when waiting rings are not allowed, the problem is solvable in polynomial time for integer data; when waiting rings are allowed, we give algorithms for small values of C.

We define and investigate properties of Nash equilibria of the Pup Matching problem in Section 4.8. We show that the cost of a user-optimal solution is always within a factor of 2 of the optimal cost. We also explore variants of Nash equilibria and how the cost of such solutions compare to the optimal cost.

In Section 4.9, we consider the variant of the Pup Matching problem where the arcs are allowed to be capacitated. We show that this problem does not admit an approximation algorithm within a finite factor unless P=NP. We conclude in Section 4.10 by reviewing these results and their implications for future research.

4.2 Problem Definition and Complexity

An instance of the Pup Matching problem is defined as follows. We are given a directed graph $G = (N, A)$ with costs c_a for all arcs $a \in A$, and no capacities on the arcs. We also have a set $\mathcal{K} = \{1, \ldots, K\}$ of pups, and a collection of node pairs $(s_1, t_1), (s_2, t_2), \ldots, (s_K, t_K)$ such that pup i must travel from s_i to t_i. The cost of traversing arc a is the same regardless of whether 1 or 2 pups (represented as units of flow) are routed on the arc. Our objective is to route each pup on a path from its source to its sink at the minimum overall cost.

We can think of the Pup Matching problem as a version of integer multicommodity flow with a different cost structure. Specifically, two pups traversing an arc together incur the arc cost only once. We can translate this to the original trucking problem [16] by requiring that whenever two pups 'share' the cost of an arc, their corresponding trailers are paired together using one cab.

In general, we can assume that all of the pups are distinct, and that they represent different 'comomodities' in the graph. However, occasionally we consider cases where certain pups or sets of pups are indistinguishable and can be treated as one 'commodity'. We refer to the problem where all pups are indistinguishable as the Single Commodity Pup Matching problem. In this problem, it does not matter which pup sends its flow to each destination. The only requirement is that each of the source nodes s_i must have one unit of outflow, and each of the sink nodes t_i must have one unit of inflow.

We now show that the Pup Matching problem is NP-hard. Previous proofs of this result are due to Li et al. [61], who reduce from the SAT problem, and Bossert [16], who reduces from the 3-Dimensional Matching problem. Both of these proofs are quite involved and feature complicated graphical constructions. In contrast, our proof is conceptually very simple and features little graphical manipulation.

Theorem 4.1 *The Pup Matching problem is NP-hard.*

Proof: We reduce from the Arc-Disjoint Paths problem. This problem is:

Instance: A graph $G = (N, A)$ and a set of node pairs $(s_1, t_1), \ldots, (s_K, t_K)$.

Question: Does there exist a collection of arc-disjoint paths $P_1, \ldots, P_K$, where P_i is an $s_i - t_i$ path?

Suppose we are given $G = (N, A)$ and $(s_1, t_1), \ldots, (s_K, t_K)$. We transform this instance into an instance of the Pup Matching problem as follows. For every arc $(i, j) \in A$, replace (i, j) with the construction in Figure 4-3:

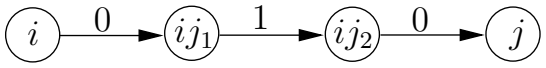

Figure 4-3: Transformation of arc (i, j) in our construction

Here we have added two new nodes and assigned costs to the arcs. Let G' be the graph resulting from these transformations. Our instance of the Pup Matching problem consists of G', along with the node pairs $(s_1, t_1), \ldots, (s_K, t_K)$ and (ij_1, ij_2) for all $(i, j) \in A$. In each of these node pairs, we wish to route one pup from the first node to the second node.

We now claim that the optimal solution to the pup matching instance has cost $\leq |A|$ if and only if there exist arc-disjoint paths in G from s_i to t_i for all i. To see the first part of this claim, suppose there exist arc-disjoint paths in G from s_i to t_i for all i. We present a routing with cost $|A|$:

> First, route the pups corresponding to demand pairs $(s_1, t_1), \ldots, (s_K, t_K)$ along the analogous arc-disjoint paths in G'. (Whenever a pup was assigned to traverse arc (i, j), it will now traverse $(i, ij_1), (ij_1, ij_2)$, and (ij_2, j) in that order.) Route these pups such that whenever pup p crosses arc (ij_1, ij_2), it shares the arc with the pup assigned to travel from ij_1 to ij_2. Finally, route all the unassigned demand from uv_1 to uv_2 singly along arc (uv_1, uv_2).

It is clear from the description of this routing that all of the arcs of cost 1 will be traversed by exactly one cab. Thus the total cost of this solution is $|A|$.

To see the other part of the claim, suppose that there is no such collection of arc-disjoint paths. Consider an optimal solution to the Pup Matching problem. Since the $s_i - t_i$

paths in this solution cannot all be disjoint, there must be two pup paths $s_{k_1} - t_{k_1}$ and $s_{k_2} - t_{k_2}$ that share some arc (ij_1, ij_2) of cost 1. Moreover, the unit of demand from ij_1 to ij_2 must also traverse this arc. Hence the total cost of pups traversing arc (ij_1, ij_2) is at least 2. Now, all other positive cost arcs (uv_1, uv_2) must be traversed at least once, since 1 unit of flow must travel from node uv_1 to node uv_2. Hence the total cost of the solution is at least $|A|+1$. Altogether, this implies that the Pup Matching problem is NP-hard. □

Bossert [16] has furthered this result to show that the Pup Matching problem is NP-hard even in the case where all the pups share a common source. As a corollary, this also gives that the Single Commodity Pup Matching problem is NP-hard.

Since the Pup Matching problem is NP-hard, in the remainder of the chapter we focus on obtaining approximation results and examining special cases of the problem. Recall from Section 2.2 that an α-approximation algorithm for a problem P is a polynomial-time algorithm that returns a feasible solution for P with cost within a factor of α of the optimum.

Bossert [16] suggested a naive 2-approximation algorithm for the Pup Matching problem, which we now review. The proof of Theorem 4.2 is our own.

Algorithm for Pup Matching

Input: A directed graph $G = (N, A)$ with sources $s_1, s_2, \ldots, s_K \in N$, sinks $t_1, t_2, \ldots, t_K \in N$, and costs on the arcs.

Output: A feasible solution to the Pup Matching problem.

1) Determine the shortest $s_i - t_i$ path for every pup i.

2) Combine these shortest paths into an overall solution and return it.

Theorem 4.2 ([16]) *The above algorithm is a 2-approximation algorithm.*

Proof: Let $c(OPT)$ be the cost of the optimal solution, and $c(ALG)$ the cost of the solution returned by the algorithm. Suppose we *double* the cost of each arc in the optimal solution. Then each arc a with cost c_a that is used $2j$ times in the optimal solution will have a new

contribution of $2jc_a$, or c_a per unit of flow. Each arc that is used $2j+1$ times will have a new contribution of $\frac{2jc_a+2}{2j+1} > c_a$ per unit of flow.

It follows that the cost of each $s_i - t_i$ path in the doubled solution will be greater than or equal to the cost of an $s_i - t_i$ path without *any* sharing. This means the cost of each $s_i - t_i$ path is greater than or equal to the shortest $s_i - t_i$ path length, ignoring the other pups.

Adding over all the pups, we get $c(ALG) \leq 2c(OPT)$, which gives the approximation bound. Finally, notice the algorithm runs in polynomial time, since finding shortest paths can be done in polynomial time. □

This bound is asymptotically tight, as is shown by the example in Figure 4-4.

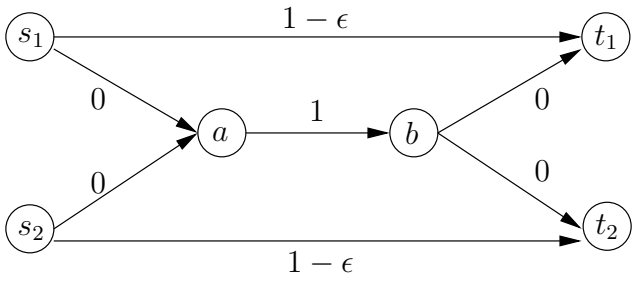

Figure 4-4: Example showing the algorithm is not better than a 2-approximation algorithm

Here the shortest path for pup 1 is the arc (s_1, t_1), and the shortest path for pup 2 is the arc (s_2, t_2), both of which give a cost of $1 - \epsilon$. Hence the algorithm returns a solution of cost $2 - 2\epsilon$. The optimal solution is for pups 1 and 2 to be routed singly to node a, then share together along arc (a, b), then proceed singly to their sinks t_1 and t_2. This has a cost of 1, so as $\epsilon \to 0$ we see that the bound is tight.

In what follows, we examine further properties of the Pup Matching problem and its variants. We begin by giving an integer programming formulation of the problem in the next section, followed by a discussion of linear programming relaxation bounds for the problem.

4.3 Integer Programming Formulations

We now present two integer programming formulations of the Pup Matching problem. The first formulation is due to Bossert [16], and the second is our own. We ultimately show that the linear programming relaxation of the new formulation is at least as strong as that of Bossert's, while in some instances it performs strictly better.

In both of the following formulations, suppose we are given a graph $G = (N, A)$, a set of pups $1, \ldots, K$, and a collection of node pairs $(s_1, t_1), \ldots, (s_K, t_K)$. The cost of arc (i, j) is denoted c_{ij}, which is the cost of sending one cab (i.e. one or two pups) across the arc.

4.3.1 Bossert's Formulation

Bossert formulates the Pup Matching problem as a special case of the Network Loading problem [16], where pups play the role of commodities and cabs play the role of facilities. He defines variables based on the amount of flow on each arc by each pup:

y_{ij}^k = amount of flow on arc (i, j) by pup k.
z_{ij} = number of cabs assigned to arc (i, j).

The formulation is:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij} z_{ij} & & \\
\text{s.t.} \quad & \sum_{j\in N} y_{s_k j}^k - \sum_{j\in N} y_{j s_k}^k = 1 & & k \in \mathcal{K} \\
& \sum_{j\in N} y_{t_k j}^k - \sum_{j\in N} y_{j t_k}^k = -1 & & k \in \mathcal{K} \\
& \sum_{j\in N} y_{ij}^k - \sum_{j\in N} y_{ji}^k = 0 & & k \in \mathcal{K},\ i \in N \setminus \{s_k, t_k\} \\
& \sum_{k\in\mathcal{K}} y_{ij}^k \le 2 z_{ij} & & (i, j) \in A \\
& z_{ij} \ge 0,\ z_{ij} \in \mathbb{Z} & & (i, j) \in A \\
& y_{ij}^k \in \{0, 1\} & & (i, j) \in A,\ k \in \mathcal{K}
\end{aligned}
$$

The first three constraints ensure that the flow is balanced with respect to the supply and demand of each pup at every node. The fourth constraint ensures that the number of cabs sent across each arc is sufficient to carry all of the pups using that arc. The pup matching condition is modeled by the objective function, which assigns cost according to the number of *cabs* (and not pups) crossing each arc. Hence this is a valid formulation of the Pup Matching problem, as we have stated it. (In Section 4.5, we will discuss practical aspects of implementing such integer programming solutions.)

4.3.2 New IP Formulation

In the following, we suggest an integer programming formulation with variables based on whether flow is *shared* or taken *singly* across an arc. Accordingly, let:

$x_{ij}^{k\ell}$ = amount of flow on arc (i,j) shared between pups k and $\ell, k \neq \ell \in \mathcal{K}$.

x_{ij}^{kk} = amount of flow on arc (i,j) routed singly along the arc by pup $k \in \mathcal{K}$.

In the following, we do not distinguish between $x_{ij}^{k\ell}$ and $x_{ij}^{\ell k}$ (they represent the same variable). With these definitions, an integer programming formulation of the problem is:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} \sum_{k,\ell \in \mathcal{K}} c_{ij} x_{ij}^{k\ell} & & \\
\text{s.t.} \quad & \sum_{j\in N, \ell \in \mathcal{K}} x_{s_k j}^{k\ell} - \sum_{j\in N, \ell \in \mathcal{K}} x_{j s_k}^{k\ell} = 1 & & k \in \mathcal{K} \\
& \sum_{j\in N, \ell \in \mathcal{K}} x_{t_k j}^{k\ell} - \sum_{j\in N, \ell \in \mathcal{K}} x_{j t_k}^{k\ell} = -1 & & k \in \mathcal{K} \\
& \sum_{j\in N, \ell \in \mathcal{K}} x_{ij}^{k\ell} - \sum_{j\in N, \ell \in \mathcal{K}} x_{ji}^{k\ell} = 0 & & k \in \mathcal{K},\ i \in N \setminus \{s_k, t_k\} \\
& x_{ij}^{k\ell} \in \{0,1\} & & (i,j) \in A,\ k,\ell \in \mathcal{K}
\end{aligned}
$$

The first three sets of constraints ensure that the flow is balanced at every node with respect to the supply and demand of each pup. The last set of constraints requires all flow values to be 0 or 1. Our choice of variables ensures that the pup matching conditions are satisfied; when pups k and ℓ share an arc, its cost is contributed only once. Hence this is another valid formulation of the problem.

4.3.3 Comparison of IP Formulations

The *linear programming relaxation* of each of the formulations is obtained by eliminating the integrality constraints on the variables.

Let LP_B be the value of an optimal solution to the LP relaxation of Bossert's IP.
Let LP_{new} be the value of an optimal solution to the LP relaxation of the new IP.

Theorem 4.3 *For all instances, $LP_B \leq LP_{new}$; there are some instances with $LP_B < LP_{new}$.*

Proof: We first show that we can convert any feasible solution to the new linear programming relaxation into a feasible solution to Bossert's linear programming relaxation with equal cost. Accordingly, suppose that x is a feasible solution to the new LP relaxation, of cost $c(x)$. Define a solution of Bossert's LP relaxation by setting:

$$y_{ij}^k = \sum_{\ell \in \mathcal{K}} x_{ij}^{k\ell} \qquad (i,j) \in A, k \in K,$$

$$z_{ij} = \sum_{k \leq \ell \in \mathcal{K}} x_{ij}^{k\ell} \qquad (i,j) \in A.$$

We can verify that (y, z) is feasible in Bossert's LP using the constraints from the new LP; for instance, $\sum_{k \in K} y_{ij}^k = \sum_{k,\ell \in \mathcal{K}} x_{ij}^{k\ell} \leq 2 \cdot \sum_{k \leq \ell \in \mathcal{K}} x_{ij}^{k\ell} = 2z_{ij}$. The cost of (y, z) is

$$c(y,z) = \sum_{(i,j) \in A} c_{ij} z_{ij} = \sum_{(i,j) \in A} \sum_{k,\ell \in \mathcal{K}} c_{ij} x_{ij}^{k\ell} = c(x),$$

so the two solutions have the same cost. Hence $LP_B \leq LP_{new}$.

To see that the new LP relaxation might perform strictly better than Bossert's LP relaxation, consider the simple example:

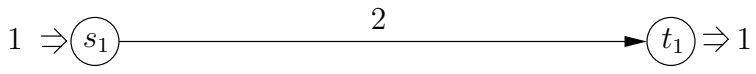

Figure 4-5: Example in which the new relaxation performs better than Bossert's relaxation

Here there is one source, one sink, and one unit of flow. Bossert's LP relaxation sets $y^1_{s_1t_1} = 1$ and $z_{s_1t_1} = \frac{1}{2}$, for a total cost of 1. The new LP relaxation sets $x^{11}_{s_1t_1} = 1$, for a total cost of 2. Hence in this instance, $LP_B < LP_{new}$. $\square$

This implies that the new formulation of the problem provides a tighter linear programming relaxation than Bossert's. Moreover, the performance gap between optimal solutions to the Bossert LP relaxation and the IP can be as large as 2; this is as large as the performance ratio for the shortest path approximation algorithm discussed in the last section (which follows, since his relaxation corresponds to a shortest path relaxation). Hence Bossert's LP relaxation is unlikely to be useful in obtaining improved approximation bounds. We instead turn our attention toward investigating the properties of the new LP relaxation, which we cover in the next section.

As a final note, observe that solutions to both integer programs correspond to flows satisfying the pup matching condition. Interestingly, such flows *may not* always correspond to a feasible routing over *time*. (When we refer to a routing, we mean an explicit sequencing of pup transitions.) We discuss this further in Section 4.5 on waiting rings.

4.4 LP Relaxation of the New Integer Program

The LP relaxation of the new pup matching formulation is:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} \sum_{k,\ell\in\mathcal{K}} c_{ij} x^{k\ell}_{ij} & & \\
\text{s.t.} \quad & \sum_{j\in N,\ell\in\mathcal{K}} x^{k\ell}_{s_kj} - \sum_{j\in N,\ell\in\mathcal{K}} x^{k\ell}_{js_k} = 1 & & k \in \mathcal{K} \\
& \sum_{j\in N,\ell\in\mathcal{K}} x^{k\ell}_{t_kj} - \sum_{j\in N,\ell\in\mathcal{K}} x^{k\ell}_{jt_k} = -1 & & k \in \mathcal{K} \\
& \sum_{j\in N,\ell\in\mathcal{K}} x^{k\ell}_{ij} - \sum_{j\in N,\ell\in\mathcal{K}} x^{k\ell}_{ji} = 0 & & k \in \mathcal{K},\ i \in N \setminus \{s_k, t_k\} \\
& 0 \le x^{k\ell}_{ij} \le 1 & & (i,j) \in A,\ k,\ell \in \mathcal{K}
\end{aligned}
$$

We now discuss properties of this relaxation, offering observations and results. In particular, we investigate the gap between the cost of optimal solutions to the IP and the LP relaxation.

4.4.1 Properties of the LP Relaxation

The LP relaxation might understimate the cost on some of the arcs as compared to the IP solution. This happens when there are multiple ways to partition the sharing on an arc. The most the LP can underestimate the cost on a single arc is by a factor of $\frac{4}{3}$, as in the following example:

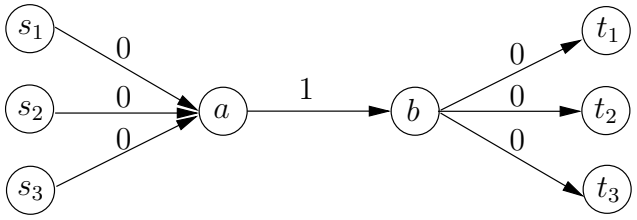

Figure 4-6: Example in which the LP relaxation underestimates the cost of an arc by $\frac{4}{3}$

Here an optimal IP solution is for pups 1 and 2 to share arc (a, b), with pup 3 routed singly, for a cost of 2. The optimal LP relaxation solution assigns $x_{ab}^{12} = x_{ab}^{13} = x_{ab}^{23} = \frac{1}{2}$, which gives an optimal cost of $\frac{3}{2}$. Hence the ratio between the IP and LP solutions is $\frac{4}{3}$. (Observe that for more than three pups traveling on an arc, the ratio can get only better.)

We now examine the structure of optimal solutions to the LP. Let

$$x_{ij}^{k} = \sum_{\ell \in \mathcal{K}} x_{ij}^{k\ell}.$$

We say a solution satisfies the *one path for every pup property* if $x_{ij}^{k} \in \{0, 1\}$ for all i, j, k. This states that the routing used by pup k from s_k to t_k in the LP solution is a single path. In other words, the solution consists of one $s_k - t_k$ path for every pup, giving K (possibly intersecting) paths overall. With this in mind we obtain the following result.

Theorem 4.4 *If the optimal LP relaxation solution satisfies the one path for every pup property, then the optimal IP solution is within a factor of $\frac{4}{3}$ of the optimal LP solution.*

Proof: Let $c(OPT)$ be the cost of an optimal pup matching solution, and let x be an optimal solution to the LP relaxation that is in terms of one path for every pup. As stated before, the LP might underestimate the cost on certain arcs. Define $\tilde{x}$ to be the following solution:

> In $\tilde{x}$, each pup k takes the same $s_k - t_k$ path assigned to it in x. Whenever there are multiple pups assigned to cross an arc, the pups are paired together one-to-one, in such a way that the minimum number of cabs is utilized.

By previous observation, we have

$$c(\tilde{x}) \leq \frac{4}{3}c(x),$$

since $\tilde{x}$ uses the same arcs as x.

We claim that $\tilde{x}$ is feasible for the IP. To see this, observe that since x is in terms of one path for every pup, each of the pups will send exactly one unit of flow over the arcs they utilize. Hence when we pair the pups together as described, the result will be an integral solution. This shows that

$$c(OPT) \leq c(\tilde{x}),$$

which gives

$$c(OPT) \leq \frac{4}{3}c(x)$$

as desired. □

This motivates the following algorithm for the Pup Matching problem:

Algorithm for Pup Matching

Input: A directed graph $G = (N, A)$ with sources $s_1, s_2, \ldots, s_K \in N$, sinks $t_1, t_2, \ldots, t_K \in N$, and costs on the arcs.

Output: A feasible solution for the Pup Matching problem.

1) Compute an optimal solution x to the LP relaxation.

2) If x satisfies the one path for every pup property, define $\tilde{x}$ as above. Else, set $\tilde{x} = x$.

3) Return $\tilde{x}$.

Corollary 4.5 *If the optimal LP relaxation solution satisfies the one path for every pup property, then the Algorithm for Pup Matching is a $\frac{4}{3}$-approximation algorithm.*

Proof: Let x be the solution returned by the LP relaxation, $\tilde{x}$ the solution returned by the algorithm, and $c(OPT)$ the cost of an optimal IP solution. By the previous theorem,

$$c(OPT) \leq c(\tilde{x}) \leq \frac{4}{3}c(x) \leq \frac{4}{3}c(OPT),$$

giving the performance bound. Finally, notice the algorithm runs in polynomial time since linear programs are solvable in polynomial time. □

Interestingly, even if the optimal LP relaxation solution satisfies the one path for every pup property, the optimal IP solution might consist of *different* paths for some of the pups. This is illustrated in the example in Figure 4-7.

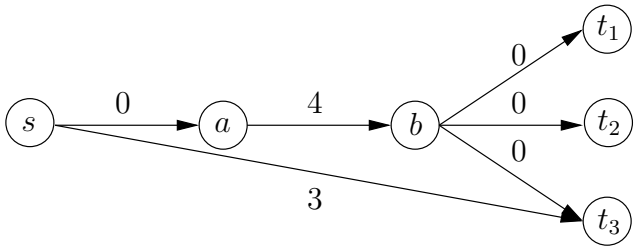

Figure 4-7: Example in which the IP solution consists of different paths

In this example there is a single source and three pups. The optimal LP solution sends pup i along the $s - a - b - t_i$ path, for all i. It assigns $x_{ab}^{12} = x_{ab}^{13} = x_{ab}^{23} = \frac{1}{2}$, for a total cost of 6:

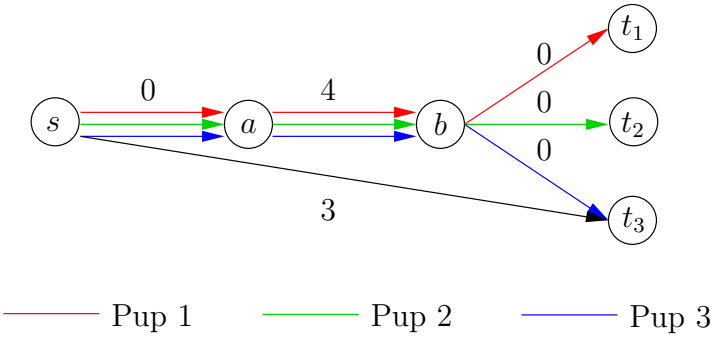

Figure 4-8: Optimal LP relaxation solution for the example

If we input the paths returned by the LP solution into the IP, we obtain a cost of 8. However, this is not optimal. The optimal IP solution is to send pups 1 and 2 along their previous paths and pup 3 along the $s - t_3$ path, for a total cost of 7:

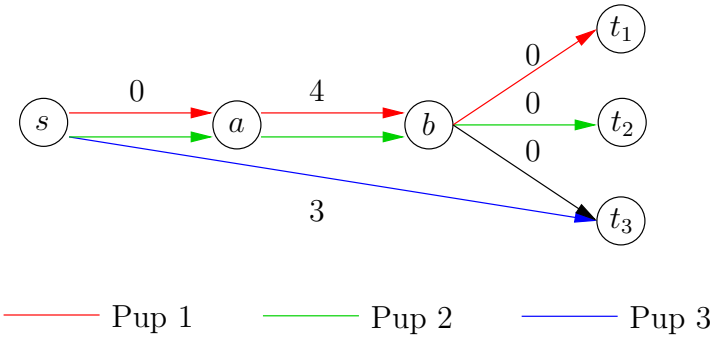

Figure 4-9: Optimal IP solution for the example

Hence even though the optimal solution to the LP relaxation is in terms of one path for every pup, the optimal IP solution contains a different path for one of the pups. (In this case, the solution returned by the LP relaxation is also not an optimal solution for the IP.)

4.4.2 Fractional LP Relaxation Solutions

We now turn our attention toward the composition of optimal solutions to the LP relaxation. Ideally, we would like all optimal solutions to the LP relaxation to satisfy the one path for every pup property; however, this situation does not always occur, as is illustrated in the following lemma.

Lemma 4.6 *It is possible for the LP relaxation to return an optimal solution that contains fractional path solutions for some of the pups.*

Proof: Consider the example in Figure 4-10. In the example, there are two pups, with a common source and sinks as shown. Each of the pups has three possible source-sink paths; one option is the $s - a_i - t_i$ path, and the other options are the $s - a_1 - a_3 - a_4 - t_i$ and $s - a_2 - a_3 - a_4 - t_i$ paths.

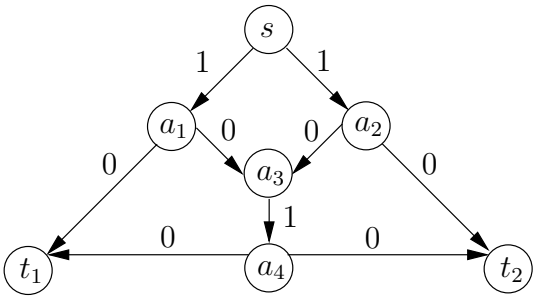

Figure 4-10: Example in which the optimal LP relaxation solution is fractional

The optimal solution to the LP relaxation splits flow from pup 1 along paths $s - a_1 - t_1$ and $s - a_2 - a_3 - a_4 - t_1$, and flow from pup 2 on paths $s - a_2 - t_2$ and $s - a_1 - a_3 - a_4 - t_2$:

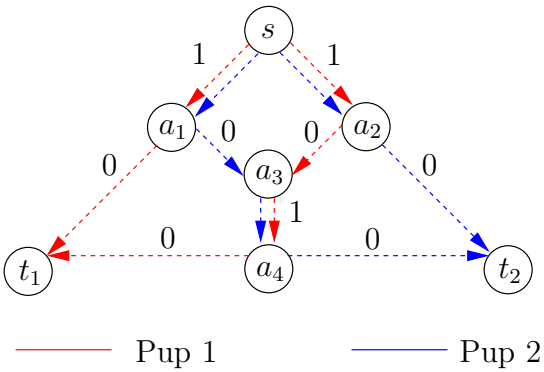

Figure 4-11: Optimal solution containing fractional paths in the LP relaxation

Each of the dashed lines indicates $\frac{1}{2}$ unit of flow on that arc. The total cost of this solution is $\frac{3}{2}$. The optimal IP solution sends pup 1 on path $s - a_1 - t_1$ and pup 2 on path $s - a_2 - t_2$.

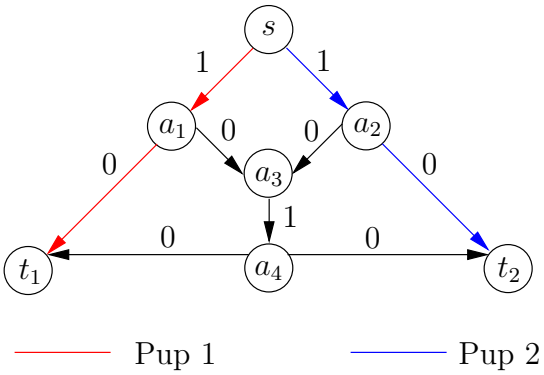

Figure 4-12: Optimal IP solution for the instance

The cost of this IP solution is 2. Thus here the optimal solution to the LP relaxation is fractional, and the ratio between the optimal fractional and integral solutions is $\frac{4}{3}$. □

We can extend this example to obtain the following result.

Lemma 4.7 *It is possible for the optimal LP relaxation solution to contain paths with arbitrarily small flow.*

Proof: Consider the example in Figure 4-13.

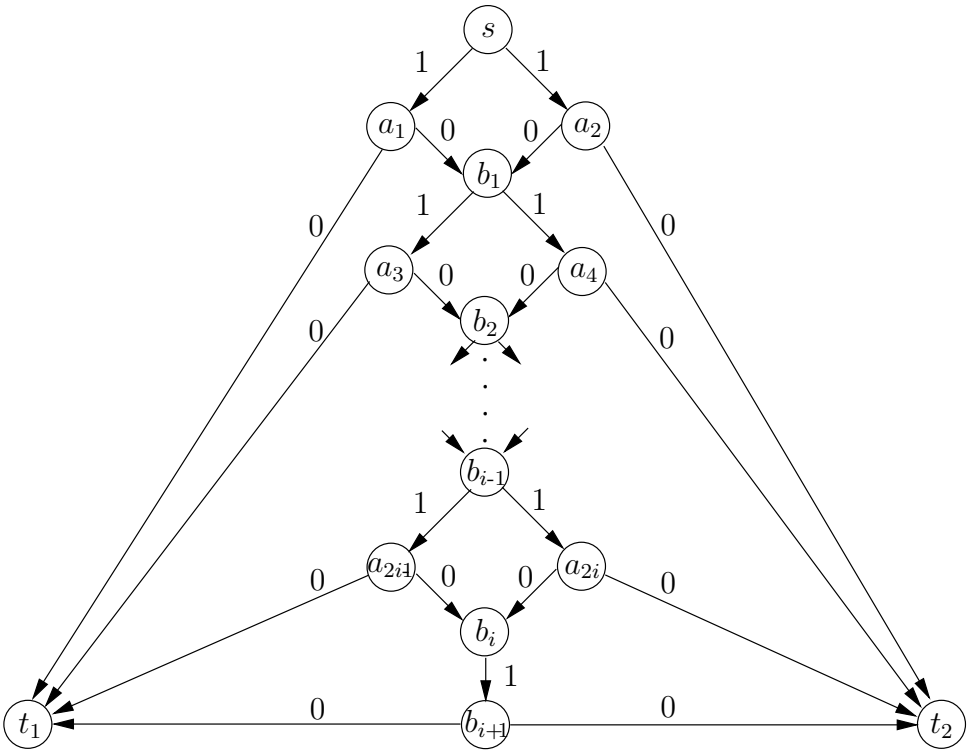

Figure 4-13: Example in which the LP relaxation solution contains arbitrarily small flow

This is similar to the previous example, but with some new nodes added. In the optimal LP relaxation solution for this example, pup 1 sends $\frac{1}{2}$ unit of flow along $s - a_1 - t_2$ and pup 2 sends $\frac{1}{2}$ unit of flow along $s - a_2 - t_2$, as before. In addition, pup 1 sends $\frac{1}{4}$ unit of flow along $s - a_2 - b_1 - a_3 - t_1$, and pup 2 sends $\frac{1}{4}$ unit of flow along $s - a_1 - b_1 - a_4 - t_2$. This pattern continues, so that pup 1 sends $\frac{1}{2^i}$ units of flow along $s - a_2 - b_1 - a_4 - \cdots - b_{i-1} - a_{2i-1} - t_1$

and pup 2 sends $\frac{1}{2^i}$ units of flow along $s - a_1 - b_1 - a_3 - \cdots - b_{i-1} - a_{2i} - t_2$. We can see that this solution is optimal by induction on the number of nodes b_i, starting with the example from Lemma 4.6. Also, as we let $i \to \infty$, this example contains paths with arbitrarily small fractional flow. □

In general, fractional solutions occur when two (or more) pups have multiple good choices for a source-sink path. Some of these choices are only attractive if the pups share a certain portion of the arcs. Each pup wishes to minimize its own cost, and sometimes this objective is best accomplished by sharing only a fraction of the flow along some path.

In the first example (Figure 4-10), pup 1 had the possible paths $s - a_1 - t_1$ and $s - a_2 - a_3 - a_4 - t_1$, and pup 2 had the possible paths $s - a_2 - t_2$ and $s - a_1 - a_3 - a_4 - t_2$ (among others). If either pup chose to send 1 unit of flow along one of these possible paths, the best possible solution would have had cost 2. However, by 'cooperating' and sending $\frac{1}{2}$ unit of flow along each of these paths, each of the positive cost arcs was only used $\frac{1}{2}$ times, for a solution of cost $\frac{3}{2}$.

4.4.3 Conjecture

Our main outstanding question with regard to the LP relaxation is the following conjecture:

Conjecture 4.8 *The cost of the optimal solution to the LP relaxation is always within a factor of $\frac{4}{3}$ of the cost of the optimal IP solution.*

We have seen that this conjecture is proven to be true if an optimal LP solution is in terms of one path for every pup; however, the optimal LP solution is not always of this form. We conjecture that even when the optimal LP solution is not of this form, this solution can be modified to give an integral solution of cost within a factor of $\frac{4}{3}$ of optimal.

We have several examples showing that the bound of $\frac{4}{3}$ can be achieved; none of these can obviously be modified to produce a greater gap. If such an example were to exist, it would need to both underestimate arc costs (as in the Example 4-6) and contain fractional paths. We believe that this cannot happen because whenever a pup chooses to send its flow

along fractional paths, it is because the flow will be shared with some other pup along the path. Conversely, whenever the arc costs are underestimated, it is because some portion of flow is *not* shared along a path. Since these are somewhat conflicting phenomena, we believe the bound of $\frac{4}{3}$ is maintained by both.

We finally note that this conjecture does not hold in the Single Commodity Pup Matching problem.

Theorem 4.9 *In the Single Commodity Pup Matching problem, the LP relaxation gap can be as large as 2.*

Proof: Consider the following example:

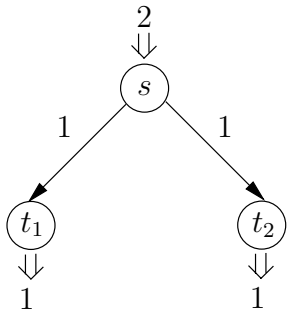

Figure 4-14: Example where the single commodity LP relaxation gap is equal to 2

An optimal solution to the LP relaxation sends $\frac{1}{2}$ unit of flow from pup 1 and pup 2 along each of the arcs (s, t_1) and (s, t_2).

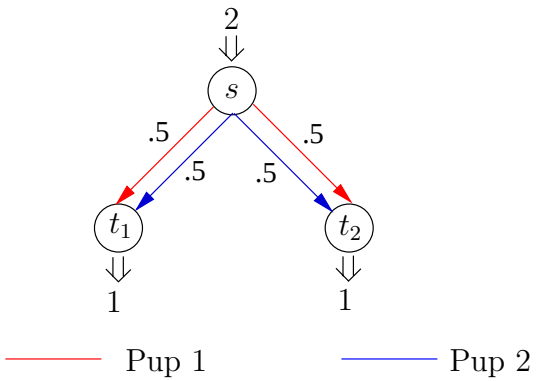

Figure 4-15: Optimal solution for the single commodity example

This solution has cost 1. The optimal IP solution is to send 1 pup to each of t_1 and t_2, which has cost 2. Hence the LP can be off by a factor of 2. The nonfixed destinations allow us to 'split' the flow entering a given sink, so that instead of sending 1 unit of flow alone along an arc, we can send .5 units of flow shared between the two pups. □

In the next section, we shift our focus from the performance of the LP relaxation to properties of integer programming solutions. Specifically, we address whether (static) solutions to the new IP can be translated to (dynamic) solutions *over time*.

4.5 Waiting Rings

4.5.1 Introduction

One issue with the new integer programming formulation is that it assumes that two pups matched to the same arc can always share a single cab. More specifically, the IP solutions represent pup paths; these paths might or might not correspond to feasible routings over *time* with the same cost. As a consequence, an IP solution can have many feasible corresponding routings, or occasionally none at all. This last case is illustrated in Figure 4-16.

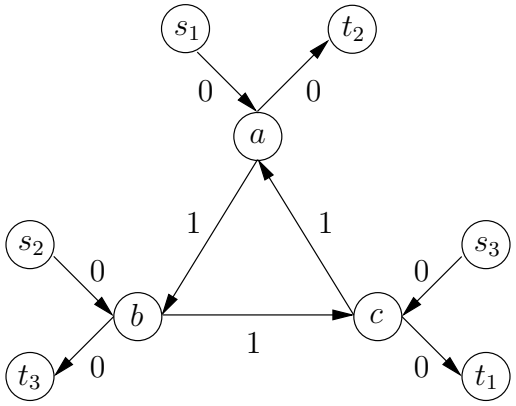

Figure 4-16: Example that gives an IP solution with no corresponding feasible routings

Here there are three pups with sources and sinks as shown, and one unit of flow is to be sent from each source to each sink. The optimal IP solution has cost 3, and it transports each pup along its unique source-sink path as shown in Figure 4-17.

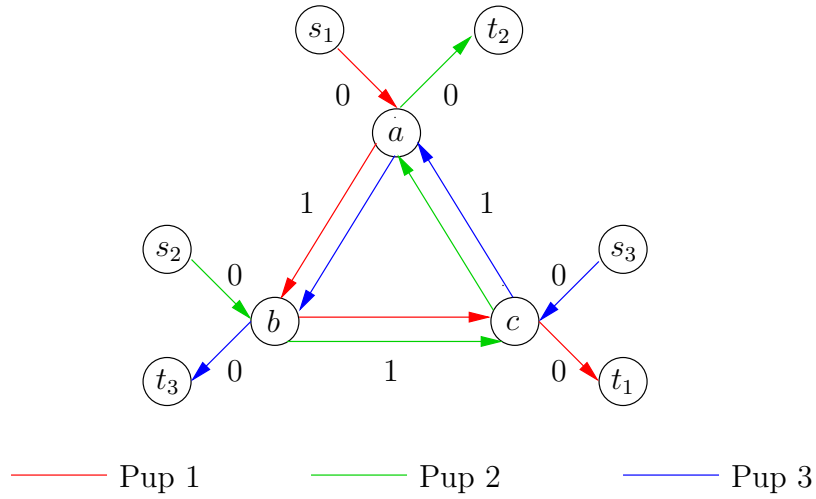

Figure 4-17: Optimal solution to the integer program with no corresponding feasible routing

Unfortunately, this solution does not have a corresponding feasible routing with the same cost. This is since to traverse arc (a, b), pup 1 must share with pup 3; thus first pup 3 must be sent along arc (c, a). However, pup 3 must share with pup 2 on (c, a). This means pup 2 must be first sent along arc (b, c). To be sent along (b, c), pup must share with pup 1; this is impossible since pup 1 has not yet traversed (a, b). This implies that any feasible routing using this solution must have greater cost.

The optimal feasible routing to this problem has cost 4: first, pup 1 is sent singly along (a, b), so that the sharing between pups 1 and 3 on (a, b) is eliminated. Then pup 1 shares with pup 2 on (b, c), and pup 2 shares with pup 3 on (c, a). Finally, pup 3 is sent singly along (a, b). Each of the pups are then routed to their respective sinks.

We formally define this phenomenon in the next subsection.

4.5.2 Definition

Suppose we have a collection of pups $P_1, P_2, \ldots, P_\ell$ with the following property:

> Each pup P_i in the collection has arrived at some node but it cannot move along its assigned path until its assigned pair P_j from the collection has arrived;

likewise, pup P_j is stuck at a different node waiting for its partner P_k to traverse their common path.

The collection is closed in that every pup in the collection is waiting for another pup in the collection that is similarly unable to advance.

The simple case of this phenomenon occurs when the pups waiting form a single 'ring'; that is, pup P_i is waiting for pup P_{i-1} for every i, and pup P_1 is waiting for pup P_ℓ. (This is the case in the previous example.) We call a *waiting ring* the portion of the graph between the nodes where the pups are stuck and the nodes where they would complete travel with the pups that are waited on. (In the previous example, the cycle of arcs $a - b - c - a$ is a waiting ring.)

The waiting ring phenomenon can also occur when there are only 2 pups, as in the example in Figure 4-18:

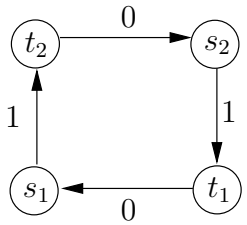

Figure 4-18: Example that produces a waiting ring with only two pups

The optimal IP solution sends pup 1 on path $s_1 - t_2 - s_2 - t_1$ and pup 2 on path $s_2 - t_1 - s_1 - t_2$:

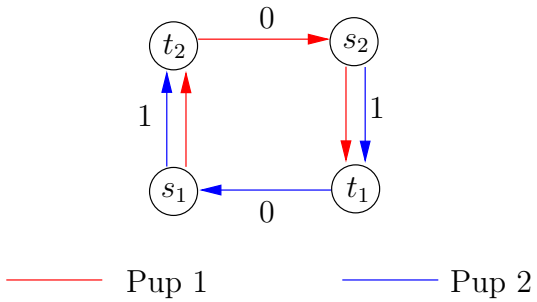

Figure 4-19: Optimal solution to the integer program for the two pup problem

Again, this solution does not correspond to a feasible routing with the same cost. Pup 1 is stuck at s_1 waiting for pup 2 to traverse arc (s_1, t_2); pup 2 is stuck at s_2 waiting for pup 1

to traverse arc (s_2, t_1). Moreover, by layering instances of examples like the one above, we can come up with even more complex scenarios:

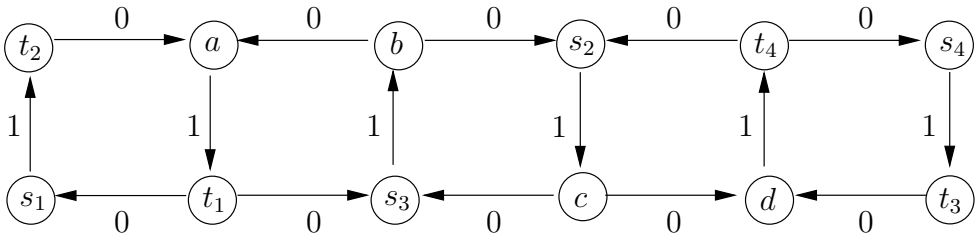

Figure 4-20: More complicated example that produces an unusual waiting ring

The optimal IP solution sends each pup along its unique $s_i - t_i$ path:

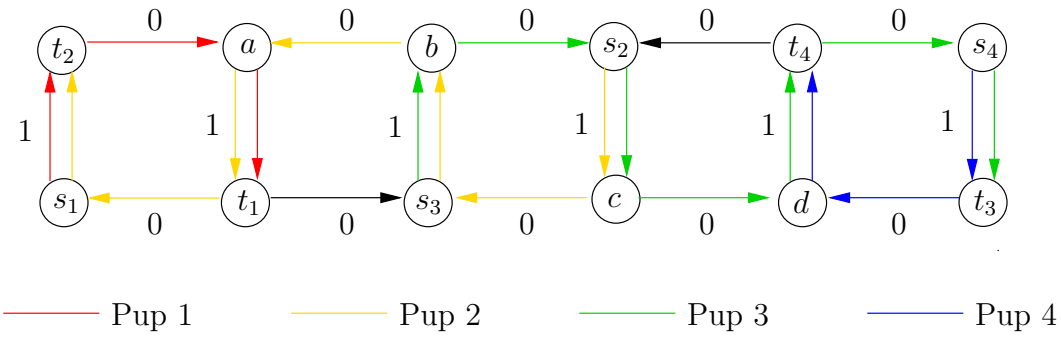

Figure 4-21: Optimal solution to the integer program that produces a waiting ring

Here, pup 1 waits for pup 2 at node s_1; pup 2 waits for pup 3 at node s_2; pup 3 waits for pup 2 at node s_3; pup 4 waits for pup 3 at node s_4. Note that the locations of these pups do not form a 'ring' as in the previous examples, but the example still satisfies the definition of a waiting ring.

Waiting rings are similar to the phenomenon of *deadlocks* in databases. Deadlocks occur when requests from separate tasks for resources are granted in such a way that a group of two or more tasks is unable to advance, because each task is simultaneously monopolizing its own resources and waiting for the release of resources held by other group members. There are four conditions that characterize deadlocks (see [22]), which are *monopolization* of resources by the tasks holding them, a set of tasks holding their own resources while *waiting for* others, *no preemption* of the resources until their task has completed, and a *circular*

chain of tasks such that each task holds one or more resources requested by the next task in the chain. (For further details on the composition of deadlocks, see [22, 30].)

We can view waiting rings as a variant of deadlocks in a transportation setting, where the routings are the tasks to be completed and the pups are the resources needed. Instead of monopolization of the resources, our problem arises because resources must be *shared* between tasks. When a resource is not available to be shared, the task waits for it to become available; if this happens in a circular fashion, we arrive at a waiting ring situation.

It should be noted that there is no general method of avoiding deadlocks in practice. Most solution techniques (see [30]) involve algorithms that are specifically designed to ignore one of the four deadlock conditions. However, the effectiveness of these techniques varies greatly depending on the difficulty of the problem in consideration. In the Pup Matching problem, even detecting whether a solution contains a waiting ring is NP-hard [16], so we need more problem-specific measures to address this phenomenon. In Section 4.5.4 we propose and discuss some of these measures.

4.5.3 Comparison of Solutions with and without Waiting Rings

In this section, we wish to determine how the presence or absence of waiting rings can affect the cost of an optimal solution to an instance of the Pup Matching problem.

Consider an arbitrary instance of the Pup Matching problem. Let WR be an optimal solution with cost $c(WR)$ to the integer program (which implicitly permits waiting rings), and NWR an optimal feasible routing of cost $c(NWR)$. We obtain the following result:

Theorem 4.10 *The ratio* $\dfrac{c(NWR)}{c(WR)} \leq 2$*, and there exist examples where the ratio can be made arbitrarily close to 2.*

Proof: To see that the ratio is less than or equal to 2, observe that one way to eliminate waiting rings in the solution WR is to route each of the pups singly along the path assigned by the integer program. This gives a feasible routing with cost at most $2c(WR)$.

We now introduce a class of examples where the ratio can be made arbitrarily close to 2. Recall the two-pup waiting ring example from the previous section:

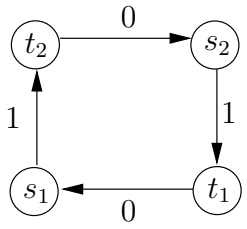

Figure 4-22: Two pup waiting ring example

Extend this example by introducing another 'ring' to the side, of opposite orientation, as in Figure 4-23.

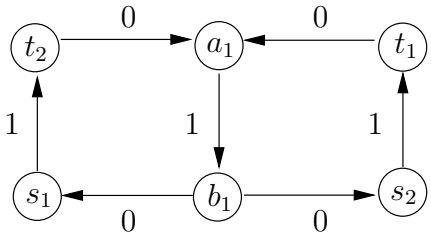

Figure 4-23: Extension of the two pup waiting ring example

The optimal IP solution has cost 3, routing each pup along its $s_i - t_i$ path:

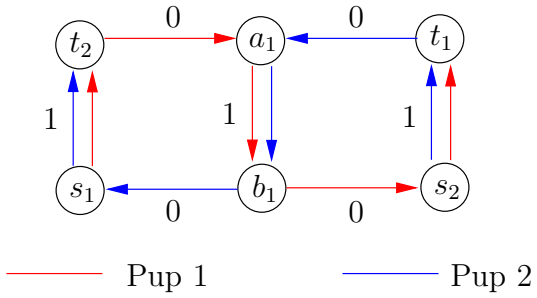

Figure 4-24: Optimal solution to the integer program for the extended waiting ring example

This solution is unique, since the source-sink paths for each pup are unique. Moreover, in any feasible routing, at most one of the arcs $(s_1, t_2), (a_1, b_1)$, and (s_2, t_1) may be shared between pups 1 and 2; this is since the pup paths are oriented in opposite directions. Hence the optimal feasible routing has cost 5, and the ratio $\frac{c(NWR)}{c(WR)} = \frac{5}{3}$.

We can further extend this example by introducing i more rings to the side (without loss of generality, suppose i is even).

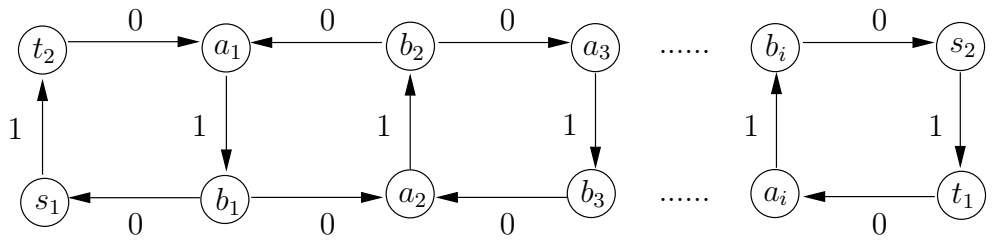

Figure 4-25: Extension of the waiting ring example by adding i rings to the side

The optimal solution here has cost $i + 2$, routing both pups on their unique $s_i - t_i$ paths:

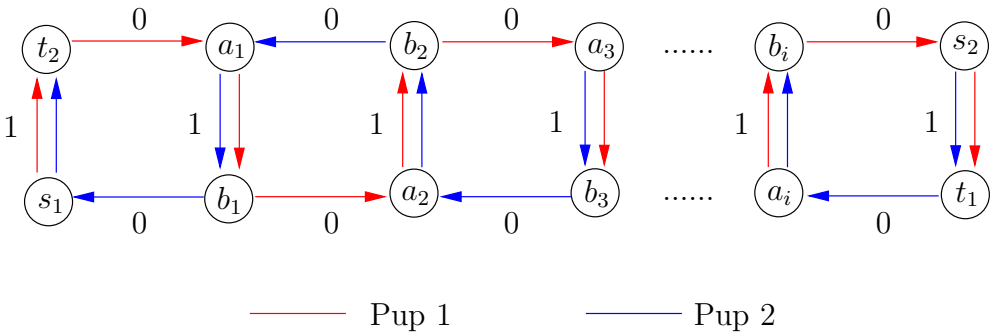

Figure 4-26: Optimal solution to the extension adding rings to the side

Again, in any feasible routing at most one of the arcs $(s_1, t_2), (s_2, t_1)$, and (a_j, b_j) for any j may be shared between pups 1 and 2. By inspection, we see this means the optimal feasible routing has cost $2i + 3$. The ratio is

$$\frac{c(NWR)}{c(WR)} = \frac{2i+3}{i+2},$$

which can be made arbitrarily close to 2. □

4.5.4 Eliminating Waiting Rings

We now address the subject of eliminating waiting rings. We first show that waiting rings in the Single Commodity Pup Matching problem can be eliminated without an increase in cost, and then we give an integer program for the Pup Matching problem that produces solutions without waiting rings.

Theorem 4.11 *Waiting rings arising in the Single Commodity Pup Matching problem can be eliminated without an increase in cost.*

Proof: For solutions that contain a single waiting ring (as in Figures 4-16 and 4-18 for multiple commodities), use the following procedure:

- First, route pup i from its origin up until the node where it becomes immobilized in the waiting ring pending the arrival of pup $i-1$.
- Next, route pup i singly along the set of arcs in the waiting ring that were assigned to be shared by pups i and $i-1$.
- From the current node on, route pup i to destination t_{i-1} following the path previously assigned to be used by pup $i-1$ from the current node to node t_{i-1}.

This reassignment breaks the ring, since each pup now has a realizable routing from its (original) origin to its (newly assigned) destination; the shared arcs that previously constituted the waiting ring are now taken singly. No new waiting rings are created, since the portion of the solution lying outside the original waiting ring is unchanged.

For solutions containing multiple layered waiting rings, as in Figure 4-20, proceed through each of the simple underlying waiting rings within the example and perform the procedure above. Once a simple waiting ring has been broken, the arcs involved in that ring will be assigned to be taken singly. Moreover, the portion of the solution outside the ring will be unchanged. Hence, no new waiting rings will be introduced and one segment of the congestion will have been eliminated.

Once this procedure has been performed on all simple waiting rings, we claim we will have a feasible routing. This is since every pup involved will now have a realizable routing from its origin to its newly assigned destination, and no new waiting rings will have been created. Finally, note the cost of the solution will not increase since each arc in the new solution is traversed the same number of times (or possibly fewer) than in the previous solution. □

For the general (multicommodity) Pup Matching problem, we eliminate waiting rings by explicitly formulating the problem over time. To do so, we introduce a time variable t that indexes the order in which node transitions occur and that is independent of the length of the arcs traversed. We divide this time into discretized periods starting at $t = 1$ and ending at $t = nK$, where K is the total number of pups.

Our variables are:

$x_{ij}^{k\ell}(t)$ = amount of flow on arc (i, j) shared by pups k and ℓ at time t.

We propose the following integer program:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} \sum_{k,\ell\in\mathcal{K}} \sum_{t=1}^{nK} c_{ij} x_{ij}^{k\ell}(t) & \\
\text{s.t.} \quad & x_{ij}^{k}(t) = \sum_{\ell\in\mathcal{K}} x_{ij}^{k\ell}(t) & (i,j) \in A,\ k,\ell \in \mathcal{K},\ t = 1,\ldots,nK \\
& \sum_{j\in N} \sum_{t=1}^{T-1} x_{ji}^{k}(t) \geq \sum_{j\in N} \sum_{t=1}^{T} x_{ij}^{k}(t) & i \in N \setminus \{s_k, t_k\}, k \in \mathcal{K}, T = 1,\ldots,nK \\
& \sum_{j\in N} \sum_{t=1}^{nK} x_{s_k j}^{k}(t) = 1 & k \in \mathcal{K} \\
& \sum_{j\in N} \sum_{t=1}^{nK} x_{j t_k}^{k}(t) = 1 & k \in \mathcal{K} \\
& x_{ij}^{k\ell}(t) \in \{0,1\} & (i,j) \in A,\ k,\ell \in \mathcal{K},\ t = 1,\ldots,nK
\end{aligned}
$$

Theorem 4.12 *The preceding integer program gives an optimal solution to the Pup Matching problem that does not contain waiting rings.*

Proof: Note that if there is a feasible solution to the original pup matching integer program that does not contain a waiting ring, we can translate it to the above IP by requiring that at least one pup is moved along one arc at every time step. Because each pup might visit at most n nodes in its path from source to sink, this gives a maximum number of nK time steps.

Also, observe that any solution to the original pup matching IP that contains a waiting ring will not be feasible in the new waiting ring IP. This is since in a waiting ring solution, there is no way to assign times to node transitions to achieve a routing of the same cost. Hence these solutions will be eliminated.

Finally, we claim that the IP accurately models multicommodity flow. Given an IP solution, we claim we can construct a path from s_k to t_k for every pup k in $\mathcal{K}$. To do so, start with node t_k. Since

$$\sum_{j \in N} \sum_{t=1}^{nK} x^k_{jt_k}(t) = 1,$$

there exists some time index T_1 and a node j_1 such that $x^k_{j_1 t_k}(T_1) = 1$. Using the first set of constraints, this implies

$$\sum_{j \in N} \sum_{t=1}^{T_1 - 1} x^k_{jj_1}(t) \geq 1,$$

which suggests that there exists some index $T_2 < T_1$ and a node j_2 such that $x^k_{j_2 j_1} = 1$. Reapplying the first set of constraints, we see that there must be inflow from some node j_3 to j_2 at a time $T_3 < T_2$. Continuing in this manner, we can trace a path all the way back to the sink s_k. (Note that we will not "get stuck" at an intermediate node j, since the first shipment of pup k must come from node s_k.) Flow along this $s_k - t_k$ path will proceed in chronological order, by the way the times T_i are defined. Hence the IP models multicommodity flow and produces a solution to the Pup Matching problem that does not contain waiting rings. □

Thus in this section we have identified the waiting ring phenomenon and analyzed its worse case performance. We have also found a way to ensure that waiting rings do not occur, which is of great practical interest.

In what follows, we examine some interesting special cases and variants of the Pup Matching problem. Whenever possible, we consider both versions of the problem with and

without waiting rings allowed. Often we find that solving these two similar problems requires two very different approaches. This is both surprising and unintuitive, and it illustrates the subtleties of the problem.

4.6 The K-Pup Problem

We define the *K-Pup problem* as the Pup Matching problem with a fixed number K of pups to send through the network. This arises in real life when there is a fixed amount of goods to ship between specified sources and destinations. We distinguish between versions of the problem when waiting rings are allowed and when they are forbidden.

4.6.1 K-Pup Problem with No Waiting Rings Allowed

For the problem when K=2, Bossert [16] gives a simple polynomial-time algorithm based on one key observation. We now review his proof.

Theorem 4.13 ([16]) *The 2-Pup Problem with no waiting rings allowed can be solved in polynomial time.*

Proof: Assume we are given two pups, each with a corresponding source and sink. We claim that there exists an optimal solution where all of the sharing between pups 1 and 2 is done in a continuous path.

To see this, suppose arcs (a_1, a_2) and (b_1, b_2) are shared by both pups in an optimal solution. Since there are no waiting rings in this solution, it must correspond to a feasible routing of the same cost. Without loss of generality, suppose (a_1, a_2) is shared first in this routing. Because we are at an optimal solution, the $a_2 - b_1$ path used by pup 1 must have the same cost as the $a_2 - b_1$ path used by pup 2. Otherwise, we could reroute one of the pups along the cheaper $a_2 - b_1$ path and obtain a lower cost. In particular, this means that we can route both pups along the same $a_1 - b_2$ path without increasing cost. Hence there exists an optimal solution where all sharing between pups 1 and 2 is done on consecutive arcs.

This suggests a simple enumerative method of solving the problem: form all $n(n-1)$ choices of nodes a and b. For each pair, calculate the cost of the shortest $s_1 - a$, $s_2 - a$, $a - b$, $b - t_1$, and $b - t_2$ paths. Combine these costs to obtain the cost of a feasible solution where pups 1 and 2 share along the $a - b$ path. Compare this to the situation where pups 1 and 2 do not share (i.e., they are routed along their shortest paths). Return the solution with the smallest cost overall.

This algorithm runs in polynomial time since the shortest path algorithm runs in polynomial time and we are running it a polynomial number of times. □

Unfortunately, the same reasoning does not apply to situations with $K \geq 3$ pups, as we now show. This means that to address the K-Pup problem for values of K greater than 2, we will need a different approach.

Lemma 4.14 *In a problem with $K \geq 3$ pups, the optimal solution might assign the sharing between two of the pups to be done inconsecutively.*

Proof: Consider the example in Figure 4-27.

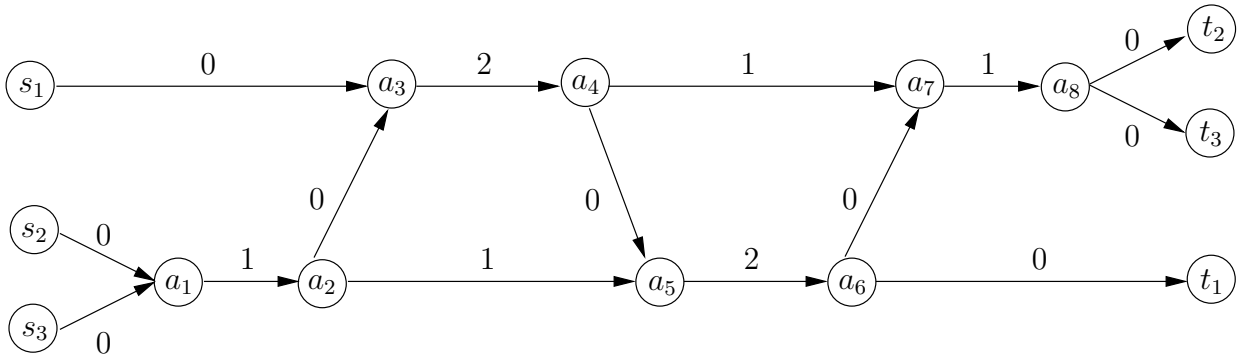

Figure 4-27: Example where two pups are assigned to share inconsecutively

Here there are three pups, with sources and sinks as shown. The optimal solution has a cost of 8, as shown in Figure 4-28. Pup 1 is routed on the path $s_1 - a_3 - a_4 - a_5 - a_6 - t_1$, pup 2 is routed on the path $s_2 - a_1 - a_2 - a_3 - a_4 - a_7 - a_8 - t_2$, and pup 3 is routed on the path $s_3 - a_1 - a_2 - a_5 - a_6 - a_7 - a_8 - t_3$.

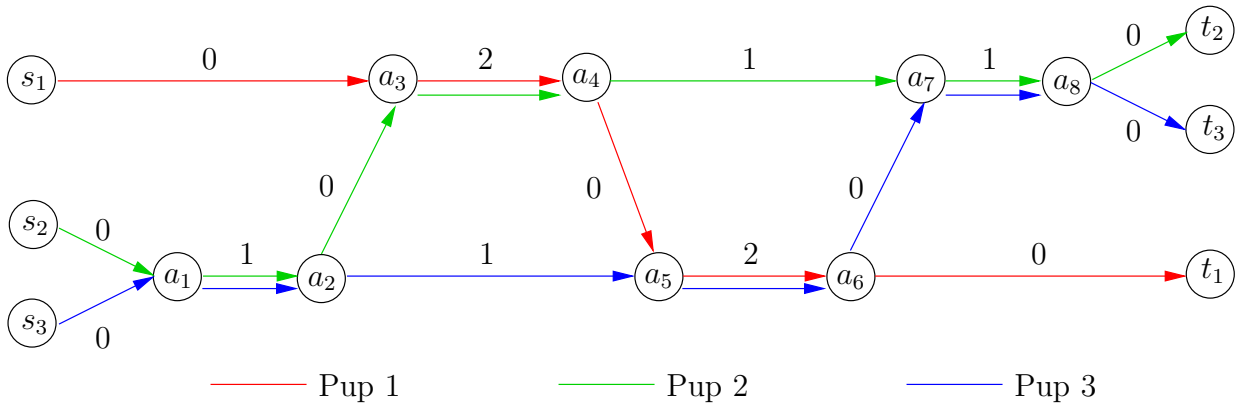

Figure 4-28: Optimal solution to the integer program where two pups share inconsecutively

Notice in the optimal solution that the routing of pup 2 follows the pattern:

alone—shares with 3—alone—shares with 1—alone—shares with 3—alone.

Similarly, pup 3 first shares with pup 2, then later with pup 1, and then with pup 2 again. Thus the sharing of pups 2 and 3 is done inconsecutively. To see that this is the only optimal solution, observe that the the best possible cost of a solution where all pups share consecutively is 9. One such routing is for pup 1 to proceed as shown and for pups 2 and 3 to share the path $a_1 - a_2 - a_3 - a_4 - a_7 - a_8$. □

Hence Bossert's proof technique does not apply to situations where $K \geq 3$. Fortunately, Li et al. [61] have proposed an alternate method that gives a polynomial time algorithm for arbitrary fixed values of K. We now review their proof.

Theorem 4.15 ([61]) *The K-Pup Problem with no waiting rings allowed can be solved in polynomial time.*

Proof: Suppose we are given a graph $G = (N, A)$, and we wish to ship K pups from nodes $s_1, \ldots, s_K$ to $t_1, \ldots, t_K$, for a fixed value of K. Construct a new graph $\overline{G} = (\overline{N}, \overline{A})$ as follows:

$$\overline{N} = \{< u_1, \ldots, u_K > \mid u_i \in N,\ i = 1, \ldots, K\} = \underbrace{N \times N \times \cdots \times N}_{K \text{ times}}.$$

$\overline{A} = \{(< u_1, \ldots, u_K >, < v_1, \ldots, v_K >) \mid$ $u_i \neq v_i$ for either 1 or 2 values of i, and there is an arc $(u, v) \in A$ such that for every i with $u_i \neq v_i$, $u_i = u$ and $v_i = v$.$\}$

$c(< u >, < v >)$ = total cost of the pup transition(s) between positions u and v.

For the instance

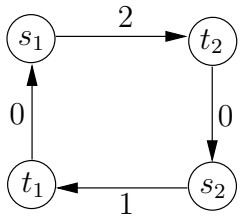

Figure 4-29: An instance of the Pup Matching problem

the corresponding graph is as shown in Figure 4-30.

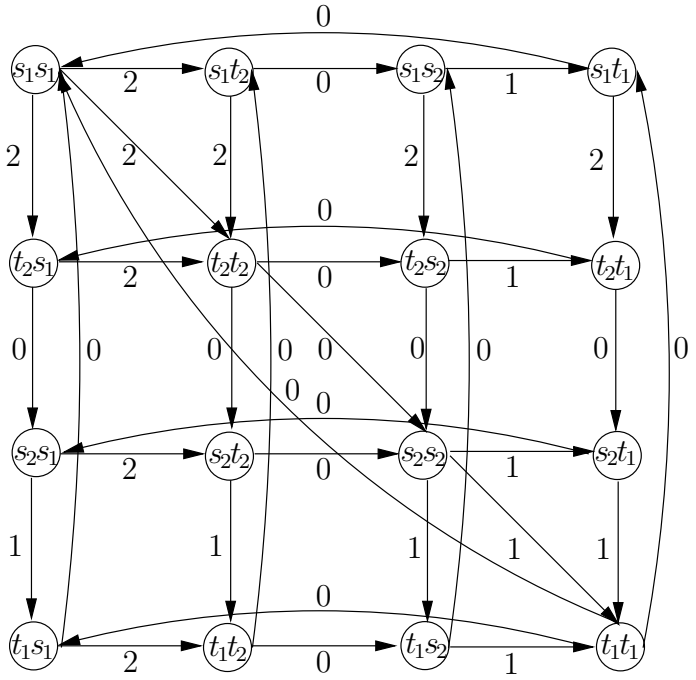

Figure 4-30: Expanded graph corresponding to the previous example

Each of the nodes in $\overline{G}$ represents a potential current location of the K pups originating from nodes $s_1, \ldots, s_K$. Specifically, node $< u_1, \ldots, u_K >$ corresponds to the state where pup i is located at node u_i, for all i. Each arc $(< u_1, \ldots, u_K >, < v_1, \ldots, v_K >)$ represents a single truck delivery across arc (u, v), changing the current locations of the pups from $< u_1, \ldots, u_K >$ to $< v_1, \ldots, v_K >$. The cost of the arc reflects the cost of this transition in the original graph.

We claim the shortest path from $< s_1, \ldots, s_K >$ to $< t_1, \ldots, t_K >$ in $\overline{G}$ gives an optimal solution to the Pup Matching problem without waiting rings. To see this, observe that the sequence of the arcs taken in the shortest path provides a means of ordering the pup transitions in time. Thus there are no waiting rings. Moreover, since the arc costs in $\overline{G}$ model the actual pup matching costs, any path from $< s_1, \ldots, s_K >$ to $< t_1, \ldots, t_K >$ in $\overline{G}$ corresponds to a feasible routing in G and the shortest path corresponds to an optimal solution.

The number of nodes in $\overline{G}$ is $|N|^K$, and the number of arcs in $\overline{G}$ is polynomial in the number of nodes. Hence the shortest path from $< s_1, \ldots, s_K >$ to $< t_1, \ldots, t_K >$ can be found in polynomial time. $\square$

4.6.2 K-Pup Problem with Waiting Rings Allowed

The K-Pup problem with waiting rings allowed is not as straightforward as the problem with waiting rings forbidden. We cannot use Li et al.'s expanded graph framework from the previous section because any such solution sequences pups in time, thus producing only solutions without waiting rings. We also cannot use Bossert's approach, since even when $K = 2$ it is possible for two pups to share inconsecutively. (For an example of this, consider Figure 4-18 in the section on waiting rings.) Instead, a different approach is needed.

We now derive a class of approximation algorithms for the Pup Matching problem with waiting rings allowed. These algorithms are motivated by the following result.

Theorem 4.16 *The 2-Pup problem with waiting rings allowed can be solved in polynomial time.*

Proof: The 2-Pup problem is equivalent to the following network design problem:

> Given a network $G = (N, A)$, arc costs, and source-sink pairs (s_1, t_1) and (s_2, t_2), determine a minimum cost subgraph of G that contains paths from s_1 to t_1 and s_2 to t_2.

The equivalence follows from noting that in the 2-Pup problem, the cost of each arc will be contributed at most once in the optimal solution. This is since each arc can be shared at most once. Hence we can model the problem as a network design problem, where the choice is whether or not to buy each arc and the goal is to obtain paths from s_1 to t_1 and s_2 to t_2.

This network design problem is a special case of the Point-to-Point Connection problem, with a fixed demand of 2. By [61], such problems are solvable in polynomial time via a dynamic programming algorithm. □

Unfortunately, the same result does not extend to other cases of the K-Pup problem with waiting rings. This is since when $K > 2$, each arc can contribute its cost multiple times in an optimal solution. Specifically, each arc a contributes its cost $\lceil \frac{f_a}{2} \rceil$ times, where f_a is the amount of flow on that arc. Such a cost structure cannot be modeled by an uncapacitated network design problem. (It is possible to model such problems as capacitated network design problems, but these are NP-hard in general and the approximation ratios are so poor as to discourage further results.)

However, if we employ a different approach, we can obtain promising *approximation* results. In particular, we propose the following algorithm for the 3-Pup problem:

3-Pup Problem Algorithm

Input: A graph $G = (N, A)$ with sources $s_1, s_2, s_3 \in N$, sinks $t_1, t_2, t_3 \in N$, and arc costs.
Output: A feasible solution for the 3-Pup problem with waiting rings allowed.

1. Compute the optimal cost of pairing pups 1-2, 1-3, 2-3 using the polynomial algorithm for the 2-Pup problem. Also find the cost of sending 1, 2, 3 alone using shortest paths.

2. Compute the lowest cost of

$$\text{1-2 (paired) + 3 (shortest path)}$$
$$\text{1-3 (paired) + 2 (shortest path)}$$
$$\text{2-3 (paired) + 1 (shortest path).}$$

Return the solution corresponding to the smallest combined cost.

Theorem 4.17 *The 3-Pup Problem Algorithm is a $\frac{5}{3}$-approximation algorithm, and this bound is tight.*

Proof: Let $c(ALG)$ be the cost of the solution produced by the algorithm and $c(OPT)$ the cost of the optimal solution. In addition, define SP_{opt} as the cost of the least expensive of the shortest paths, and $MAXPAIR$ the cost of the most expensive of the 2-Pup pairings. By definition of the algorithm, we have

$$c(ALG) = \text{optimal(2-pairing + shortest path)} \ \leq MAXPAIR + SP_{opt},$$

since one potential solution is taking the shortest path overall and then pairing the two remaining pups.

If we pick two arbitrary pups and neglect the third, the solution to this 2-Pup problem will always be a lower bound for the 3-Pup optimum. Hence

$$MAXPAIR \leq c(OPT).$$

We know that combining the shortest path solutions gives a 2-approximation algorithm, so

$$\text{SP for 1 + SP for 2 + SP for 3} \leq 2c(OPT)$$

which implies

$$SP_{opt} \leq \frac{2}{3}c(OPT).$$

Altogether, this gives

$$c(ALG) \leq c(OPT) + \frac{2}{3}c(OPT) = \frac{5}{3}c(OPT).$$

Finally, observe that this is a polynomial-time algorithm, since the 2-Pup algorithm runs in polynomial time. Hence the algorithm is a $\frac{5}{3}$-approximation algorithm.

This bound is tight, as is shown by the following example:

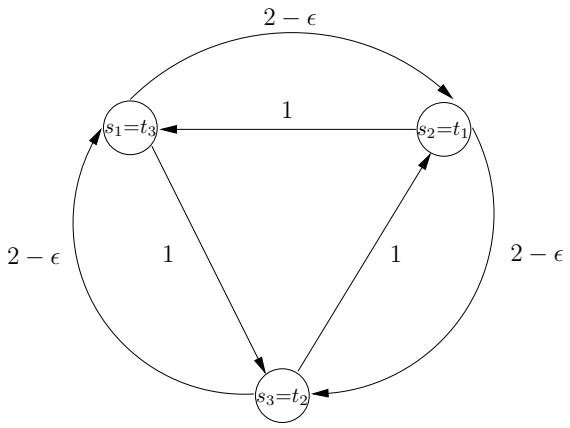

Figure 4-31: Example showing that the $\frac{5}{3}$ bound is tight

The shortest paths for each pup have length $2-\epsilon$, using the clockwise arcs. The solutions to the 2-Pup algorithm for pairings 1-2, 1-3, and 2-3 all have cost 3, using the counter-clockwise arcs. Hence the algorithm returns a solution of cost $5-\epsilon$. The optimal solution to the problem is to have each one of the counter-clockwise arcs shared by two pups, for a cost of 3. This solution contains a waiting ring.

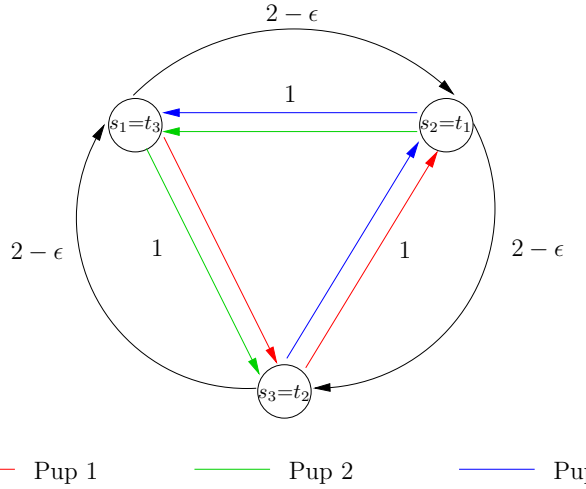

Figure 4-32: Optimal solution to the example showing that the $\frac{5}{3}$ bound is tight

The ratio of the solutions is $\frac{5-\epsilon}{3}$, which approaches $\frac{5}{3}$ as ϵ goes to 0. Note here we also have that $MAXPAIR = c(OPT) = 3$, and SP_{opt} approaches $\frac{2}{3}c(OPT)$ as ϵ goes to 0, so both of the bounds we used in the previous proof are tight. □

We next turn our attention to greater values of K. For $K = 4$, a similar algorithm gives the same performance bound.

4-Pup Problem Algorithm

Input: A graph $G = (N, A)$, sources $s_1, s_2, s_3, s_4 \in N$, sinks $t_1, t_2, t_3, t_4 \in N$, and arc costs.
Output: A feasible solution to the 4-Pup problem with waiting rings allowed.

1. Compute the cost of pairing pups 1-2, 1-3, 1-4, 2-3, 2-4, and 3-4 using the polynomial algorithm for the 2-Pup problem.

2. Compute the lowest cost of

   1-2 (paired) + 3-4 (paired)
   1-3 (paired) + 2-4 (paired)
   1-4 (paired) + 2-3 (paired).

   Return the solution corresponding to the smallest combined cost.

Before deriving the performance bound, we introduce some terminology. Let:

$c(ALG)$ = cost of solution returned by 4-Pup Problem algorithm.
$c(OPT)$ = cost of optimal solution to 4-Pup problem.
SH_{ij} = cost of flow shared between pups i and j in optimal solution.
$c(OPT)_i$ = cost of routing pup i in the optimal solution, disregarding sharing.
$OPTNS$ = cost of optimal solution with no sharing = $\sum_{i=1}^{4} c(OPT)_i$.

Theorem 4.18 *The 4-Pup Problem Algorithm is a $\frac{5}{3}$-approximation algorithm, and this bound is tight.*

Proof: First, notice that

$$c(ALG) \leq (c(OPT)_1 + c(OPT)_2 - SH_{12}) + (c(OPT)_3 + c(OPT)_4 - SH_{34}),$$

since the solution to the 2-Pup problem with pups 1-2 (3-4) paired is at most the cost of the 1-2 (3-4) routings in the optimal solution. Similarly,

$$c(ALG) \leq (c(OPT)_1 + c(OPT)_3 - SH_{13}) + (c(OPT)_2 + c(OPT)_4 - SH_{24})$$

and

$$c(ALG) \leq (c(OPT)_1 + c(OPT)_4 - SH_{14}) + (c(OPT)_2 + c(OPT)_3 - SH_{23}),$$

by enumerating all choices for the pairs of pups returned by the algorithm. Adding these three inequalities, we see

$$3c(ALG) \leq 2OPTNS + c(OPT).$$

We also know

$$OPTNS \leq 2c(OPT),$$

since the cost of each arc can at most double when there is no sharing. This implies that

$$3c(ALG) \leq 5c(OPT) \quad \Rightarrow \quad c(ALG) \leq \frac{5}{3}c(OPT),$$

using the inequality above. Finally, note the algorithm runs in polynomial time since the 2-Pup algorithm runs in polynomial time.

To observe that the bound is tight, we use the same example (Figure 4-31) as for the 3-Pup algorithm. Just add a fourth source and sink pair with a distance of 0 from source to sink. □

In generalizing the 3-Pup and 4-Pup algorithms, observe that the number of partitions of K pups (K even) into mutually disjoint pairs is

$$\frac{(K)!}{2^{\frac{K}{2}}(\frac{K}{2})!}.$$

This is the same as the number of perfect matchings in a complete graph with K nodes. It arises since there are $\frac{K!}{2^{\frac{K}{2}}}$ ways to choose $\frac{K}{2}$ pairs out of K pups; we divide by $(\frac{K}{2})!$ because the order of the pairs is irrelevant. This leads to:

K-Pup Problem Algorithm

Input: A graph $G = (N, A)$ with sources $s_1, s_2, \ldots, s_K \in N$, sinks $t_1, t_2, \ldots, t_K \in N$, and costs on the arcs.

Output: A feasible solution for the K-Pup problem with waiting rings allowed.

1. Compute the optimal cost of all $\binom{K}{2}$ pup pairs using the 2-Pup algorithm.
2. Compute the lowest total cost of the pairs (pairs plus shortest path, for K odd) in each of the $\frac{(2\lceil\frac{K}{2}\rceil)!}{2^{\lceil\frac{K}{2}\rceil}(\lceil\frac{K}{2}\rceil)!}$ mutually disjoint pair combinations. Return the solution corresponding to the smallest combined cost.

Theorem 4.19 *The K-Pup Problem Algorithm is a 2-* $\frac{2^{\lceil\frac{K}{2}\rceil}\lceil(\frac{K}{2})\rceil!}{(2\lceil\frac{K}{2}\rceil)!}$*-approximation algorithm, for fixed values of K.*

Proof: The proof of the performance bound is virtually identical to the proof of the 4-Pup algorithm. The algorithm runs in polynomial time for fixed values of K, since $\binom{K}{2}$ is $O(K^2)$, and the 2-Pup algorithm runs in polynomial time. □

As K becomes large, the performance ratio of the K-Pup Problem Algorithm approaches 2. Hence asymptotically, the performance of this algorithm is comparable to that of the shortest path algorithm discussed earlier. In the next section, we shift our focus onto a special class of problems dealing with an arbitrary number of pups.

4.7 The C-Problem

We define the *C-Problem* as the decision problem:

"Is there a feasible solution to the Pup Matching problem with cost at most C?"

We assume that a network is given with integral arc costs, unless otherwise noted. This problem is interesting because if we can show that the C-Problem is NP-hard for some fixed value of C, it implies that no approximation algorithm can have a performance guarantee strictly better than $\frac{C+1}{C}$ unless $P = NP$. (If there was such an algorithm, it would give a polynomial-time algorithm for the decision problem.)

As in the previous section, we consider separately the cases where waiting rings are allowed and where they are forbidden.

4.7.1 C-Problem with No Waiting Rings Allowed

Theorem 4.20 *For the Pup Matching problem with no waiting rings allowed and integral arc costs, the C-Problem is solvable in polynomial time for any fixed value of C.*

Proof: Assume we are given $G = (N, A)$ and a fixed value of C. Since the arc costs are integral, any pup that follows a path of nonzero cost must follow a path of length at least 1. Hence there can be at most $2C$ pups following paths of nonzero length. This suggests the algorithm:

Algorithm for C-Problem with No Waiting Rings Allowed

1. Determine which pups can follow paths of length 0 in the graph by deleting all arcs of positive cost and seeing whether s_i and t_i are still connected. Discard all such pups.
2. If there are more than $2C$ pups remaining, reject. Else, go to step 3.
3. Run the algorithm for fixed demand and no waiting rings (Theorem 4.15) on the remaining pups. If the solution returned has cost no more than C, accept. Else, reject.

The algorithm first determines whether there are at most $2C$ pups following paths of nonzero length; if there are, it checks to see whether the solution to the problem restricted to these pups has cost no more than C. The algorithm returns the correct answer, by our previous observation. It also runs in polynomial time, since step 1 can be executed via a breadth-first search and step 3 uses the polynomial-time algorithm from Theorem 4.15 for a fixed number of pups. □

The same argument establishes the following corollary.

Corollary 4.21 *For the Pup Matching problem where no waiting rings are allowed and the cost of all nonzero arcs is bounded from below by a fixed positive constant, the C-Problem is solvable in polynomial time.*

The majority of the real-world instances of this problem have one of these two cost structures, so in practice the C-Problem is solvable in polynomial time.

4.7.2 C-Problem with Waiting Rings Allowed

For the problem with waiting rings allowed, the situation is more complicated. For integral costs, we again know that at most $2C$ pups can have nonzero cost; however, since we do not have an algorithm to optimally solve for a constant number of pups with waiting rings allowed, this is not as helpful. However, we can solve some special cases of the problem, as we now discuss.

Theorem 4.22 *The 1-Problem with waiting rings allowed is solvable in polynomial time.*

Proof: There are three situations in which a favorable outcome can arise:

Situation A: The shortest paths for all pups have length 0.

Situation B: The shortest path for 1 pup has length 1; all others have length 0.

Situation C: The shortest paths for 2 pups have length 1; all others have length 0.

Consider the following algorithm:

1. Run a shortest path algorithm for every pup origin-destination pair.
2. If the result is anything other than situation A, B, or C, return "False."
3. If the result is situation A or B, return "True."
4. If the result is situation C, suppose pups 1 and 2 have shortest path length 1. For every arc (a, b) with cost 1 in the network, create a new graph $G_{(a,b)}$ in which all other arcs with positive cost are removed.
5. Run a breadth-first search on $G_{(a,b)}$ to determine whether t_1 and t_2 are reachable from s_1 and s_2 respectively.
6. If the result of any of the bfs's is "True," return "True." Else, return "False."

The algorithm is correct because situations A and B are trivially "True"; for situation C, steps 4-6 test whether it is possible for the two pups with shortest path length 1 to share in an optimal solution.

It takes $O(n^4)$ time to run the shortest path algorithms, since there are $O(n^2)$ choices for the pup pairs and it takes $O(n^2)$ time to find the shortest path for each pair. There are $O(m)$ choices for the arc (a, b), and for each choice it takes $O(m)$ time to form $G_{(a,b)}$ and $O(m)$ time to run the bfs. Hence the overall running time is $O(n^4) + O(m^2)$, which is polynomial-time. □

In this case we could also have used the algorithm for when waiting rings are not allowed, since there are no waiting rings in a solution with only one positive cost arc. However, the method is useful in understanding the following result.

Theorem 4.23 *The 2-Problem with waiting rings allowed is solvable in polynomial time.*

Proof: There are now five situations in which a favorable outcome can arise:

Situation A: The sum of the shortest path lengths for all pups is 0, 1, or 2.

Situation B: The shortest paths of 3 pups have length 1; all others have length 0.

Situation C: The shortest paths of 4 pups have length 1; all others have length 0.

Situation D: The shortest paths of 1 or 2 pups have length 1; 1 has length 2; all others are 0.

Situation E: The shortest paths of 2 pups have length 2; all others have length 0.

Consider the following algorithm:

1. Run a shortest path algorithm for each pup origin-destination pair.
2. If the result is anything other than situation A, B, C, D, or E, return "False."
3. If the result is situation A, return "True."
4. If the result is situation B, suppose pups 1, 2, and 3 are the pups with shortest path length 1. Run the 1-algorithm on pups 1-2, 1-3, and 2-3. If any of these instances return "True," return "True"; else, return "False."
5. If the result is situation C, suppose pups 1, 2, 3, and 4 are the pups with shortest path length 1. Run the 1-algorithm on pup pairs 1-2, 1-3, 1-4, 2-3, 2-4, 3-4. If either pair and its complement (i.e. the two pups not in the pair) are both "True," return "True." Else, return "False."
6. If the result is situation D, suppose without loss of generality that pups 1 and 2 have shortest path length 1 and pup 3 has shortest path length 2. Form all possible pairs $\{a_1, a_2\}$ of arcs with cost 1. Create graphs $G_{(a_1,a_2)}$ and $G_{(a_2,a_1)}$ by deleting all other arcs of positive cost and assigning cost 2 to a_1 in $G_{(a_1,a_2)}$ and cost 2 to a_2 in $G_{(a_2,a_1)}$. (The construction for when there is only one pup with shortest path length 1 is very similar.)
7. Run shortest paths for pups 1, 2, and 3 on $G_{(a_1,a_2)}$ and $G_{(a_2,a_1)}$. If the sum of the shortest path lengths is ≤ 6 for both cases, return "True"; else, return "False."
8. If the result is situation E, suppose pups 1 and 2 are the pups with shortest path length 2. In a favorable solution either both pups share one arc of cost 2, or they share two arcs of cost 1.

9. To check for the first case, form a graph G' where the cost of all arcs of cost 2 in G is changed to 1 and all arcs with cost 1 are deleted. Run the 1-algorithm with pups 1 and 2 on G'.

10. To check for the second case, form all possible pairs $\{a_1, a_2\}$ of arcs with cost 1. For each pair, form a graph $G_{(a_1,a_2)}$ by deleting all other arcs with positive cost. Run a bfs on $G_{(a_1,a_2)}$ to determine whether there exists a $s_1 - t_1$ and a $s_2 - t_2$ path.

11. If either of the previous check cases return "True," return solution "True"; else, return "False."

The correctness of the algorithm follows by inspection and by comparison to the 1-algorithm. The running time to find the shortest paths is $O(n^4)$, again because there are $O(n^2)$ choices for the pup pairs and $O(n^2)$ time to run the shortest path algorithms. In situations B and C, the running time is $O(n^4) + O(m^2)$, as in the 1-algorithm. In situation D, there are $O(m^2)$ arc pairs, for which it takes $O(m)$ time to form $G_{(a_1,a_2)}$ and $O(n^2)$ time to run the shortest paths. For situation E, the first check takes $O(n^4) + O(m^2)$ time and the second check takes $O(m^3)$ time, since there are $O(m^2)$ pairs and it takes $O(m)$ time to form the new graphs and run the bfs. Hence the running time of the entire algorithm is $O(n^4) + O(m^3n^2)$, which is polynomial in the size of the input. □

We can extend this approach to give algorithms for the C-Problem with waiting rings allowed for $C \geq 3$, but in general this approach is not practical. The number of partitions of C grows exponentially; for $C = 3$ there are already 16 cases to check and for $C = 4$ there are 40. In general the number of partitions of C is greater than $2^{\sqrt{C}}$, which suggests that establishing algorithms for large values of C is not computationally feasible. Nevertheless, the results established in this section are valuable for the special cases that they model.

4.8 Nash Equilibria and Discounting

In this section, we consider the pups as *autonomous users* in a network, rather than commodities to be routed. Each user wishes to travel from their source s_i to their destination t_i in a minimum cost manner. The cost we choose to associate with user i in solution s is:

$$c_i(s) = \sum_{a \in A \mid \text{ pup } i \text{ takes } a \text{ singly}} c_a + \frac{1}{2} \sum_{a \in A \mid \text{ pup } i \text{ shares } a} c_a.$$

In other words, a user pays the full arc cost if the arc is taken singly; the user pays half the cost if the arc is shared. This cost structure presupposes that the pairing of all pups is explicitly indicated in the solution. (We could also define a cost structure based on having pups split the total cost of the arc evenly, and all results in this subsection would still hold.)

An optimal solution to the Pup Matching problem from a central perspective is one that routes all users through the network such that the sum of the costs of each user is minimized. We refer to this solution as a *system optimal* solution. We call a solution a *user optimal* solution if no one user i can decrease their associated cost by switching to a different $s_i - t_i$ path. Such a solution is also known as a *Nash equilibrium*, as described in Section 2.4.

The results we prove in this section will hold for versions of the problem both with and without waiting rings. (For simplicity, it may be easiest to consider the problem without waiting rings allowed while reading the results.)

4.8.1 Nash Equilibria

We first note that a Nash equilibrium may not always correspond to a system optimal solution, and vice versa. For example, consider the following problem instance:

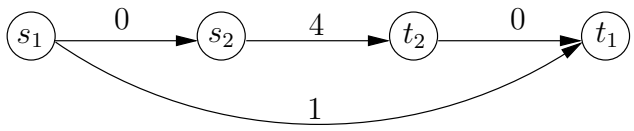

Figure 4-33: Example showing that Nash equilibria and system optima may be different

A Nash equilibrium for this example is for pup 1 to take arc (s_1, t_1) and pup 2 to take (s_2, t_2):

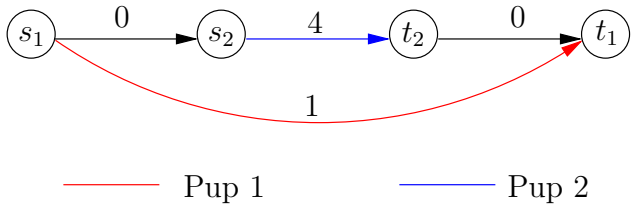

Figure 4-34: Nash equilibrium for the previous example

Here, the cost assigned to pup 1 is 1 and the cost assigned to pup 2 is 4. If pup 1 were to switch to the path $s_1 - s_2 - t_2 - t_1$, then the cost associated with pup 1 would be 2. This is higher than pup 1's current cost, so it will not want to switch.

The system optimal solution is for pup 1 to take the path $s_1 - s_2 - t_2 - t_1$ and pup 2 to take the arc (s_2, t_2):

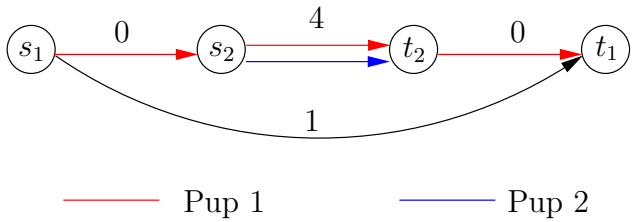

Figure 4-35: System optimal solution for the previous example

Here, the cost assigned to pup 1 is 2 and the cost assigned to pup 2 is 2, for a total of 4. This is smaller than the cost associated with the previous Nash equilibrium, implying that the Nash equilibrium is not a system optimal solution. Moreover, this system optimal solution is not a Nash equilibrium because if pup 1 switches to the path $s_1 - t_1$, it can lower its associated cost from 2 to 1. Hence a user optimal solution may not correspond to a system optimal solution, and vice versa.

We next note that (pure) Nash equilibria are *always guaranteed to exist* in the Pup Matching problem, as we can formulate the Pup Matching problem as an unweighted network congestion game (see Sections 2.3.2 and 5.5), and Nash equilibria always exist in such

games [67, 73]. The question of whether a Nash equilibrium can be computed efficiently is probably a difficult one to answer, as for small numbers of pups the problem resembles the Fair Connection game studied by Anshelevich et al. [8], for which no polynomial-algorithm is known for finding a Nash equilibrium.

Because Nash equilibria are guaranteed to exist for the Pup Matching problem, a natural question is to determine the worst-case performance gap (the *price of anarchy*, as discussed in Section 2.4) between Nash equilibria and system optimal solutions. We obtain the following result.

Theorem 4.24 *The price of anarchy in the Pup Matching problem is at most 2.*

Proof: Suppose we have a Nash equilibrium x for an instance of the Pup Matching problem over a graph $G = (N, A)$. We first claim that the cost assigned to pup i in x must be less than or equal to the length of the shortest path from s_i to t_i in G (disregarding sharing). This is since one possible routing of pup i is to traverse its shortest path and not share with any other pups. Since we are at a user optimal solution, the cost assigned to i must be less than or equal to the cost of this path (or we could switch). Hence

$$c(x) \leq \sum_i (\text{shortest } s_i - t_i \text{ path length}).$$

Now, from Theorem 4.2, we know that

$$\sum_i (\text{shortest } s_i - t_i \text{ path length}) \leq 2OPT,$$

where OPT is the cost of the system optimal solution. This follows since the cost of the solution consisting of all shortest paths is within a factor of 2 of optimum. Thus $c(x) \leq 2OPT$, and the claim is proved. □

We now show that this bound is actually tight; that is, there exist examples where the user optimal solution is arbitrarily close to a factor of 2 from the cost of the system optimal solution. One such example is shown in Figure 4-36.

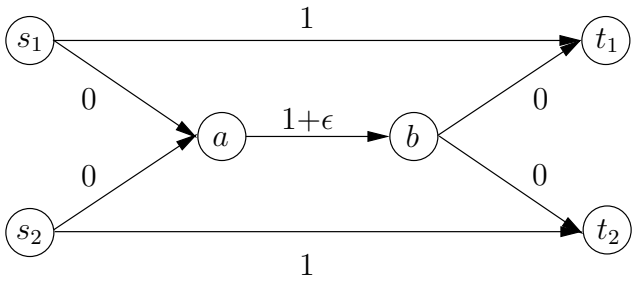

Figure 4-36: Example in which user and system optimal solutions may be a factor of 2 apart

In this example, a Nash equilibrium arises when pup 1 takes arc (s_1, t_1) and pup 2 takes arc (s_2, t_2), for a total cost of 2. Neither pup will want to switch to the arc of cost $1 + \epsilon$, because it would increase their assigned cost. The system optimal solution is for pup 1 to take the path $s_1 - a - b - t_1$ and pup 2 to take the path $s_2 - a - b - t_2$, with both pups sharing arc (a, b). The cost of this solution is $1 + \epsilon$.

An interesting question is what happens when we make the price of a path appear to be cheaper if it is unused. This leads us to the idea of discount properties, which we cover in the next subsection.

4.8.2 Discounting Properties

We extend the concept of a Nash equilibrium as follows:

> We say a solution satisfies the *d-discount property* $(0 < d \leq 1)$ if for each pup, the cost of the path the pup is using (as defined in Section 4.8) is cheaper than d times the cost of any other path the pup could switch to.

In other words, if this property holds then it is not beneficial to switch paths even when an incentive is given.

Recall our notation from a previous section:

> Let OPT be an optimal solution to the Pup Matching problem, of cost $c(OPT)$.
> Let SH_{ij} be the total cost of the flow shared between pups i and j in OPT.
> Let SH_{jj} be the total cost of the flow that is not shared by pup j in OPT.

With this notation, the optimal cost satisfies $c(OPT) = \sum_{i,j} SH_{ij}$.

Theorem 4.25 *If a solution satisfies the d-discount property, then its cost is within a factor of*

$$\begin{array}{ll} 1 + \frac{d}{2} & d \leq \frac{2}{3} \\ 2d & d > \frac{2}{3} \end{array}$$

of the optimal cost.

Proof: Let s be a solution satisfying the d-discount property, for some value of d. Let $c_j(s)$ be the cost associated with routing pup j in the solution s. (With this notation, the cost of s is $c(s) = \sum_j c_j(s)$.) Let OPT be an optimal solution.

Consider the solution s. We will compare the cost of s to the cost of the optimal solution, which is $\sum_{i,j} SH_{ij}$. We do so by bounding the cost contributed by each pup, and summing over all the pups. (Again, we assume that in any solution the pairing of the pups is specified.) Observe:

- If pup j is routed on its optimal path (the same path as in OPT),

$$c_j(s) \;\leq\; SH_{jj} + \frac{1}{2}\sum_{i|i \text{ on its optimal path, } i\neq j} SH_{ij} \;+ \sum_{i|i \text{ not on its optimal path, } i\neq j} SH_{ij}.$$

- If pup j is not routed on its optimal path,

$$c_j(s) \;\leq\; d\left(SH_{jj} + \frac{1}{2}\sum_{i|i \text{ on its optimal path, } i\neq j} SH_{ij} \;+ \sum_{i|i \text{ not on its optimal path, } i\neq j} SH_{ij}\right).$$

The first inequality is an upper bound on the cost of the optimal path in s. The second inequality uses the same upper bound, but discounted by a factor of d because pup j is not on its optimal path.

If we add together these inequalities for every j, we obtain a bound for $c(s) = \sum_j c_j(s)$ in terms of a sum of SH_{ij} values. The highest coefficient of a term in the sum is

$$\begin{array}{ll} 1 + \frac{d}{2} & d \leq \frac{2}{3} \\ 2d & d > \frac{2}{3} \end{array}$$

by inspection. (Each SH_{ij} term is represented at most twice in the set of inequalities, so we sum these values and compare.) This implies that if $d \leq \frac{2}{3}$,

$$c(s) = \sum_j c_j(s) \leq \sum_{i,j} (1 + \frac{d}{2}) SH_{ij} = (1 + \frac{d}{2}) OPT.$$

If $d > \frac{2}{3}$,

$$c(s) = \sum_j c_j(s) \leq \sum_{i,j} 2dSH_{ij} = 2dOPT. \ \square$$

Thus we have obtained results on the performance of Nash equilibria and solutions satisfying discounting properties. Unfortunately, not all instances contain equilibria that satisfy discounting properties, so this technique is only useful for those instances in which the d-discount property can be show to hold. In the next section, we shift our focus to a *capacatiated* variant of the Pup Matching problem.

4.9 Capacitated Pup Matching Problem

We define the Capacitated Pup Matching problem as the Pup Matching problem where there are capacities limiting the number of cabs that can cross each arc. This arises in real trucking networks when certain roads in the network can only carry a limited amount of traffic flow.

An instance of the Capacitated Pup Matching problem is given by a graph $G = (N, A)$, costs c_a for each arc $a \in A$, capacities u_a for each $a \in A$, and the collection $(s_1, t_1), \ldots,$ (s_K, t_K) of pup origin-desination pairs. With respect to complexity, we obtain the following result.

Theorem 4.26 *The Capacitated Pup Matching problem is NP-hard, and no approximation algorithm exists with a finite factor unless P=NP.*

Proof: We use the same basic construction as in the Pup Matching hardness result, modified slightly. Again we reduce from the Arc-Disjoint Paths problem.

Suppose we are given an instance $G = (N, A)$ of the Arc-Disjoint Paths problem, with node pairs $(s_1, t_1), \ldots, (s_K, t_K)$. We transform this into an instance of the Capacitated

Pup Matching problem as follows. For every arc (i, j) in A, replace (i, j) with the construction in Figure 4-37.

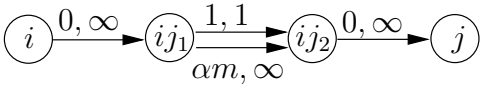

Figure 4-37: Transformation of arc (i, j) in the capacitated pup matching instance

Here, $m = |A|$ and the size of α is polynomial in the size of the input. The first number on an arc is its cost and the second is its capacity. We have added two new nodes for every arc and assigned costs to the arcs. All arcs are uncapacitated except the arcs of cost 1 from ij_1 to ij_2, which have a capacity of 1.

Let G' be the graph resulting from these transformations. Our instance of the Capacitated Pup Matching problem consists of G', along with the node pairs $(s_1, t_1), \ldots, (s_K, t_K)$ and (ij_1, ij_2) for all $(i, j) \in A$. In each of these node pairs, we wish to route one pup from the first node to the second node.

The same analysis as in Theorem 4.1 shows that if there exist arc-disjoint paths in G from s_i to t_i for all i, then the cost of the optimal solution is $\leq |A|$. If there do not exist such arc-disjoint paths, in any solution we must take one of the 'expensive' arcs of cost $\alpha|A|$. This shows:

There exist arc-disjoint paths $\Rightarrow$ the cost of the optimal solution is $\leq |A|$.

Arc-disjoint paths do not exist $\Rightarrow$ the cost of the optimal solution is $> \alpha|A|$.

Hence the Capacitated Pup Matching problem is NP-hard. Since α was arbitrary, we also have that no approximation exists with a finite factor, unless P=NP. $\square$

Notice that this argument applies for versions of the problem both with and without waiting rings allowed. Overall, the result suggests that obtaining algorithms within a provable performance guarantee is probably not possible, so future efforts on this problem would do best to focus on heuristic methods.

4.10 Final Comments on Pup Matching

We conclude by reviewing our findings and discussing areas for future research. We have examined properties of solutions of the Pup Matching problem, as well as several variants of the problem.

In Sections 4.3 and 4.4, we presented an integer programming formulation of the Pup Matching problem, and we investigated properties of its corresponding LP relaxation. We conjectured that the cost of the optimal solution to the LP relaxation is always within a factor of $\frac{4}{3}$ of the optimal IP solution. This is a major outstanding question and an interesting avenue for future research. If we could show that this bound holds, it might be possible to construct better approximation algorithms for the general problem.

During Section 4.6, we investigated the K-Pup problem both with and without waiting rings allowed. We gave a polynomial-time algorithm for the problem where waiting rings are not allowed, and we presented approximation algorithms for the problem where waiting rings are allowed. It would be useful to further determine the complexity of this problem for $K \geq 3$, as this is one area we did not investigate.

In Section 4.7, we discussed the C-Problem and gave algorithms for the cases with and without waiting rings allowed. It would be interesting to see if our algorithms for the case with waiting rings allowed could be extended to work for greater values of C, although this seems unlikely.

In Section 4.8 we defined and examined properties of Nash equilibria of the Pup Matching problem. Finally, in Section 4.9 we looked briefly at the Capacitated Pup Matching problem and concluded that it was not approximable within a finite factor unless P=NP. Another avenue for future research would be to look at special cases of the Capacitated Pup Matching problem and see if any of the earlier results could be extended.

Thus we have discussed several aspects and variants of the Pup Matching problem, as well as identified several areas for future research. It is interesting how such a simple variant on multicommodity flow can be shown to possess so many different layers of complexity.

Chapter 5

Complexity and Congestion Games

In this chapter, we classify the complexity of finding a minimum cost solution to network and general congestion game problems under the Rosenthal [73] model (as described in Section 2.3) with respect to several parameters. In particular, we introduce different variants of the problems according to the *structure* of the problems and the type of associated *cost functions*.

5.1 Introduction

Congestion games were introduced by Rosenthal [73] as a simple class of games possessing pure-strategy Nash equilibria. A formal definition is given in Section 2.3, which may be summarized as follows: we are given a finite number of players, each of which possesses a finite set of strategies. Each strategy consists of a subset of a master set of resources. The cost of employing a particular strategy is the sum of the costs of the resources associated with that strategy, where the cost of using a particular resource is solely a function of the number of players using that resource.

As described in Section 2.3.2, one example of a congestion game occurs when the set of strategies is associated with paths in a network; in a *network congestion game*, each player i is associated with two nodes s_i and t_i, and their set of strategies consists of all (simple) $s_i - t_i$ paths. The arcs play the role of the resources, and the cost associated with each arc

is a function of the number of players using that arc.

Rosenthal proposed two practical applications of congestion games, one concerning road networks and the other involving factory production. In the first application, a network of roads is given and each player travels from a certain origin to a certain destination. The cost of traveling on each road is an increasing function of the number of people traveling on that road (hence the use of the word 'congestion'). In the second application, a number of firms are engaged in production, each of which has several production processes available that employ different resources. The cost of using a resource is a function of the number of firms that use the resource. Rosenthal showed that regardless of the cost structure on the set of resources, such games always possess a (pure) Nash equilibrium.

Monderer and Shapley [67] generalized these congestion games to a class of games they called potential games, which are games that incorporate information about Nash equilibria in a single real-valued potential function over the strategy space. By definition, such games always possess pure Nash equilibria, and they have since been studied in their own right [34, 79, 85, 89, 90]. In particular, it has been shown that congestion games are isomorphic to potential games that admit an exact potential function [67, 90]. Others have examined potential games with an infinite set of strategies [89], continuous player sets [79], or incomplete information [34].

Another major variant of the problem is that of 'nonatomic' congestion games, in which the number of players is assumed to be so large that the effect an individual player has on the outcome is negligible. Roughgarden and Tardos [74, 77] provide a bound on the inefficency of pure Nash equilibria in such games, by comparing the cost of a Nash equilibrium to that of a best-possible outcome. They provide an exact bound on this worst-case efficiency under certain conditions on the cost function, as well as identify games for which the equilibria are approximately optimal. Correa, Schulz, and Stier-Moses [25, 26] later simplified, strengthened, and generalized these analyses, and Chau and Sim [20] extended the results to a more general class of nonlinear cost functions.

Returning to Rosenthal's original concept of congestion games, several researchers

have studied a special class of network congestion games consisting of n users traveling over m parallel links. Koutsoupias and Pamadimitriou [57] initiated this line of research, and were later followed by Czumaj and Vöcking [29], Czumaj, Krysta and Vöcking [28], Mavronicolas and Spirakis [63], and Koutsoupias, Mavronicolas, and Spirakis [56], among many others. Their focus was mainly on the price of anarchy for pure and mixed Nash equilibria in such games, which we further detail in Chapter 6. Others have looked at the case where the cost function is linear in the number of players [55, 84] or where players anticipate the effect of their actions on the price of the links [49].

In a related vein, other research [53, 64, 65] concerns the congestion game problem where players travel over a parallel set of links, but in which different players experience different *player-specific* amounts of congestion. Milchtaich [64, 65] shows that such problems always possess a pure Nash equilibrium, and that there is at least one sequence of 'best' moves that transform an arbitrary solution into an equilibrium. Conditions for optimality of such equilibria in the nonatomic case are established in [66].

Other recent work concerns the *existence of equilibria* in generalizations of congestion games, which we examine further in Chapter 6. Fotakis, Kontogiannis, and Spirakis [39] study the existence of equilibria in weighted congestion games, in which each player may control different amounts of demand (see Section 2.3.3). Holzman and Law-Yone [48] study necessary and sufficient conditions for the existence of a strong equilibrium, in which no coalition of players has an incentive to deviate to an alternate strategy that is profitable for all of its members. Beier, Czumaj, Krysta, and Vöcking [14] address the problem of computing a pure Nash equilibrium in congestion games with imperfect information.

Most relevant to our work, Fabrikant, Papadimitriou, and Talwar [33] initiated the study of *complexity* issues in congestion games. They showed that a (pure) Nash equilibrium can be computed in polynomial time in network congestion games with nondecreasing arc costs where all players share a source and sink, via a potential function; however, in general the problem is PLS-complete, which suggests it is unlikely that any locally optimal solution can be found for the potential function in polynomial time. Papadimitriou [70] showed that

a generalization of a Nash equilibrium known as a correlated equilibrium may be calculated in polynomial time in compactly encoded congestion games, which are congestion games in which the set of strategies are given implicitly rather than explicitly.

Furthering the study of complexity, Feldmann, Gairing, Lücking, Monien, and Rode [36] studied the problem of computing a pure Nash equilibrium starting from an arbitrary solution in a congestion game. They gave an $O(nm^2)$ algorithm to 'Nashify' a given solution on a network of parallel links. Gairing, Lücking, Mavronicolas, Monien, and Spirakis [42] examined the same problem, showing that for any $k > 0$ it is NP-hard to decide whether a solution can be 'Nashified' in k selfish steps. They further proposed that the 'worst' Nash equilibrium (in terms of cost) is the fully mixed nash equilibrium (also studied in [62, 63]), which was recently shown [37] to not always be the case.

Up to this point, most of the work on congestion games has concerned the existence and difficulty of finding Nash equilibria, as discussed here and in Chapter 6, and various properties of such equilibria. Our current work takes a slightly different approach, investigating the complexity of congestion games from a *system optimal* approach. Specifically, we address the complexity of finding an *overall minimum cost solution* to the congestion game problem. (Note that such a solution does not have to be a Nash equilibrium.)

One motivation for classifying the complexity of finding a minimum cost solution is that in some cases, we may be less interested in the performance of individual players than we are in the system optimum. This can occur in situations where the players are not selfish (i.e., if the players are all working together), but in which congestion effects are still felt. Another motivation arises in the work of Anshelevich et al. [8], who showed how to obtain a provably good Nash equilibrium in certain special cases when starting from an optimal solution. For this method to be relevant in practice, we must know the complexity of finding an optimal solution. A third motivation is that in some problems, such as network congestion games with a single source and sink and nondecreasing costs, finding a Nash equilibrium may be done efficiently (see [33]) while computing the optimum is NP-hard (as we will show). In such cases, an algorithm for finding a Nash equilibrium may be used as an approximation

algorithm for the problem of finding a minimum cost solution. In this way, our complexity results add a new interpretation to the concept of the price of anarchy.

The topic of system-optimal solutions in congestion games was recently and independently studied in a paper of Chakrabarty, Mehta, Nagarajan, and Vazirani [19], in which they examined a notably different model of congestion games due to Milchtaich [64]. In their model, the players possess different player-specific cost functions, which are increasing in the amount of congestion, and all players travel over a set of parallel links. They proved that finding a system-optimal solution to their problem is NP-hard and no finite-factor approximation algorithm exists. They showed that in the special case where all of the strategies cost the same and the matrix of player costs is anti-Monge, the system optimum may be computed in polynomial time. They also gave a number of complexity results relating to the difficulty of finding 'fair' (minmax) allocations.

Our work differs from that of Chakrabarty et al. in that we do not consider the case of parallel links or player-specific cost functions. We also consider a greater number of structural aspects of the problem and a variety of different resource cost functions, as we will describe later.

In what follows, we present our results on the complexity of finding system-optimal solutions to the network and general congestion game problems. In Section 5.2, we introduce variants of the problems differing in *structure* and the type of associated *cost functions*. With regards to structure, we consider whether all players have the same set of strategies (symmetric) or not (asymmetric). In the network case, we also consider whether players have the same source or sink. With respect to arc costs, we consider five different cost functions (nondecreasing, convex nondecreasing, nonincreasing, concave nonincreasing, and nonmonotonic) that model different forms of congestion and economies of scale.

We fully categorize the complexity of the network congestion game problem and all of its variants under these parameters in Section 5.3. In most cases, we find the problem is NP-hard; however, in four cases (symmetric games with convex nondecreasing, nonincreasing, or concave nonincreasing arc costs, and single source games with concave nonincreasing arc

costs) the problem is solvable in polynomial time.

We examine the complexity of the general congestion game problem in Section 5.4. In several cases, our results follow directly from the network case, but in others (for instance, convex nondecreasing costs) we are able to derive stonger results. Overall, we find that in almost all cases, the problems are NP-hard and difficult to approximate with a finite factor. The exceptions are the asymmetric case with concave nonincreasing costs, which is NP-hard (without a corresponding approximation result), and the symmetric case with nonincreasing or concave nonincreasing costs, which is solvable in polynomial time.

5.2 Problems Studied

We study the complexity of several variants of network and general congestion games, as defined in Section 2.3. There are two main types of variants we consider: *structural* variants and *cost* variants. We now outline each of these.

In terms of structural variants, we consider two basic alternatives: *symmetric* problems, in which all players share the same set of strategies, and *asymmetric* problems, where players may have different sets of strategies. In the network problem, the symmetric case corresponds to all players having the same source and sink, and the asymmetric case corresponds to having different sources and sinks. In addition, in the network problem we also consider the *single source* case, in which all players share a single source (but may have different sinks).

With regards to cost functions, we consider five different classes of cost structures. We say that the arc costs are *nondecreasing* if $c_a(1) \leq c_a(2) \leq \ldots \leq c_a(n)$ for all $a \in A$, and *nonincreasing* if $c_a(1) \geq c_a(2) \geq \ldots \geq c_a(n)$ for all $a \in A$. Nondecreasing cost functions model the negative effects of congestion on the availability of resources, while nonincreasing cost functions reflect economies of scale. We say that a cost structure is *convex nondecreasing* if it is nondecreasing and the differences between consecutive aggregate arc costs are nondecreasing; in other words, $ic_a(i)-(i-1)c_a(i-1) \leq (i+1)c_a(i+1)-ic_a(i)$ for all $i = 1, \ldots, n-1$. Similarly, we say a structure is *concave nonincreasing* if it is nonincreasing

and $ic_a(i) - (i-1)c_a(i-1) \geq (i+1)c_a(i+1) - ic_a(i)$ for all $i = 1, \ldots, n-1$. (Note that any function that is convex nondecreasing is also nondecreasing, and any function that is concave nonincreasing is also nonincreasing.) If an arc cost function fits into none of these categories, we say that it is *nonmonotonic*.

This gives us fifteen different problems in the network case (three structural variants and five cost variants), and ten different problems in the general case (two structural variants and five cost variants). As it turns out, many of the complexity results we prove apply to multiple problems, with minor changes.

5.3 Network Complexity Results

Our complexity results for network congestion games are as illustrated in Table 5.1.

type→ / costs ↓	symmetric	single source	asymmetric
nondecreasing	NP-hard [5.1]	NP-hard [5.1]	NP-hard; inapprox.
convex nondecreasing	P	P [5.6]	NP-hard; inapprox. [5.4]
nonincreasing	P [5.5]	NP-hard; inapprox. [5.2]	NP-hard; inapprox.
concave nonincreasing	P	NP-hard [5.3]	NP-hard
nonmonotonic	NP-hard; inapprox. [5.2]	NP-hard; inapprox.	NP-hard; inapprox.

Table 5.1: Complexity results for network congestion games.

(The numbers in brackets indicate in which theorem (or discussion thereafter) the results are proved. Note that the results for all unlabeled entries follow directly from other entries on the table. By 'NP-hard; inapprox.' we mean that the problem is NP-hard and no approximation algorithm exists with a finite factor, unless P=NP.)

We now prove these results. We cover the hardness results first: we begin by presenting the single source hardness results, and we show how slight modifications can be made to derive the symmetric hardness results. We then give the asymmetric hardness results, followed by the polynomial time algorithms.

Our first theorem concerns single source unweighted congestion games with nondecreasing costs. (Note that the hardness of this problem does not follow from the hardness of the general single-source unsplittable flow problem (see [9]), since this problem translates to *weighted* congestion games.) We assume that arc costs are given explicitly, rather than

compactly encoded (in which the costs are given implicitly as a function of the demand, so that the size of the encoding does not depend on the size of the demand). Our results also hold for compactly encoded cost functions except for Theorem 5.6.

Theorem 5.1 *The single source unweighted network congestion game problem with nondecreasing costs is strongly NP-hard.*

Proof: We reduce from the 3-PARTITION problem, which is strongly NP-complete [43]. This problem is:

Instance: A set S of $3q$ elements, a bound $B \in \mathbb{Z}^+$, and a size $s(i) \in \mathbb{Z}^+$ for each $i \in S$, such that $\frac{B}{4} < s(i) < \frac{B}{2}$ and $\sum_{i \in S} s(i) = qB$.

Question: Can S be partitioned into q disjoint sets $D_1, \ldots, D_q$ such that, for all $1 \leq j \leq q$, we have $\sum_{i \in D_j} s(i) = B$?

Suppose we are given an instance of the 3-PARTITION problem. Build the following congestion game (see Figure 5-1):

1. Create 1 source node s, with a supply of $qB + 3q^2$.
   Create $3q$ transshipment nodes s_i.
   Create q sink nodes D_j, each with demand B.
   Create $3q^2$ sink nodes a_{ij}, $1 \leq i \leq 3q$ and $1 \leq j \leq q$, each with demand 1.
2. Add arcs (s, s_i) of cost $0/\ldots/0/M/\ldots/M$, where the last '0' is in place $s(s_i)+q$ and M is a large number.
   Add arcs (s_i, a_{ij}) of cost $0/1/\ldots/1$, for all i, j satisfying $1 \leq i \leq 3q$ and $1 \leq j \leq q$.
   Add arcs (a_{ij}, D_j) of cost $0/0/\ldots/0$, for all i, j satisfying $1 \leq i \leq 3q$ and $1 \leq j \leq q$.

In terms of the game structure, this corresponds to $qB + 3q^2$ players having origin s, B players having destination D_j, and one player having destination a_{ij}, for all i and j.

We claim that if the 3-Partition answer is 'yes,' then the congestion game cost is equal to $qB + 3q$; if the answer is 'no,' the congestion game cost is greater than or equal to $qB + 3q + 1$. To see the first implication, suppose the elements in S are $s_1, \ldots, s_{3q}$ and the

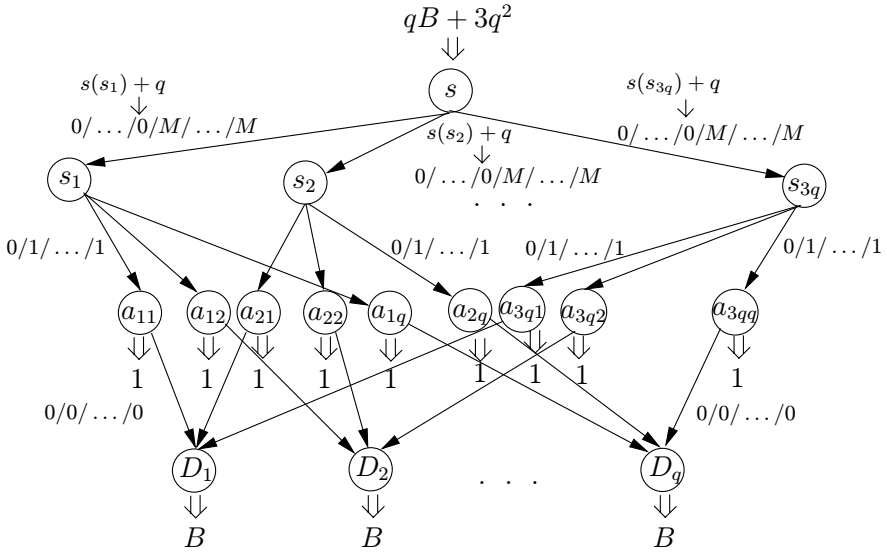

Figure 5-1: Constructed instance of the congestion game problem with nondecreasing arc costs.

sets in our 3-partition are $D_1, \ldots, D_q$. We construct a solution as follows. First, send 1 unit of flow along each of the paths $s - s_i - a_{ij}$, for all $i \in \{1, \ldots, 3q\}$ and $j \in \{1, \ldots, q\}$. Next, for all $s_i \in S$ such that $s_i \in D_j$, send $s(s_i)$ units of flow along the path $s - s_i - a_{ij} - D_j$. This solution will be feasible, since D is a 3-partition and thus the inflow at each node D_i will be equal to B. By inspection, the cost contributed by arc (s_i, a_{ij}) is equal to $s(s_i) + 1$ if $s_i \in D_j$, and 0 otherwise. Hence the total cost of the solution is $qB + 3q$.

To see the second implication, suppose there is a solution to the congestion game problem of cost at most $qB + 3q$. Because of the way the graph is constructed, this means that no node s_i has more than one unit of flow exiting along both the arcs (s_i, a_{ij}) and $(s_i, a_{ij'})$, for $j \neq j'$. This implies in particular that s_i must send $s(s_i) + 1$ units of flow along one of the arcs (s_i, a_{ij}), and consequently $s(s_i)$ units of flow travel from s_i to that corresponding node D_j. Consider the partition where each element s_i is mapped to the set D_j that it sends flow to in the solution. Each of the sets D_j will have size B, by the way the graph is constructed, and no element s_i will be mapped to more than one of the sets D_j. Hence this is a 3-Partition of the elements s_i. $\square$

We can easily extend this result to the symmetric version of the problem: add a new node t and new arcs (D_j, t) for all j. Set the cost of the new arcs to $0/0/\ldots/0/M/\ldots/M$, where the last '0' occurs in the B-th position. Adjust the demand so that the nodes D_j have demand 0 and the node t has a demand of qB. The same conclusions will hold.

A similar construction gives an even stronger result for single source games with nonincreasing arc costs. Recall that an α-approximation algorithm (see Section 2.2) is a polynomial-time algorithm that produces a feasible solution of cost within a factor of α of the optimum.

Theorem 5.2 *The single source network congestion game problem with nonincreasing arc costs is strongly NP-hard, and no approximation algorithm exists with a finite factor unless P=NP.*

Proof: Again we reduce from the 3-PARTITION problem. We use a simplified version of the construction in Theorem 5.1. We build a graph as follows (see Figure 5-2):

1. Create 1 source node s, with a supply of $qB + 3q$.
2. Create $3q$ sink nodes s_i, each with a demand of 1.
   Create q sink nodes D_j, each with a demand of B.
3. Add arcs (s, s_i) of cost $M/M/\ldots/M/0/\ldots/0$, for all i, where the first '0' occurs in the $(s(s_i)+1)$-st place, and M is a large number.
   Add arcs (s_i, D_j) of cost $M/M/\ldots/M/0/\ldots/0$, for all i, j, where the first '0' occurs in the $s(s_i)$-th place.

In terms of the game framework, this corresponds to $qB+3q$ players having origin s, 1 player having destination s_i, and B players having destination D_j, for all i and j.

If the answer to the 3-PARTITION problem is 'yes,' we can obtain a routing of cost 0 by sending $s(s_i) + 1$ units of flow from the source s to node s_i, and then routing $s(s_i)$ of those units from s_i to the node D_j corresponding to the set it is mapped to in the partition. Conversely, if the optimal solution to the constructed instance of the congestion

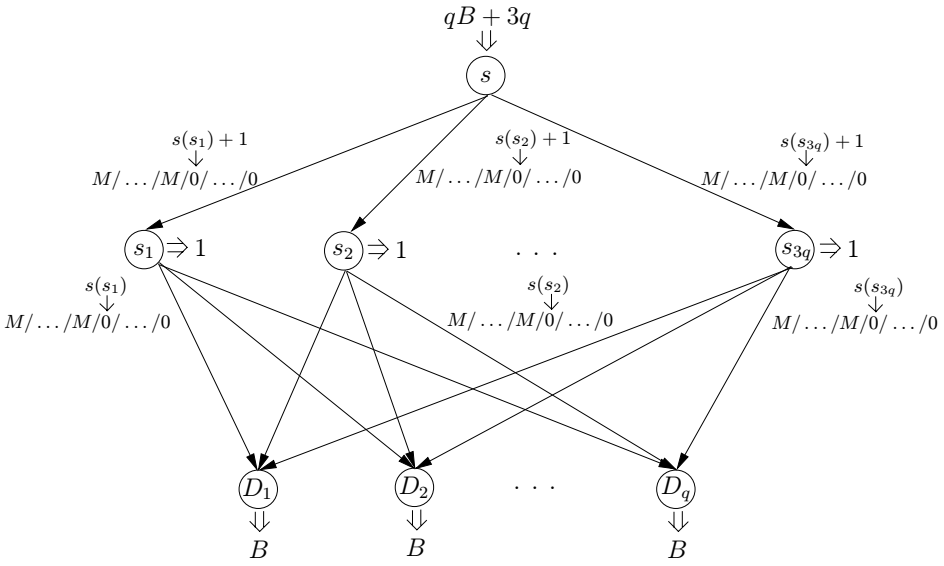

Figure 5-2: Constructed instance of the congestion game problem with nonincreasing arc costs.

game problem has cost 0, we can obtain a 3-partition of the elements s_i by placing each element s_i into the set D_j that it sends flow to in the congestion game solution. There will only be one such set, because of the cost structure, and each set D_j will have size B, due to the way the demands are defined.

This implies that our problem is NP-hard. Since M may be arbitrarily large, we observe that the gap between solutions corresponding to 'yes' and 'no' instances of the 3-Partition problem can be made arbitrarily large. This implies that no approximation algorithm can exist for the problem with a finite factor unless P=NP. □

The result can be extended to the symmetric problem with nonmonotonic arc costs: add a super-sink t as in the discussion following Theorem 5.1, and arcs (D_j, t) for all $j \in \{1, \ldots, q\}$. Set the arc costs to $0/\ldots/0/M/\ldots/M$, where the last '0' is in the Bth position. The identical conclusions follow.

The same argument does not apply to concave nonincreasing arc costs, but a simple reduction gives that this problem is NP-hard as well.

Theorem 5.3 *The single source network congestion game problem with concave nonincreasing arc costs is strongly NP-hard.*

Proof: We reduce from the DIRECTED STEINER TREE problem, which is strongly NP-complete [43]. This problem is:

Instance: A directed graph $G = (N, A)$ with weights $c_a \in \mathbb{Z}^+$ for all $a \in A$, a root node s, a set of terminals $\{t_1, t_2, \ldots, t_n\} \subseteq V$, and a bound $B \in \mathbb{Z}^+$.

Question: Does there exist a directed tree T rooted at node s, such that T contains an $s - t_i$ path for all $i = 1, \ldots, n$, and the sum of the combined arc costs in T is at most B?

Suppose we are given $G = (N, A)$, the node s, and terminals $t_1, \ldots, t_n$. We define an instance of the congestion game on G as follows. First, we assign one player to travel from node s to node t_i, for all $i \in \{1, \ldots, n\}$. Second, we set the cost of the arcs $a \in A$ equal to the concave nonincreasing function $c_a / \frac{c_a}{2} / \frac{c_a}{3} / \ldots / \frac{c_a}{n}$.

We claim that there is a solution to this single source congestion game problem of cost at most B if and only if there is a solution to the DIRECTED STEINER TREE problem of cost at most B. To see the first direction, suppose there exists a directed Steiner tree T of cost at most B. Since T is a Steiner tree, it must contain a path from s to t_i for all i. Consider a solution to the congestion game problem where we route all players from s to t_i using only arcs contained in T. This solution will be feasible since it contains a feasible path for every player, and its total cost will be at most B, by the way the costs in the congestion game are defined.

To see the other direction, suppose there exists a solution to the congestion game problem of cost at most B. The collection of arcs used in this solution must contain a path from s to t_i for all i, since the congestion game solution is feasible. Hence it must also contain a directed Steiner tree T. The cost of this tree will be at most B, by the way the congestion game costs are defined. Hence the congestion game problem is NP-hard. □

Another relatively simple reduction provides a strong hardness result for the asymmetric congestion game problem with convex nondecreasing costs.

Theorem 5.4 *The asymmetric network congestion game problem with convex nondecreasing arc costs is strongly NP-hard, and no approximation algorithm exists with a finite factor unless P=NP.*

Proof: We reduce from the ARC-DISJOINT PATHS problem, which is strongly NP-complete [43]. This problem is:

Instance: A directed graph $G = (N, A)$ and a set of node pairs $(s_1, t_1), \ldots, (s_n, t_n)$.

Question: Does there exist a collection of arc-disjoint paths $P_1, \ldots, P_n$, where P_i is an $s_i - t_i$ path?

Suppose we are given $G = (N, A)$ and $(s_1, t_1), \ldots, (s_n, t_n)$. We transform this into an instance of our problem as follows. First, we assign one player to travel from node s_i to node t_i, for all i. Second, for every arc $(i, j) \in A$, we introduce the cost structure $1/Mm/2Mm \ldots /(n-1)Mm$, where M is a large number and $m = |A|$. Each arc has n different labels, since n is the greatest number of players that can traverse an arc. Moreover, the cost function is convex, because the differences between consecutive arc costs are increasing.

If there exist arc-disjoint paths, then any routing using these paths will have cost m or less, since each arc will be taken at most once. Conversely, if there do not exist arc-disjoint paths, in any routing some arc will have to be taken twice, for a cost of at least Mm. Hence our problem is NP-hard; moreover, since M was arbitrary, we can make this gap as large as we want. This shows that the problem cannot be approximated within a finite factor unless P=NP. □

We have now covered all of the hardness results. We next address variants of the problem that are solvable in polynomial time.

Theorem 5.5 *The symmetric network congestion game problem with nonincreasing arc costs is solvable in polynomial time.*

Proof: Suppose we are given an instance of this problem, consisting of $G = (N, A)$, designated nodes s and t, costs on the arcs, and a collection of players $\{1, \ldots, n\}$. We first claim

that in such a problem, there exists an optimal solution where all players follow the same path from their origin to their destination.

To see this, suppose in a solution at least two players follow different paths. Let c_i denote the cost of the path followed by player i. Further suppose that among all the players, player k is following a path of minimal cost c_k. Now, consider rerouting all of the other players onto the path followed by player k. Since the arc costs are nonincreasing, the cost of this path will change to $c'_k \leq c_k$. The total cost of the solution will change to $nc'_k \leq nc_k \leq \sum_i c_i$. Hence there is some optimal solution where all players follow the same path.

In a solution where all players follow the same path, the cost of each arc a in the solution is equal to the cost $c_a(n)$ of routing n players across the arc. This suggests a simple algorithm for solving the problem: first, fix the cost of each arc $a \in A$ equal to $c_a(n)$; next, find the shortest $s - t$ path in G with respect to the new arc costs, and route all n players along this path. This gives a minimum cost solution to the problem where all players follow the same path, so it is optimal. □

We have one final complexity result, which relates to convex nondecreasing arc costs.

Theorem 5.6 *The single source network congestion game problem with convex nondecreasing arc costs is solvable in polynomial time.*

Proof: This result was independently proved by Chakrabarty et al. [19], though their proof was only stated for the symmetric case and linear costs. For completeness, we review the result here, noting that it also extends to the single source case.

Suppose we have an instance of the single source problem, which consists of a graph $G = (N, A)$ and a cost structure on the arcs. We give a reduction to the minimum cost flow problem. We create a new graph G' on the same node set N, where the arcs are defined as follows. For every arc $a \in A$ with cost structure $c_a(1)/c_a(2)/\ldots/c_a(n)$, we introduce n parallel arcs $a_1, a_2, \ldots, a_n$ in G' with the same head and tail nodes as a, where the cost of arc a_k is equal to $kc_k - (k-1)c_{k-1}$ and the capacity of each arc is 1.

We claim a minimum cost flow on G' gives a minimum cost flow on G, by setting the flow on $a \in A$ equal to the sum of the flows on the corresponding arcs in G'. To see this, first observe that there exists an integral minimum cost flow on G', since standard network flow problems always admit an integral optimal solution. Moreover, because the costs are convex and nondecreasing, in such an optimal solution any flow traveling across the parallel arcs $a_1, \ldots, a_n$ in G' will fill in order of increasing index (from 1 to n). This implies that if there are k units traveling across a set of parallel arcs, the corresponding cost will be

$$c_a(1) + (2c_a(2) - c_a(1)) + \ldots + (kc_a(k) - (k-1)c_a(k-1)) = kc_a(k).$$

Thus the cost structure in G' mimics that of G, and it follows that a minimum cost flow in G' gives an integral minimum cost flow in G. □

Fixed Number of Players We now comment on the complexity of the aforementioned problems with a fixed number of players. In this situation, we are given a set of players $\{1, 2, \ldots, n\}$, where n is a fixed constant. (Hence the running time of a polynomial-time algorithm may depend exponentially on n.)

In general congestion games, where the set of strategies is given explicitly, the congestion game problem with a fixed number of players can be solved in polynomial time (every player tries every strategy). In the case of network congestion games, however, there may be an exponential number of strategies: here the strategy space is compactly encoded, and the number of strategies can be exponential in the number of nodes and arcs in the network. Thus for this problem, the case with a fixed number of players is a nontrivial variant.

Several of our earlier results can be extended to apply to network congestion games with a fixed number of players. In particular, Theorem 5.4 holds for a fixed number of players, since the Arc-Disjoint Paths problem is NP-complete even for only two terminal pairs [38]. Similarly, Theorems 5.5 and 5.6 apply, since anything that can be solved in polynomial time with an arbitrary number of players can be solved in polynomial time with a fixed number of players.

We also observe that the single source network congestion game problem with concave nonincreasing arc costs can be solved in polynomial time for a fixed number of players. This is since we can model the problem as a minimum cost integer network flow problem with a concave cost function, by taking a piecewise linearization of the arc costs. (In other words, we create a continuous cost function for the problem by fitting a straight line between each two consecutive arc costs that are specified.) Such problems can be solved in polynomial time for fixed demand using the Send-and-Split method proposed by Erickson, Monma, and Veinott [31].

Undirected Network Congestion Games Another variant of the congestion game problem is that of network congestion games with undirected arcs. In this case, it turns out that most of our results are directly applicable, with one minor exception.

The polynomial algorithms extend directly to the undirected case, because the underlying problems of computing a shortest path and finding a minimum cost flow can be solved in polynomial time on undirected networks [5]. Similarly, the hardness result for asymmetric congestion game problems with convex nondecreasing costs is directly applicable because the Edge-Disjoint Paths problem is NP-complete [43]. The hardness construction for the single source problem with convex nondecreasing costs also extends directly, though some argument is needed to establish that the same properties hold.

The only case that does not apply directly is the hardness result for asymmetric congestion games with nonincreasing costs. Fortunately, the result for asymmetric congestion games with concave nonincreasing costs does extend, since the undirected Steiner Tree problem is also NP-complete, so it follows that the case with nonincreasing costs is still NP-hard. We lose the approximability gap in this case, but that is the only difference.

Weighted Network Congestion Games Another major variation of the problem is that of weighted congestion games. In this situation (see Section 2.3.3), each player i possesses $w_i \in \mathbb{Z}^+$ associated units of demand. (The unweighted case corresponds to $w_i = 1$ for all i.)

For weighted games, all of our hardness results apply, and in fact we can derive even stronger results as well. A simple reduction from the NP-complete Bin Packing problem [43] gives that the unsplittable weighted congestion game problem is NP-hard and no approximation algorithm exists with a finite factor, even in the case of symmetric games with convex nondecreasing costs. (In this case, players correspond to items; we have a graph with one source and one sink, and parallel arcs that represent bins. The cost structure on each arc is $0/0/\ldots/0/M/2M/3M\ldots$, where the last '0' is positioned to represent the capacity constraint of the bin; we can represent such costs compactly. The cost of the congestion game will be 0 if the bin packing answer is 'yes' and $\geq M$ if it is not.)

The only polynomial results that apply to this variant are for symmetric games with nonincreasing arc costs. Here, the players' weights do not affect the problem, since the optimal solution is to route all players on the same path. This gives all of the complexity results for this problem.

In k-Splittable Congestion Games We now comment on the complexity of obtaining a minimum cost solution in k-splittable congestion games. Further detailed in Chapter 6, a k-splittable congestion game is one in which each player may split their flow among at most k different paths. (Thus, the version we have been studying thus far corresponds to the case where $k = 1$.) In what follows, we assume that the arc cost function is explicitly given and that costs may differ for each increment of $\frac{1}{k}$ units of flow.

We observe that our hardness results in Theorems 5.1-5.3 extend to this version of the problem, by using the same reductions and modifying the cost functions appropriately. Theorem 5.4 also holds, by observing that even if players are allowed to split their flow into k paths, the only way to achieve a cost of less than M is if there are two arc-disjoint paths in the network. The polynomial algorithms in Theorems 5.5 and 5.6 extend as well, though in the case of convex nondecreasing arc costs we must ensure that the costs are convex and nondecreasing in every increment of $\frac{1}{k}$ units of flow. In weighted k-splittable congestion games, all of the results from the previous subsection apply, except for the nonapproximability reduction from the Bin Packing problem.

5.4 General Complexity Results

Our complexity results for general congestion games are given in Table 5.2. We note that since the set of strategies in general congestion games is given *explicitly* rather than *implicitly*, this problem is different from the network case and the same results do not immediately apply. As in the previous section, we first present the hardness results and then a polynomial time algorithm.

type→ costs ↓	symmetric	asymmetric
nondecreasing	NP-hard; inapprox.	NP-hard; inapprox.
convex nondecreasing	NP-hard; inapprox. [5.7]	NP-hard; inapprox.
nonincreasing	P [5.10]	NP-hard; inapprox. [5.8]
concave nonincreasing	P	NP-hard [5.9]
nonmonotonic	NP-hard; inapprox.	NP-hard; inapprox.

Table 5.2: Complexity results for general congestion games

(The numbers in brackets indicate in which theorem the results are proved. Note that the results for all unlabeled entries follow directly from other entries on the table. By 'NP-hard; inapprox.' we mean that the problem is NP-hard and no approximation algorithm exists with a finite factor, unless P=NP.)

Theorem 5.7 *The symmetric general congestion game problem with convex nondecreasing arc costs is strongly NP-hard, and no approximation algorithm exists with a finite factor unless P=NP.*

Proof: We reduce from the 3-Dimensional Matching problem, which is strongly NP-complete [43]. This problem is:

Instance: A set $S \subseteq X \times Y \times Z$, where X, Y, and Z are disjoint sets.

Question: Does S contain a subset $S' \subseteq S$ such that $|S'| = q$ and no two elements of S' agree in any coordinate?

Suppose we are given X, Y, Z, and S. We define an instance of the general congestion game problem as follows: let the members of X, Y, and Z correspond to the resources, and let each member $s \in S$ correspond to a potential strategy. (Thus, each strategy contains

three resources: one from X, one from Y, and one from Z.) Define q players, each of which possesses the same set of strategies S. Set the cost of each resource to $0/M/2M/\ldots/(q-1)M$, where M is a very large number. This cost function is convex, as the differences between consecutive arc costs are increasing.

We claim that if the answer to the 3-DIMENSIONAL MATCHING problem is 'yes,' then the optimal cost of this congestion game problem is 0; if the answer is 'no,' then the cost is $\geq M$. To see this, observe that a solution to the congestion game problem has cost 0 if and only if the strategies chosen by players in that solution constitute a matching. This is since the cost of any resource is prohibitively large if it is chosen more than once. Moreover, if there is no matching, some resource will have to be chosen more than once, for a cost of at least M. □

In the case of nonincreasing arc costs, our proof from the previous section carries over.

Theorem 5.8 *The asymmetric general congestion game problem with nonincreasing arc costs is strongly NP-hard, and no approximation algorithm exists with a finite factor unless P=NP.*

Proof: This follows directly from Theorem 5.2. Note that in our construction, there are a total of $3q^2$ possible strategies, which is polynomial in the input size. □

For concave nonincreasing costs, we give a somewhat different argument.

Theorem 5.9 *The asymmetric general congestion game problem with concave nonincreasing arc costs is strongly NP-hard.*

Proof: In the previous section, we showed that this problem is NP-hard for network congestion games. This implies that the problem can be hard in congestion games with an exponential number of strategies that are compactly encoded. We show a stronger result here, which is that the problem can be hard even in cases with only a polynomial number of strategies.

We reduce from the MINIMUM COVER problem, which is strongly NP-complete [43]. This problem is:

Instance: A finite set X, a collection S of subsets of X, and an integer $K \leq |S|$.

Question: Does S contain a subset $S' \subseteq S$ with $|S'| \leq K$, such that every element of X belongs to at least one member of S'?

Suppose we are given X, S, and K. We construct an instance of our problem as follows. Let the sets in S correspond to both the resources and the strategies, so that each strategy consists of one resource. Set the cost of each resource to $1/\frac{1}{2}/\frac{1}{3}/\ldots/\frac{1}{n}$, where $n = |X|$. Define n players, each corresponding to an element of X, and set the possible strategies associated with player $x \in X$ to be those sets $S_x \subseteq S$ containing element x.

We claim that the answer to the MINIMUM COVER problem is 'yes' if and only if the optimal cost of the congestion game is less than or equal to K. To see this, first note that each resource $s \in S$ costs the same regardless of how many players are using it. Thus the optimal solution to the congestion game corresponds to the smallest collection of sets that cover all the elements in X. It follows that if the optimal cost is less than or equal to K, then the corresponding instance is a 'yes' instance of the problem. Conversely, if there is a minimum cover of size less than or equal to K, we can obtain a solution to the congestion game of cost less than or equal to K by selecting for each element a strategy that contains it in the minimum cover. □

Finally, we have one polynomial-time algorithm.

Theorem 5.10 *The symmetric general congestion game problem with nonincreasing arc costs is solvable in polynomial time.*

Proof: By a similar argument to that in Theorem 5.5, we see that in any such problem it is optimal for all players to choose the same strategy. Hence we need only determine the cheapest strategy, where the cost of each resource $a \in A$ is set to $c_a(n)$. This can be done in polynomial time, because the number of strategies is part of the input. □

5.5 Concluding Remarks

We have provided the first extensive study of the complexity of finding optimal *minimum cost solutions* to congestion games, from a central perspective. For the most part, these problems are NP-hard, but we have identified several variants that are solvable in polynomial time. We examined a variety of different structural aspects and several different types of cost functions. We also touched on four variants of the problem, involving a *fixed number of players*, *undirected networks*, *varying amounts of player demand*, and *k-splittable games*. These problems are somewhat different in character, and there are several questions we leave open for future research.

We finally note that the Pup Matching problem (covered in Chapter 4) can be considered as a special case of a congestion game, where each arc has the cost function $c_a/\frac{c_a}{2}/\frac{2c_a}{3}/\frac{c_a}{2}/\ldots/\lceil\frac{n}{2}\rceil\cdot\frac{c_a}{n}$. Another problem that can be formulated as a congestion game is a generalization of the Pup Matching problem known as the Point-to-Point Delivery problem, studied in [61] and outlined in Section 4.1. In this case the cost function on every arc is $c_a/\frac{c_a}{2}/\ldots/\frac{c_a}{K}/\frac{2c_a}{K+1}/\ldots/\lceil\frac{n}{K}\rceil\cdot\frac{c_a}{n}$.

We have not yet addressed the approximability of those NP-hard problems contained in this paper for which inapproximability results were not obtained. This is an intriguing area for further study, as it may provide new insights into the structure and properties of the problem.

Chapter 6

Equilibria in k-Splittable Congestion Games

In this chapter, we apply the idea of k-splittable flows to network congestion games. We address the *existence*, *computability*, and *price of anarchy* of pure Nash equilibria in such games. We consider both weighted and unweighted versions of the problem (see Section 2.3) with directed networks and linear costs. We seek to discover whether the properties of k-splittable games are closer to those of splittable games or unsplittable games, and whether the answer depends on the value of k. This builds on the existing research by helping to bridge the gap between these separately examined cases.

6.1 Introduction

As outlined in Chapter 5, congestion games were first introduced in a paper of Rosenthal [73] as a simple class of games possessing pure-strategy Nash equilibria. In a network setting, congestion games can be used to model the effects of traffic congestion. In such a setting, players correspond to demands to be routed through the network, and the cost (latency) of each arc varies with the number of players traversing that arc. Typically, the number of players in such a network is either modeled as finite, each with a set amount of demand

(the *atomic* case) or infinite, each with an infinitesimal demand (the *nonatomic* case). In what follows we consider the atomic version of the problem. We address both the *unweighted* games, in which players all have unit demand, and *weighted* games (see Section 2.3.3), in which players possess an arbitrary amount of demand.

Rosenthal [73] showed that Nash equilibria are guaranteed to exist in unweighted network congestion games in which each player routes their flow *unsplittably*, via a potential function argument. (Here and from this point on we use the term 'Nash equilibrium' to refer to *pure* Nash equilibria, unless noted otherwise.) Monderer and Shapley [67] extended this argument, showing that Nash equilibria exist by presenting an exact potential function, which is a function that decreases every time an improving defection is made by an amount equal to the improvement in the defecting player's cost. In the weighted case, Fotakis, Kontogiannis, and Spirakis [39, 40] showed that equilibria always exist when there are linear costs, and they demonstrated an example showing that equilibria may not exist with arbitrary costs. Goemans, Mirrokni, and Vetta [45] furthered this result, showing that Nash equilibria may not exist even if the cost functions are quadratic.

If instead players are allowed to split their demand into an *arbitrary* number of paths, a result of Rosen [72] shows that Nash equilibria always exist if the total cost on each arc is a convex function. His argument does not extend to arbitrary cost functions, as it relies on a fixed-point theorem that requires the convexity of the function.

Much of the literature on congestion games has focused on finding the *price of anarchy* in various types of games. The price of anarchy (see Section 2.4) is defined as the ratio of the total cost of the worst Nash equilibrium to that obtained in an optimal solution. Koutsoupias and Papadimitriou [57] were the first to study this parameter, with respect to *mixed* Nash equilibria and in the special case of network congestion games consisting of two identical parallel links. Mavronicolas and Spirakis [63] extended these results to a more general case, and Czumaj and Vöcking [29] tightened the analysis, showing a bound of $\Theta(\frac{\log m}{\log \log m})$ in the case of identical links and $\Theta(\frac{\log m}{\log \log \log m})$ in general. A sizeable body of research has been devoted to such problems with parallel links, which is surveyed in [54].

For *pure* Nash equilibria and *unsplittable* flows, Christodoulou and Koutsoupias [21] and Awerbuch, Azar, and Epstein [10] recently and independently proved a bound of 2.5 in unweighted congestion games with linear costs. Awerbuch et al. [10] also gave a bound of 2.618 for unsplittable weighted congestion games with linear costs, and a bound of $d^{\Theta(d)}$ for weighted games with polynomial costs of degree d. Tight examples were given for all bounds.

With regard to *splittable* flows and linear costs, Cominetti, Correa, and Stier-Moses [23] proved an upper bound of 1.5 in both weighted and unweighted games and a lower bound of 1.343 in weighted games. In the unweighted case, an example of Roughgarden and Tardos [76] shows that the price of anarchy can be as large as 1.333.

Thus for *unweighted* games, the price of anarchy for (pure) Nash equilibria is 2.5 in the unsplittable case and between 1.333 and 1.5 in the splittable case. For *weighted* games, the price of anarchy is 2.618 in the unsplittable case and between 1.343 and 1.5 in the splittable case. Our motivation was to discover what happens in the intermediate cases– for instance, where the flow may be split onto 2 or 3 paths, but not an infinite number. In this respect we contribute to the body of literature on k-splittable flows, which is a growing topic in its own right.

The concept of k-splittable flows was first studied in a paper of Baier, Köhler, and Skutella [11], as an intermediate problem between splittable and unsplittable flows. In such a situation, each commodity may split its flow along a finite number k of different paths. Baier et al. showed that the maximum k-splittable flow problem is NP-hard, and they provided both approximability and nonapproximability results. Koch and Spenke [52] later extended this analysis with new complexity and approximability results.

The area of k-splittable flows has also been studied in the domain of scheduling, where tasks can be split into at most k parts. Shachnai and Tamir [80] were the first to study this topic, showing that the problem of finding a schedule of minimum makespan is NP-hard and providing an approximation algorithm. They were later followed by Krysta, Sanders, and Vöcking [58], who gave an exact approximation algorithm, and Agarwal et al. [2], who did an experimental study of k-splittable scheduling.

In what follows, we examine the questions of existence, computability, and the price of anarchy for Nash equilibria in *k-splittable congestion games*, which are network congestion games in which each player may split their flow onto at most k different paths (see Section 6.2). Our ultimate goal is to discover whether properties of k-splittable games are closer to those of splittable or unsplittable games, and whether the answer depends on the value of k.

With regard to *existence*, in Section 6.3 we show that Nash equilibria always exist in $\frac{1}{k}$-integral weighted network congestion games with linear costs. These are k-splittable games in which the flow by each player on each arc is restricted to being a multiple of $\frac{1}{k}$. We prove the result by providing an exact potential function for the problem, which decreases any time an improving defection is made. We extend the result to show that Nash equilibria always exist in k-splittable weighted network congestion games with linear costs, with the added condition that the flow on each arc must be a multiple of $\frac{1}{kM}$ for some M. This strongly suggests that Nash equilibria exist in general k-splittable network congestion games with linear costs, by taking the value of M to be arbitrarily large.

One by-product of this result is that we provide an exact potential function for unsplittable weighted network congestion games with linear costs. This contradicts a result of Fotakis et al. [39], who claim that no such potential function exists. A corollary of this result is that weighted network congestion games with linear costs are isomorphic to unweighted congestion games, as Monderer and Shapley [67] have shown that any game that admits an exact potential is isomorphic to an unweighted congestion game.

In terms of *computability*, we show in Section 6.4 that for all games with guaranteed existence of pure Nash equilibria, such an equilibrium may be computed in pseudopolynomial time. We also show that we can check whether a solution to a $\frac{1}{k}$-integral unweighted network congestion game is a Nash equilibrium in polynomial time. It is an open question whether this result can be extended to weighted network congestion games as well.

As for the *price of anarchy*, in Section 6.5 we give lower and upper bounds on the price of anarchy in k-splittable and $\frac{1}{k}$-integral network congestion games, for both weighted and unweighted versions of the problem. In $\frac{1}{k}$-integral games, we show that the lower bound

of 2.5 for the unweighted, unsplittable case [10, 21] carries over to the $\frac{1}{k}$-integral case as well, as does the lower bound of 2.618 [10] in the weighted case. In k-splittable games, we show a lower bound of $\frac{60k}{25k-1}$ in the unweighted case, and a bound of $\frac{32k\phi+24k}{9k\phi+17k-\phi-1}$ in the weighted case, where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. The first of these bounds tends to 2.4 as $k \to \infty$, and the second tends to 2.401 as $k \to \infty$.

With regard to upper bounds, we show an upper bound of 2.618 for all versions of the problem considered. This builds on the proof given by [10] in the weighted, unsplittable case. We strengthen the bound slightly for the special case of $\frac{1}{2}$-integral unweighted network congestion games, giving a bound of 2.6. We then show that the price of anarchy for k-splittable flows in a given instance need not be monotone with the value of k, and that the ratio of the cost of a worst Nash equilibrium to a *splittable* system optimum may also not be monotone with k. We show how several of our techniques can also be extended to the case of *undirected* network congestion games. We conclude by addressing areas for future research.

6.2 Problems Studied

We use the model of network congestion games defined in Section 2. Here we consider congestion games with *linear costs*, in which the cost of sending x_a units of flow along arc $a \in A$ is given by the linear function $c_a(x_a) = q_a(x_a) + r_a$, for some nonnegative q_a and r_a.

We investigate several different variants with regard to the structure of the paths taken by each player. In an *unsplittable* network congestion game, players are restricted to route all of their flow along a single path. More generally, in a *k-splittable* network congestion game each player may route their flow along at most k paths. If there is no restriction on the number of paths a player can take, we call the game *splittable* or *infinitely splittable*.

Another variant we consider is that of *$\frac{1}{k}$-integral* network congestion games. These are k-splittable congestion games in which the flow on each arc by each player is restricted to being a multiple of $\frac{1}{k}$. These games are more tractable than k-splittable games, because there are only a finite number of possible flow values on each arc. A special case of $\frac{1}{k}$-integral games is that of *integer splittable* games, in which each player must route their flow integrally

on each arc. (To translate from an integral splittable game to a $\frac{1}{k}$-integral game, divide the demand in the integer splittable instance by the maximum player demand D.)

This gives us a number of different problem variants, according to the degree of splittability and whether we are considering a k-splittable or a $\frac{1}{k}$-integral version of the probem. In the next section, we address the existence of equilibria in such games.

6.3 Existence of Nash Equilibria

We claim that Nash equilibria always exist in the following congestion games:

- $\frac{1}{k}$-integral weighted network congestion games with linear costs
- k-splittable weighted network congestion games with linear costs, where flow values on all arcs are multiples of $\frac{1}{kM}$ for some M.

These results are both proved by exhibiting a potential function that decreases each time an improving defection is made. To motivate these proofs, we first show the existence of Nash equilibria for two simpler games, and we then generalize to the broader cases.

Theorem 6.1 *A Nash equilibrium always exists in unsplittable weighted network congestion games with demand equal to 1 or 2 per player and linear costs.*

Proof: Suppose we are given an instance of such a game with costs $c_a(x_a) = q_a x_a + r_a$ for all $a \in A$, and a solution x. Let x_a be the total amount of flow on arc a under x, and let $x_a^{\leq i}$ be the amount of flow up to and including player i on arc a. Also, let X_{i_ℓ} be the set of arcs on which player i sends ℓ units of flow in x. Further suppose that the order of the players is arbitrary. Consider the potential function:

$$\Phi(x) = \sum_a \sum_{j=1}^{x_a} c_a(j) + \sum_{i=1}^{n} \sum_{a \in X_{i_2}} q_a.$$

This is similar to the potential function given in [33, 67] (based on the proof in [73]) for the existence of Nash equilibria in unweighted network congestion games, with the notable addition of the second term. Furthermore, since

$$\sum_{a\in A}\sum_{j=1}^{x_a} c_a(j) = \sum_{i=1}^{n}(\sum_{a\in X_{i_1}} c_a(x_a^{\leq i}) + \sum_{a\in X_{i_2}} [c_a(x_a^{\leq i}-1) + c_a(x_a^{\leq i})]),$$

we can rewrite the potential as:

$$\Phi(x) = \sum_{i=1}^{n}\sum_{a\in X_{i_1}} c_a(x_a^{\leq i}) + \sum_{i=1}^{n}\sum_{a\in X_{i_2}} [c_a(x_a^{\leq i}-1) + c_a(x_a^{\leq i}) + q_a].$$

This is since any time we send one unit of flow on an arc, its contribution is counted once in the potential; when we send two units of flow, it is counted twice for different amounts. The q_a term serves to 'correct' for the discrepancy in the cost when two units of flow are sent on an arc.

Now, suppose (x, x') is an improving defection; i.e., exactly one player changes their strategy and achieves a lower cost as a result. Without loss of generality, we can assume that player n defects, since the order of the players is arbitrary. Define x'_a, $x'^{\leq i}_a$, and X'_{i_ℓ} analogously to before. Then:

$$\begin{aligned}
\Phi(x') - \Phi(x) &= \sum_{a\in X'_{n_1}} c_a(x'^{\leq n}_a) + \sum_{a\in X'_{n_2}} [c_a(x'^{\leq n}_a - 1) + c_a(x'^{\leq n}_a) + q_a] \\
&- (\sum_{a\in X_{n_1}} c_a(x_a^{\leq n}) + \sum_{a\in X_{n_2}} [c_a(x_a^{\leq n}-1) + c_a(x_a^{\leq n}) + q_a]) \\
&= \sum_{a\in X'_{n_1}} c_a(x'_a) + \sum_{a\in X'_{n_2}} [c_a(x'_a - 1) + c_a(x'_a) + q_a] \\
&- (\sum_{a\in X_{n_1}} c_a(x_a) + \sum_{a\in X_{n_2}} [c_a(x_a - 1) + c_a(x_a) + q_a]) \\
&= \sum_{a\in X'_{n_1}} c_a(x'_a) + \sum_{a\in X'_{n_2}} [q_a(x'_a - 1) + r_a + q_a(x'_a) + r_a + q_a] \\
&- (\sum_{a\in X_{n_1}} c_a(x_a) + \sum_{a\in X_{n_2}} [q_a(x_a - 1) + r_a + q_a(x_a) + r_a + q_a]) \\
&= \sum_{a\in X'_{n_1}} c_a(x'_a) + \sum_{a\in X'_{n_2}} 2\cdot c_a(x'_a) - \sum_{a\in X_{n_1}} c_a(x_a) - \sum_{a\in X_{n_2}} 2\cdot c_a(x_a) \\
&= C_n(x') - C_n(x) < 0.
\end{aligned}$$

The second equality follows from the definition of x_a and x'_a, and the third equality follows by substituting in for the cost function. Additionally, recall from Chapter 2 that $C_n(x)$ is the cost experienced by player n in solution x. This result implies that Φ is an

exact potential for this game, since any time a player defects, the change in Φ is equal to the change in cost. Thus a Nash equilibrium is guaranteed to exist. $\square$

We can also extend this argument to the case of arbitrary demand.

Theorem 6.2 *A Nash equilibrium always exists in unsplittable weighted network congestion games with linear costs.*

Proof: Suppose we are given an instance of such a game, where the maximum demand of any player is D. Suppose we are also given a solution x. Define $x_a, x_a^{\leq i}$, and X_{i_ℓ} as in Theorem 6.1. Consider the potential function:

$$\Phi(x) = \sum_a \sum_{j=1}^{x_a} c_a(j) + \sum_{i=1}^{n} \sum_{\ell=1}^{D} \sum_{a \in X_{i_\ell}} \frac{\ell^2 - \ell}{2} q_a.$$

This is similar to the potential function proposed in Theorem 6.1, though the term on the right hand side has been expanded. Now, when we send ℓ units of flow on an arc, its contribution is counted ℓ times in the first term and once in the second term. Again the second term serves to 'correct' the for discrepancy in the cost of sending $\ell > 1$ units of flow.

As before, we can rewrite this potential function as:

$$\Phi(x) = \sum_{i=1}^{n} \sum_{\ell=1}^{D} \sum_{a \in X_{n_\ell}} \left[\sum_{j=0}^{l-1} c_a(x_a^{\leq i} - j) + \frac{\ell^2 - \ell}{2} q_a \right].$$

Now, suppose that (x, x') is an improving defection. Without loss of generality, we can assume that player n defects. Using the same arguments as last time, we obtain:

$$\begin{aligned}
\Phi(x') - \Phi(x) &= \sum_{\ell=1}^{D} \sum_{a \in X'_{n_\ell}} \left[\sum_{j=0}^{\ell-1} c_a(x'_a - j) + \frac{\ell^2 - \ell}{2} q_a \right] - \sum_{\ell=1}^{D} \sum_{a \in X_{n_\ell}} \left[\sum_{j=0}^{\ell-1} c_a(x_a - j) + \frac{\ell^2 - \ell}{2} q_a \right] \\
&= \sum_{\ell=1}^{D} \sum_{a \in X'_{n_\ell}} \left[\sum_{j=0}^{\ell-1} q_a(x'_a - j) + r_a + \frac{\ell^2 - \ell}{2} q_a \right] - \sum_{\ell=1}^{D} \sum_{a \in X_{n_\ell}} \left[\sum_{j=0}^{\ell-1} q_a(x_a - j) + r_a + \frac{\ell^2 - \ell}{2} q_a \right] \\
&= \sum_{\ell=1}^{D} \sum_{a \in X'_{n_\ell}} \ell \cdot c_a(x'_a) - \sum_{\ell=1}^{D} \sum_{a \in X_{n_\ell}} \ell \cdot c_a(x_a) \\
&= C_n(x') - C_n(x) < 0.
\end{aligned}$$

In the second equality, we have substituted for $c_a(x_a)$; in the third equality, we used the fact that $\sum_{j=0}^{\ell-1} j = \frac{\ell^2-\ell}{2}$. This result implies that Φ is an exact potential for the game. Hence, a Nash equilibrium exists. □

Our proof of this result is stronger than the Fotakis et al. [39] proof of the same result, in that we show an *exact* potential exists and they show only a b-potential exists in such games. Moreover, Theorem 6.2 also contradicts a result of Fotakis et al. [39], who claim to have a proof that no such exact potential function exists in unsplittable weighted network congestion games. Finally, our proof also provides us with the following corollary, which is of theoretical significance.

Corollary 6.3 *Unsplittable weighted network congestion games with linear costs are isomorphic to unsplittable unweighted congestion games.*

Proof: In the previous theorem, we observed that all unsplittable weighted congestion games with linear costs possess an exact potential function. Monderer and Shapley [67] showed that all games that possess an exact potential function are isomorphic to unsplittable unweighted congestion games. (By 'isomorphic,' we mean that strategies in one game correspond to strategies in the other game with the same cost.) The result follows. □

With regard to $\frac{1}{k}$-integral games, we need only one modification to extend the result of Theorem 6.2 to this case.

Theorem 6.4 *A Nash equilibrium always exists in $\frac{1}{k}$-integral weighted network congestion games with linear costs.*

Proof: Suppose we are given an instance of such a game with maximum demand D, and suppose we are also given a solution x. Define $x_a, x_a^{\leq i}$, and X_{i_ℓ} as before, and assume the value of k is given.

The idea behind this proof is very similar to that in Theorems 6.1 and 6.2. The main difference is that previously the first term in our potential function incremented with every

unit of flow; our new potential function will increment for every $\frac{1}{k}$ units of flow. Accordingly, the potential is:

$$\Phi(x) = \frac{1}{k} \sum_a \sum_{j=1}^{kx_a} c_a\big(\frac{j}{k}\big) + \sum_{i=1}^{n} \sum_{\ell=1}^{Dk} \sum_{a \in X_{i\,\frac{\ell}{k}}} \frac{\ell^2 - \ell}{2k^2} q_a.$$

In this potential function, when we send ℓ units of flow on an arc, its contribution is counted $\ell \cdot k$ times in the first term. The second term, as before, 'corrects' for the cost discrepancy.

Once again, we rewrite the function:

$$\Phi(x) = \sum_{i=1}^{n} \sum_{\ell=1}^{Dk} \big[\sum_{a \in X_{i\,\frac{\ell}{k}}} \big(\frac{1}{k} \sum_{j=0}^{\ell-1} c_a\big(x_a^{\leq i} - \frac{j}{k}\big)\big) + \frac{\ell^2 - \ell}{2k^2} q_a \big].$$

Similar algebra to that in Theorems 6.1 and 6.2 shows that for any improving defection (x, x'):

$$\begin{aligned} \Phi(x') - \Phi(x) &= \sum_{\ell=1}^{Dk} \sum_{a \in X'_{n\,\frac{\ell}{k}}} \frac{\ell}{k} \cdot c_a(x'_a) - \sum_{\ell=1}^{Dk} \sum_{a \in X_{n\,\frac{\ell}{k}}} \frac{\ell}{k} \cdot c_a(x_a) \\ &= C_n(x') - C_n(x) < 0. \end{aligned}$$

Hence again Φ is an exact potential for the game, and a Nash equilibrium exists. □

Our last result concerns k-splittable weighted network congestion games, in which the flow by each player on an arc must be a multiple of $\frac{1}{kM}$, for some value of M. In other words, these are k-splittable weighted congestion games in which the flow is restricted to take a (possibly very large) finite set of values, and in which the flow values on each arc may be extremely small (in contrast to being integral). Also in this more general case we can show that Nash equilibria always exist.

Theorem 6.5 *A Nash equilibrium always exists in k-splittable weighted network congestion games with linear costs, in which the flow on each arc must be a multiple of $\frac{1}{kM}$ for some M.*

Proof: We again exhibit a potential function. The only difference between the potential function for this game and that of Theorem 6.4 is that instead of incrementing the potential function for every $\frac{1}{k}$ units of flow, we now increment the potential for every $\frac{1}{kM}$ units of flow. The potential is:

$$\Phi(x) = \frac{1}{kM} \sum_{a} \sum_{j=1}^{kMx_a} c_a\left(\frac{j}{kM}\right) + \sum_{i=1}^{n} \sum_{\ell=1}^{DkM} \sum_{a \in X_{i\,\frac{\ell}{kM}}} \frac{\ell^2 - \ell}{2(kM)^2} q_a.$$

Note that we have merely substituted kM for k in the potential function from Theorem 6.4. This is valid because nowhere in the proof of Theorem 6.4 did we explicitly rely on the fact that the flow was k-splittable. The only fact we used was that the flow on each arc is a multiple of $\frac{1}{k}$. In our new game, we are given the fact that the flow on each arc is a multiple of $\frac{1}{kM}$, so the same arguments apply.

By the observation above, the remainder of the proof of Theorem 6.4 applies to these games as well. Hence Nash equilibria are guaranteed to exist. □

This last result suggests that Nash equilibria exist in general for k-splittable weighted games, by allowing the value of M to become arbitrarily large. It should be noted that the standard technique for proving the existence of Nash equilibria in the infinitely splittable case cannot be applied to this problem, since the feasible set is not necessarily convex.

6.4 Computability of Nash Equilibria

With regard to computability, we have two main results. The first concerns the complexity of finding a pure Nash equilibrium, for the games discussed in the previous section. The second addresses whether we can check that a given solution constitutes a Nash equilibrium in polynomial time.

Theorem 6.6 *For all congestion games in which we showed a Nash equilibrium exists (Theorems 6.1-6.5), we can compute such an equilibrium in pseudopolynomial time.*

Proof: All of our existence results in the previous section were proved by exhibiting a potential function that decreases (by at least a certain set amount) with every improving player defection. The potential function values are pseudopolynomial in size, as it grows polynomially with the number of players, the splittability factor, the arc costs, the demand, and (where applicable) the value of M. Thus minimizing the potential function via best-response moves gives a pseudopolynomial-time algorithm for these congestion games. □

For the problem of checking whether a given solution is a Nash equilibrium, we obtain the following result.

Theorem 6.7 *We can check whether a given solution to a $\frac{1}{k}$-integral unweighted congestion game with convex and nondecreasing costs is a Nash equilibrium in time polynomial in k.*

Proof: Suppose we are given a solution x to this game, and suppose all arc costs are convex and nondecreasing. For every player i, we would like to verify that i has no incentive to deviate. We show how this operation may be performed for each player in polynomial time.

For a given player i, let x^{-i} be the (partial) solution obtained when all players are fixed to the same paths as in x, except for player i which is unassigned. For each arc $a \in A$, add k new arcs as follows: for every arc $a = (u, v)$ that now carries x_a^{-i} total units of flow, replace arc a by the construction in Figure 6-1.

Here, arc a has turned into k arcs, each with capacity $\frac{1}{k}$. It can be verified that the arc costs are nondecreasing from the first to the kth arc, since the cost function $c_a(x)$ is

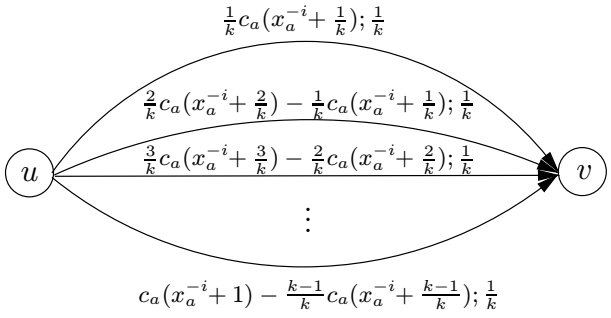

Figure 6-1: Replacement construction for arc $a = (u, v)$

convex and nondecreasing.

Next, solve a minimum cost flow problem on this new network, where the objective is to send one unit of flow from s_i to t_i as cheaply as possible. We claim player i will have an incentive to change its strategy if and only if the minimum cost flow on this network is cheaper than the cost assigned to player i in solution x.

To prove this claim, we first observe that flows in the original $\frac{1}{k}$-integral game correspond to flows in the new network with the same cost. If player i sends $\frac{j}{k}$ units of flow on arc a in the original instance, then we simply require player i to use the top j (cheapest) of the parallel arcs in the new formulation. The cost to player i for using these arcs is

$$\frac{j}{k} c_a(x_a^{-i} + \frac{j}{k}),$$

since the arc costs form a telescoping sum. This is the same as the cost associated with taking arc a in the original formulation.

We then observe that since all the capacities in the network flow problem are $\frac{1}{k}$-integral and the parallel arcs are always taken in order of nondecreasing cost, the minimum cost flow will be $\frac{1}{k}$-integral as well, with the same cost as that in the original congestion game.

Together, this implies that if any solution to the minimum cost flow problem gives a lower cost than that which is currently experienced by player i in the congestion game, then there is a solution in which player i can deviate and attain a better overall cost.

Finally, observe that the running time of the checking algorithm is polynomial in the network size and the degree of splittability. Hence we can check whether a solution is a Nash equilibrium in time polynomial in k and the size of the network. □

As a final note, we observe that this approach does not extend to k-splittable unweighted network congestion games with flow values that are multiples of $\frac{1}{kM}$, or to weighted congestion games. This is since we cannot guarantee in these cases that the solution returned by the minimum cost flow problem will use *at most* k paths.

6.5 Price of Anarchy

We obtain the following bounds on the price of anarchy in network congestion games with linear costs. Previously shown results are indicated with a reference to the paper in which they are proved. Recall that the price of anarchy is the ratio of the cost of a worst-case Nash equilibrium to that of an optimal solution.

	$\frac{1}{k}$-Integral			k-Splittable	
problem	lower bound	upper bound	problem	lower bound	upper bound
1-integral	$2.5^{[10,\ 21]}$	$2.5^{[10,\ 21]}$	1-splittable	$2.5^{[10,\ 21]}$	$2.5^{[10,\ 21]}$
$\frac{1}{2}$-integral	2.5	2.6	2-splittable	2.449	2.618
$\frac{1}{3}$-integral	2.5	2.618	3-splittable	2.432	2.618
$\frac{1}{4}$-integral	2.5	2.618	4-splittable	2.424	2.618
$\frac{1}{5}$-integral	2.5	2.618	5-splittable	2.419	2.618
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\frac{1}{k}$-integral	2.5	2.618	k-splittable	$\frac{60k}{25k-1}$ *	2.618
			∞-splittable	$1.333^{[76]}$	$1.5^{[23]}$

$* \to 2.4$ as $k \to \infty$

Table 6.1: Price of anarchy in unweighted network congestion games

	$\frac{1}{k}$-Integral			k-Splittable	
problem	lower bound	upper bound	problem	lower bound	upper bound
1-integral	$2.618^{[10]}$	$2.618^{[10]}$	1-splittable	$2.618^{[10]}$	$2.618^{[10]}$
$\frac{1}{2}$-integral	2.618	2.618	2-splittable	2.505	2.618
$\frac{1}{3}$-integral	2.618	2.618	3-splittable	2.469	2.618
$\frac{1}{4}$-integral	2.618	2.618	4-splittable	2.451	2.618
$\frac{1}{5}$-integral	2.618	2.618	5-splittable	2.441	2.618
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\frac{1}{k}$-integral	2.618	2.618	k-splittable	$\frac{32k\phi+24k}{9k\phi+17k-\phi-1}$ **	2.618
			∞-splittable	$1.343^{[23]}$	$1.5^{[23]}$

$** \to 2.401$ as $k \to \infty$

Table 6.2: Price of anarchy in weighted network congestion games

6.5.1 Lower Bounds on the Price of Anarchy

Unweighted Network Congestion Games

Theorem 6.8 *For any value of k, there are instances of $\frac{1}{k}$-integral unweighted network congestion games with linear costs in which the price of anarchy is at least 2.5.*

Proof: Suppose we are given a value of k. We construct an example of a $\frac{1}{k}$-integral unweighted congestion game with price of anarchy equal to 2.5. Our example is an extension to an example that Christodoulou and Koutsoupias [21] give to show a price of anarchy result for the unsplittable case. Their example is as follows:

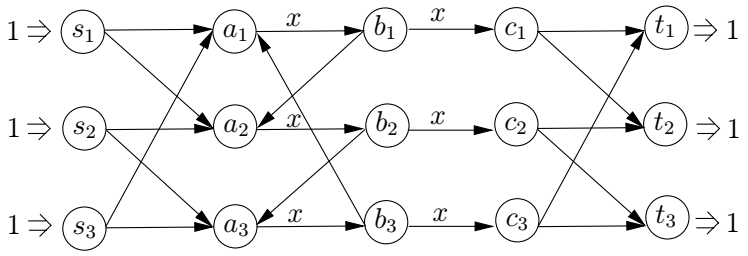

Figure 6-2: Christodoulou and Koutsoupias example

Here there are three players, each with source s_i and sink t_i, and unsplittable flows. The costs of all unlabeled arcs are 0. The system optimal solution is for all players to traverse the straight path from their source to their sink (in which player i follows path $s_i - a_i - b_i - c_i - t_i$), with a total cost of 6.

A Nash equilibrium arises when each player traverses a 'crooked' path from source to sink, as follows: player 1 takes $s_1 - a_2 - b_2 - a_3 - b_3 - c_3 - t_1$, player 2 takes $s_2 - a_3 - b_3 - a_1 - b_1 - c_1 - t_2$, and player 3 takes $s_3 - a_1 - b_1 - a_2 - b_2 - c_2 - t_3$. The total cost of this solution is 15.

In the following, we extend this example to the case of $k > 1$. Create k copies of the Christodoulou-Koutsoupias graph. Add super sources S_1, S_2, and S_3 and connect source S_i $(i = 1, 2, 3)$ to all the k copies of s_i in the original graph. Similarly, add super sinks T_1, T_2, and T_3, where sink T_i $(i = 1, 2, 3)$ connects to all copies of t_i in the original graph. For $k = 2$, this gives the graph shown in Figure 6-3.

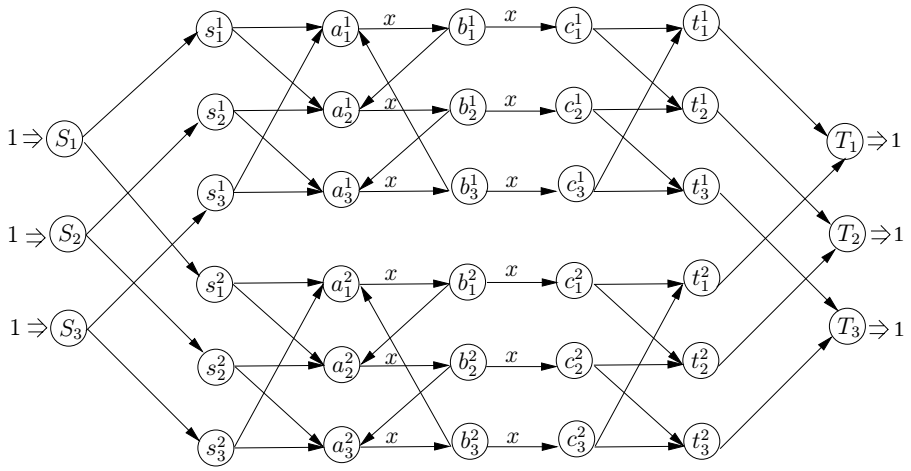

Figure 6-3: Expansion of Christodoulou and Koutsoupias graph for $k = 2$

An optimal solution is for each player to send $\frac{1}{k}$ units of flow along each of the 'straight' $S_i - s_i^j - a_i^j - b_i^j - c_i^j - t_i^j - T_i$ paths. In the case of $k = 2$, this yields:

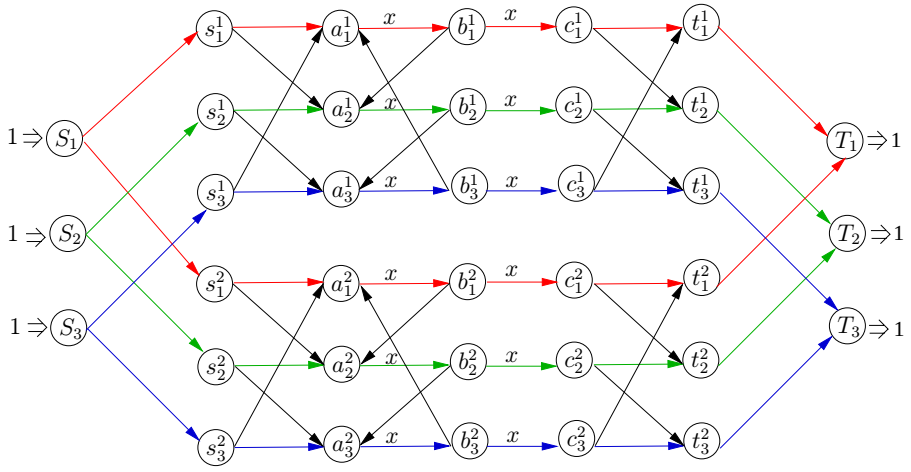

Figure 6-4: Optimal solution for $k = 2$

This solution is optimal since the flow from each player must traverse at least two arcs of cost x on each path it uses from source to sink, and the cheapest way for it to do so is to split the flow into k paths. The total cost of this solution is $\frac{6}{k}$, since each player experiences a cost of $\frac{2}{k}$.

We claim that the solution where each player sends $\frac{1}{k}$ units of flow along each of the 'crooked' paths is a Nash equilibrium. In the case of $k = 2$, the situation is as follows:

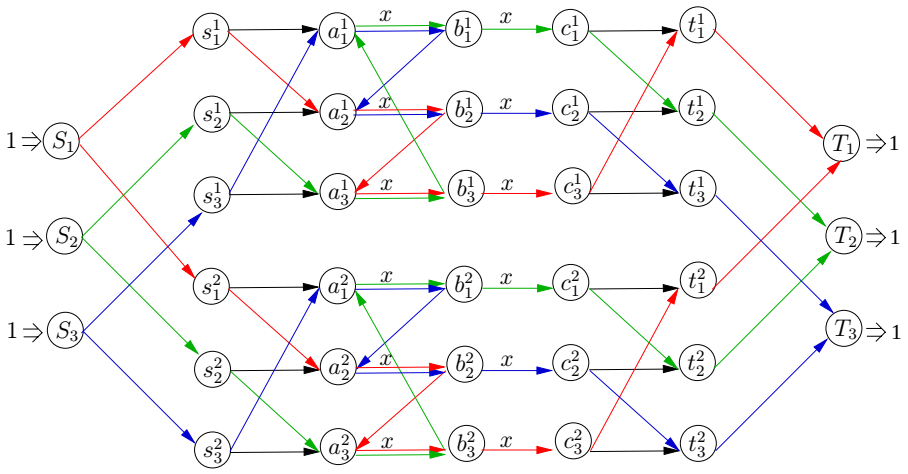

Figure 6-5: Nash equilibrium for $k = 2$

To verify that this solution is a Nash equilibrium, we first claim that in any Nash equilibrium each player will send $\frac{1}{k}$ units of flow into every copy of Christodoulou-Koutsoupias graph. To see this, first observe that it is always advantageous for a player to send its flow along as many paths as possible, since the cost accrued by each player is a quadratic function of the amount of flow that it places on each path. Thus, given that each player sends $\frac{1}{k}$ units of flow along k different paths, it is then beneficial to split the flow evenly among the k different copies of the graph.

Once the players have each sent $\frac{1}{k}$ units of flow to each of the k copies of the graph, the fact that our solution is a Nash equilibrium follows from the fact that the solution restricted to each copy is a Nash equilibrium, and the cost of each player in each copy is the same. In other words, the proof by Christodoulou and Koutsoupias [21] in the unsplittable case applies to each copy, implying that the entire overall solution is a Nash equilibrium.

The total cost associated with this Nash equilibrium is $\frac{15}{k}$, since the cost associated with each player is $\frac{5}{k}$. Hence the price of anarchy for this example is at least 2.5. □

In the k-splittable case, we derive a different lower bound on the price of anarchy. Note that although $\frac{1}{k}$-integral games are a special case of k-splittable games, the bounds on the price of anarchy do not necessarily carry over because Nash equilibria in $\frac{1}{k}$-integral games are not always Nash equilibria in the k-splittable case. In particular, in k-splittable games the flow on each arc can be *arbitrary*, so there are a far greater number of possible deviations for each player.

Theorem 6.9 *For any value of k, there are k-splittable unweighted network congestion game instances with linear costs in which the price of anarchy is at least $\frac{60k}{25k-1}$.*

Proof: We use the same example as in the previous proof, with one alteration. A linear cost of Bx is added each of the arcs (s_j^i, a_j^i), for all i and j, so that each of the subgraphs is as pictured in Figure 6-6.

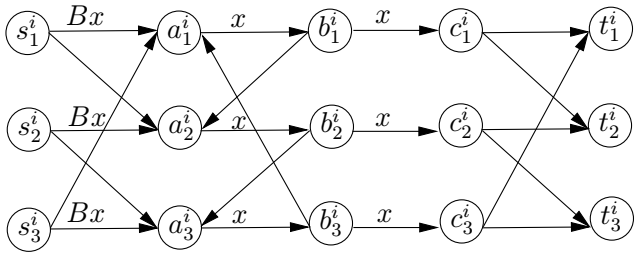

Figure 6-6: Alteration of the basic graph

The value of B is chosen such that $B = \frac{k-1}{12k}$.

We claim that the solution where players send $\frac{1}{k}$ units of flow along each of the straight paths is still optimal. To see this, observe that the costs of the arcs (s_j^i, a_j^i) are very small compared to the costs of the arcs (a_j^i, b_j^i) and (b_j^i, c_j^i), for all i and j. This implies that adding the new costs does not affect the optimality of the solution. The total cost associated with this solution is $\frac{6+3B}{k}$.

We also claim the solution where all players send $\frac{1}{k}$ units of flow along each of the crooked paths is still a Nash equilibrium. To show this, we will prove that no player p has an incentive to deviate. We begin by supposing that a player p deviates, and we show that it cannot achieve a lower total cost than it is currently experiencing.

First, note that if player p deviates to any path P that is not a 'straight' path or a 'crooked' path, we can verify that either the 'straight' path or the 'crooked' path in the same subgraph will contain a strict subset of the arcs of cost x that are used in P, and no additional arcs of cost x. (It may contain an arc of cost Bx, but this is negligible compared to the cost of x.) Hence it suffices to consider only the 'straight' paths and 'crooked' paths in each subgraph.

We next claim that if player p wishes to send y units of flow along j paths, which are either all straight or all crooked and have the same current flow in the considered solution, then the minimum cost occurs when p sends $\frac{y}{j}$ units of flow along each path. This fact can easily be shown; because the total cost of each path is a quadratic function of the flow on that path, the minimum cost solution is to distribute the total flow as evenly as possible.

Using this fact, if player p sends y units of flow along i straight paths, and $1-y$ units of flow along $k-i$ crooked paths, then the minimum cost solution is obtained when $\frac{y}{i}$ units of flow are routed along each straight path and $\frac{1-y}{k-i}$ units of flow are routed along each crooked path. Let $\frac{y}{i} = \Delta$. If player p sends Δ units of flow along each of i straight paths and $\left(\frac{1-\Delta i}{k-i}\right)\Delta$ units of flow along $k-i$ crooked paths, then it can be verified that the cost associated with this solution is

$$\frac{(2ik^2 + i^2k + ik(k-i)B)\Delta^2) - (5ik + i^2)\Delta + 5k - 2i}{k(k-i)}.$$

For our solution to be a Nash equilibrium, we need this quantity to be greater than or equal to $\frac{5}{k}$, which is the total cost associated with each player in the solution. Setting the above formula greater than or equal to $\frac{5}{k}$ and solving for B, we obtain:

$$B \geq \frac{k-i}{12k}.$$

This implies that the strongest restriction on B occurs when we send flow on 1 straight path and $k-1$ crooked paths, giving an overall bound of

$$B \geq \frac{k-1}{12k},$$

which is the same value that we selected earlier. Hence our solution is a Nash equilibrium. The price of anarchy here is

$$\frac{\frac{15}{k}}{\frac{6+3B}{k}} = \frac{60k}{25k-1}. \quad \square$$

Weighted Network Congestion Games

Theorem 6.10 *For any value of k, there are instances of $\frac{1}{k}$-integral weighted network congestion games with linear costs in which the price of anarchy is at least 2.618.*

Proof: We use the same proof technique as in Theorem 6.8, this time starting with the Awerbuch, Azar, and Epstein [10] example for the 2.618 bound in the unsplittable case. Their example is:

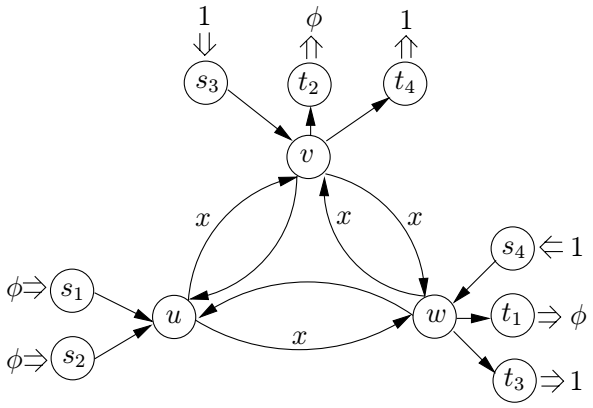

Figure 6-7: Awerbuch, Azar, and Epstein example

Here, $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio, and all unlabeled arc costs are 0. An optimal solution is for each player to take the most direct 'short' path from source to sink, using only one of the inner arcs (i.e, $s_1 - u - w - t_1$). The cost of this solution is $2\phi^2 + 2$. A Nash equilibrium arises when each player takes the less direct 'long' path from source to sink, using two of the inner arcs (i.e., $s_1 - u - v - w - t_1$). The cost of this solution is $4\phi^2 + 4\phi + 2$, which gives that the price of anarchy is at least $\frac{4\phi^2+4\phi+2}{2\phi^2+2} = \phi + 1 \approx 2.618$.

We now extend this example to the case of $k > 1$. We first create k copies of the graph above. Next, we add super sources S_1, S_2, and S_3 and connect source S_i $(i = 1, 2, 3)$

to all the copies of s_i in the original graph. We also add super sinks T_1, T_2, and T_3, where sink T_i $(i = 1, 2, 3)$ connects to all copies of t_i in the original graph.

We claim that the solution where each player sends $\frac{1}{k}$ units of flow along the short path in each of the copies is an optimal solution, and the solution where each player sends $\frac{1}{k}$ units of flow along the long path in each of the copies is a Nash equilibrium. To see that sending $\frac{1}{k}$ units of flow along the short paths is optimal, we observe that each player must traverse at least one arc of cost x in the path from their source to their sink; as in Theorem 6.8, the cheapest way to do this is to split the flow onto k paths. The cost of this optimal solution is $\frac{2\phi^2+2}{k}$.

To see that the given solution is indeed a Nash equilibrium, the same reasoning as in Theorem 6.8 applies. We observe that a player can always do best by splitting their flow among k paths, using the same analysis as in Theorem 6.8. Further, given that a player is splitting their flow on k paths, it is optimal to send $\frac{1}{k}$ units of flow to each copy of the Christodoulou-Koutsoupias graph. Because the solution restricted to each of these graphs is a Nash equilibrium, this shows that the overall solution is a Nash equilibrium as well. The cost of this equilibrium is $\frac{4\phi^2+4\phi+2}{k}$, which gives a price of anarchy of at least $\frac{4\phi^2+4\phi+2}{2\phi^2+2} = \phi + 1 \approx 2.618$. □

Our last lower bound on the price of anarchy concerns k-splittable weighted network congestion games.

Theorem 6.11 *For any value of k, there are instances of k-splittable weighted network congestion games with linear costs in which the price of anarchy is at least $\frac{32k\phi+24k}{9k\phi+17k-\phi-1}$.*

Proof: The technique behind this proof is similar to that used in Theorem 6.9. We modify each of the component graphs in the example of Theorem 6.10 as shown in Figure 6-8. This has the effect of establishing a penalty of Bx if player 1 or 2 takes their short path. We choose a value of $B = \frac{k-1}{8k}$.

We claim that sending $\frac{1}{k}$ units of flow along each of the short paths constitutes an optimal solution, and sending $\frac{1}{k}$ units of flow along each of the long paths gives a Nash

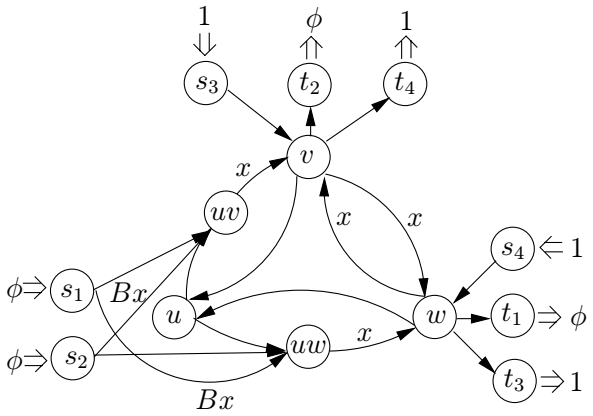

Figure 6-8: Modification of the component graphs

equilibrium. To see that sending $\frac{1}{k}$ units of flow along the short paths is optimal, we use a nearly identical analysis to that in Theorem 6.9. The cost of this optimal solution is $\frac{2\phi^2+2+2\phi^2 B}{k}$.

The proof that the given solution constitutes a Nash equilibrium is also shown in an identical manner to Theorem 6.9. We again observe that a minimum cost solution always has the flow distributed evenly among the 'short' and 'long' paths that it uses. We can use this to determine a value of B based on the number i of short paths taken; we find

$$B \geq \frac{k-i}{8k}.$$

This implies that the strongest condition on B occurs when we send flow along 1 short path and $k-1$ long paths, giving a bound of

$$B \geq \frac{k-1}{8k},$$

which is exactly the bound we chose. Hence the solution is again a Nash equilibrium, with cost $\frac{4\phi^2+4\phi+2}{k}$. The price of anarchy in this case is

$$\frac{\frac{4\phi^2+4\phi+2}{k}}{\frac{2\phi^2+2+2\phi^2 B}{k}} = \frac{32k\phi + 24k}{9k\phi + 17k - \phi - 1}. \quad \square$$

6.5.2 Upper Bounds on the Price of Anarchy

The first of our upper bounds is based on a theorem of Awerbuch et al. [10] for unsplittable network congestion games, which we modify to apply to splittable network congestion games. Note that the splittability factor is not explicitly referenced in the proof, so the same proof applies to *k-splittable* network congestion games as well.

Theorem 6.12 (based on [10]) *The price of anarchy in (infinitely) splittable weighted network congestion games with linear costs is at most* $\frac{3+\sqrt{5}}{2} \approx 2.618$.

Proof: Let x be a Nash equilibrium for the game, and let x^* be an optimal solution. Let x_a and x_a^* denote the total amount of flow on arc a under solutions x and x^* respectively. Let f_a^i be the amount of flow on arc a by player i in solution x, and f_a^{*i} be the amount of flow on arc a by player i in solution x^*.

Since x is a Nash equilibrium, the cost $C_i(x)$ of player i in solution x satisfies:

$$C_i(x) = \sum_{a \in A} (q_a x_a + r_a) f_a^i \leq \sum_{a \in A} (q_a (x_a + f_a^{*i}) + r_a) f_a^{*i}.$$

The term on the right hand side is an upper bound on the cost if player i switches to its flow vector in the optimal solution x^*.

Summing over all players $i = 1, \ldots, n$, we obtain that the total cost $C(x)$ is equal to:

$$C(x) = \sum_{a \in A} \sum_{i=1}^{n} (q_a x_a + r_a) f_a^i \leq \sum_{a \in A} \sum_{i=1}^{n} [(q_a x_a + r_a) f_a^{*i} + q_a (f_a^{*i})^2].$$

Note that we have reversed the usual order of summation here, which is valid since the arcs $a \in A$ and the players i are independent. We have also rewritten the right hand term.

Next, we will make use of the following three facts:

$$\sum_{i=1}^{n} f_a^i = x_a \qquad \sum_{i=1}^{n} f_a^{*i} = x_a^* \qquad \sum_{i=1}^{n} (f_a^{*i})^2 \leq (x_a^*)^2 \tag{6.1}$$

The first two facts follow by definition, and the third follows since $f_a^{*i} \geq 0$ for every arc $a \in A$ and player i. Substituting these facts into the expression for $C(x)$, we obtain:

$$C(x) = \sum_{a \in A} (q_a x_a + r_a) x_a \leq \sum_{a \in A} [(q_a x_a + r_a) x_a^* + q_a (x_a^*)^2].$$

which is equal to

$$\sum_{a \in A} q_a x_a x_a^* + \sum_{a \in A} (q_a x_a^* + r_a) x_a^*.$$

Using the Cauchy-Schwartz inequality on the term $\sum_{a \in A} q_a x_a x_a^*$ on the right hand side, we get:

$$\begin{aligned} C(x) &= \sum_{a \in A} (q_a x_a + r_a) x_a \\ &\leq \sqrt{\sum_{a \in A} q_a (x_a)^2 \sum_{a \in A} q_a (x_a^*)^2} + \sum_{a \in A} (q_a x_a^* + r_a) x_a^*. \end{aligned}$$

This in turn implies that

$$\begin{aligned} C(x) &= \sum_{a \in A} (q_a x_a + r_a) x_a \\ &\leq \sqrt{\sum_{a \in A} (q_a x_a + r_a) x_a \sum_{a \in A} (q_a x_a^* + r_a) x_a^*} + \sum_{a \in A} (q_a x_a^* + r_a) x_a^*, \end{aligned}$$

since we have merely increased the size of the right hand side. Simplifying, this gives:

$$C(x) \leq \sqrt{C(x) C(x^*)} + C(x^*).$$

Now, let $y = \sqrt{\frac{C(x)}{C(x^*)}}$ be the square root of the ratio of the cost of the Nash equilibrium to the cost of the optimal solution. Dividing both sides of the previous inequality by $C(x^*)$ and rewriting it in terms of y, we obtain:

$$y^2 \leq y + 1.$$

Using the quadratic formula, this implies

$$y \leq \frac{1+\sqrt{5}}{2} \quad \Rightarrow \quad y^2 \leq \frac{3+\sqrt{5}}{2}.$$

Finally, note that this proof holds regardless of the value of k, since at no point did we explicitly rely on the value of k in any of our arguments. $\Box$

For the special case of $\frac{1}{2}$-integral unweighted network congestion games with linear costs, we can do slightly better, as is outlined in the next theorem.

Theorem 6.13 *The price of anarchy in $\frac{1}{2}$-integral unweighted network congestion games with linear costs is at most 2.6.*

Proof: Let x be a Nash equilibrium, and let x^* be an optimal solution. Define x_a, x_a^*, f_a^i, and f_a^{*i} as in the proof Theorem 6.12.

Recall the bound on the overall cost $C(x)$ derived in the proof of Theorem 6.12:

$$C(x) \leq \sum_{a \in A} \sum_{i=1}^{n} [q_a(x_a + f_a^{*i}) f_a^{*i} + r_a f_a^{*i}].$$

Using the three facts (6.1) from the previous theorem, along with the fact that $f_a^{*i} \leq 1$ for all i, we can rewrite the sum as:

$$C(x) \leq \sum_{a \in A | x_a^* = \frac{1}{2}} [q_a(x_a + x_a^*)x_a^* + r_a x_a^*] + \sum_{a \in A | x_a^* \geq 1} [q_a(x_a + 1)x_a^* + r_a x_a^*].$$

We now make use of the following two inequalities, which can easily be proven.

Inequality 1. $(y+z)z \leq \frac{2}{7}y^2 + \frac{13}{7}z^2$ for $y \geq 0$ and $\frac{1}{2}$-integral, and $z = \frac{1}{2}$

Inequality 2. $(y+1)z \leq \frac{2}{7}y^2 + \frac{13}{7}z^2$ for $y \geq 0$ and $z \geq 1$, with y, z $\frac{1}{2}$-integral

Using these inequalities, we can rewrite the sum as:

$$C(x) \leq \sum_{a \in A} [q_a(\frac{2}{7}{x_a}^2 + \frac{13}{7}{x_a^*}^2) + r_a x_a^*].$$

By adding $\frac{2}{7} r_a x_a + \frac{6}{7} r_a x_a^*$ to the right hand side, this further implies:

$$C(x) = \sum_{a \in A} (q_a x_a + r_a) x_a \leq \sum_{a \in A} \left[\frac{2}{7} (q_a x_a + r_a) x_a + \frac{13}{7} (q_a x_a^* + r_a) x_a^* \right],$$

which shows that

$$C(x) \leq \frac{13}{5} C(x^*).$$

Hence the price of anarchy is at most 2.6. □

Finally, we comment that the approach used in Theorem 6.13 can be extended to $\frac{1}{k}$-*integral* unweighted network congestion games. Using the same arguments, we obtain a price of anarchy of $\frac{8k-3}{3k-1}$ in the general case. As this bound is weaker than the bound of Theorem 6.12 for all values of k except $k = 1$ and $k = 2$, this does not appear to be an advantageous method for obtaining stronger bounds.

6.5.3 Nonmonotonicity of the Price of Anarchy

We conclude with two results on the behavior of the price of anarchy in k-splittable flows as the value of k varies. The first result shows that the price of anarchy in k-splittable flows may not always be monotone with the value of k, and the second result gives that the ratio of the cost of a worst k-splittable Nash equilibrium to that of an (infinitely) splittable system optimum may also be nonmonotonic.

Theorem 6.14 *There exists a class of congestion games on the same network where the price of anarchy of k-splittable flows in the instance is not monotone with k.*

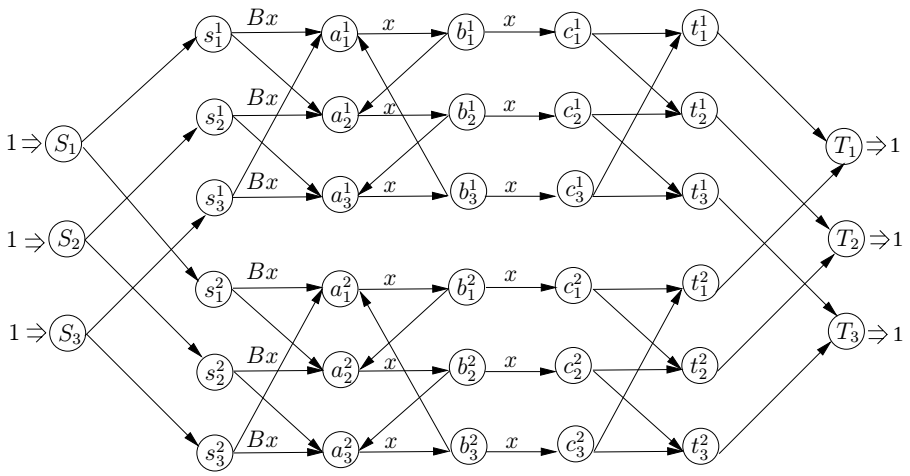

Figure 6-9: Example for the price of anarchy in 2-splittable flows

Proof: We give an instance of an unweighted congestion game in which the price of anarchy for 2-splittable flow is greater than the price of anarchy for *both* unsplittable flow and 3-splittable flow. The network is the same as the example used to prove the lower bound on the price of anarchy in 2-splittable flows, as covered in Theorems 6.8 and 6.9 and shown in Figure 6-9. Here, we set the value of B equal to $\frac{2-1}{12(2)} = \frac{1}{24}$.

We claim the price of anarchy for *unsplittable* flows in this instance is 1. In other words, the only Nash equilibria in this case are also minimum cost solutions. To prove this, we can verify by inspection that minimum cost solutions occur when each player is sent along one of the straight paths. This follows from the optimality of such solutions in the original Christodoulou and Koutsoupias [21] example. To see that these solutions are at the same time the only Nash equilibria, we can either exhaustively check the possible solutions or simply observe that no matter the situation, a player can always improve their cost by switching from a crooked path to one of the straight paths.

From Theorem 6.9, we know the price of anarchy for *2-splittable* flows in this instance is 2.449.

In the case of *3-splittable* flows, we claim that the minimum cost solution occurs when each player sends $\frac{1}{2}$ unit of flow along each of their straight paths. This is the same as the optimal solution for 2-splittable flows (see Theorem 6.9), and it has a cost of 3.0625. To see

that this solution is optimal, we observe that each player must traverse at least two arcs of cost x on each path it uses from source to sink, and the cheapest way to do so is to send all of the flow along only the straight paths, while splitting the flow evenly.

To see that the price of anarchy for 3-splittable flows is less than that of 2-splittable flows, we first notice that by a similar argument as in the proofs of Theorems 6.8 and 6.9, in a Nash equilibrium each player will route all of their flow along either 'straight' or 'crooked' paths. This is since any other possible path is strictly dominated by either a straight or a crooked path. Moreover, in any Nash equilibrium each player will send $\frac{1}{2}$ unit of flow to each of the copies of the Christodoulou-Koutsoupias graph.

Given that all paths are either 'straight' or 'crooked' and that each player sends $\frac{1}{2}$ unit of flow to each graph, the most expensive possible solution occurs when each player sends $\frac{1}{2}$ unit of flow on each of the crooked paths, as this solution maximizes the number of expensive arcs taken by each player. The cost of this solution is 7.5, by Theorem 6.9. Finally, note that this solution is *not* a Nash equilibrium in the 3-splittable case; any player can improve their solution slightly by rerouting some of the flow on its crooked path in one of the copies to an additional (third) straight path. This implies that any Nash equilibrium must have cost strictly less than 7.5, which shows that the price of anarchy for 3-splittable flows is strictly less than 2.449.

Altogether, this implies that the price of anarchy for unsplittable and 3-splittable flows is *less* than that of 2-splittable flows for games on this network. Hence the price of anarchy is not necessarily monotone with the value of k. □

Corollary 6.15 *The ratio of the cost of a worst k-splittable Nash equilibrium to that of an (infinitely) splittable system optimum in an instance may not be monotone with k.*

Proof: The same example from Theorem 6.14 establishes this result. In this instance, the (infinitely) splittable system optimum is for each player to send $\frac{1}{2}$ units of flow along each of the straight paths, for a total cost of $\frac{6+3B}{2} = 3.0625$.

The only *unsplittable* Nash equilibrium is for each player to send all their flow along one of the straight paths, which has a cost of $6 + 3B = 6.125$. As discussed in Theorem 6.9,

the worst *2-splittable* Nash equilibrium is for each player to send $\frac{1}{2}$ units of flow along each of the crooked paths, for a cost of 7.5. By a similar argument to Theorem 6.14, the worst *3-splittable* Nash equilibrium is strictly cheaper than this solution.

This gives that the ratio of the cost of the worst *unsplittable* Nash equilibrium to that of the system optimum is 2, the ratio of the worst *2-splittable* Nash equilibrium is 2.449, and the ratio of the worst *3-splittable* Nash equilibrium is less than 2.449. Hence the ratios for a given example may be nonmonotonic. □

6.5.4 Price of Anarchy in Undirected Network Congestion Games

We now comment on the price of anarchy in *undirected* network congestion games. These are network congestion games in which each edge may be traversed in either direction. The load on an edge is equal to the total number of players traversing the edge. The existence and computability results for Nash equilibria in Sections 6.3 and 6.4 for directed networks can be seen to apply to this version of the problem as well.

All of our upper bounds on the price of anarchy from Section 6.5.2 can also be shown to apply to this problem. This is since in all of our proofs of the upper bounds, at no point did we explicitly rely on the directionality of any of the arcs (in fact, these proofs would apply equally well to general congestion games). Thus these results also extend to undirected network games.

Unfortunately, the examples used to prove the lower bounds in Section 6.5.1 do not translate to the undirected case. All of these examples specifically rely on the directionality of the arcs, and the same construction techniques do not apply when the arcs are undirected. The best lower bound we have currently been able to find is a bound of 2, as illustrated in the following two theorems.

Theorem 6.16 *There are instances of unsplittable, unweighted network congestion games on undirected networks in which the price of anarchy is at least 2.*

Proof: Consider the following unweighted, unsplittable congestion game:

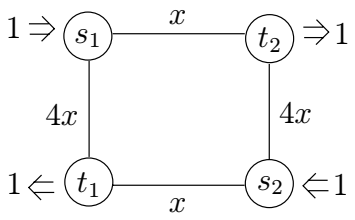

Figure 6-10: Example for the price of anarchy in undirected network congestion games

Here there are two players, each with a single unit of demand, and undirected edges. An optimal solution is for player 1 to take the edge $s_1 - t_1$ and for player 2 to take the edge $s_2 - t_2$, giving a solution of cost 8. Note that this solution also defines a Nash equilibrium. However, there is another more Nash equilibrium that arises when player 1 takes the path $s_1 - t_2 - s_2 - t_1$ and player 2 takes the path $s_2 - t_1 - s_1 - t_2$. The cost of this solution is 16, giving a price of anarchy of 2. □

We can extend this example to apply to $\frac{1}{k}$-integral network congestion games, by using the same techniques as in Section 6.5.1. This gives the following theorem.

Theorem 6.17 *There are instances of $\frac{1}{k}$-integral unweighted network congestion games on undirected networks in which the price of anarchy is at least 2.*

Proof: The technique behind this result is the same as that used in Theorems 6.8 and 6.10. We create k copies of the graph in Theorem 6.16, super-sources S_1 and S_2, and super-sinks T_1 and T_2. Each of the super-sources and super-sinks is connected via an undirected edge to its corresponding source or sink in every one of the copies. We can verify that the solution where both players send $\frac{1}{k}$ units of flow along each of the 'short' paths is an optimal solution, of cost $\frac{8}{k}$. A Nash equilibrium arises when each player sends $\frac{1}{k}$ units of flow along each of the 'long' paths, for a cost of $\frac{16}{k}$. This gives a price of anarchy of 2. □

We have not yet been able to extend this result to k-splittable network congestion games in the manner that was used in Theorems 6.9 and 6.11. The difficulty is that we would like to 'force' players taking the straight paths to incur a small additional cost; however, with

the undirected edges it is difficult to enforce such a penalty on just those players that are taking short paths. This remains an intriguing open question, as is the matter of obtaining a stronger overall bound on the price of anarchy in undirected games.

6.6 Conclusions and Open Questions

We have shown a number of results on the existence, computability, and price of anarchy of pure Nash equilibria in k-splittable and $\frac{1}{k}$-integral network congestion games. In terms of existence and computability, we have seen that Nash equilibria are guaranteed to exist for both problems, with a slight restriction, and that such equilibria may be computed in pseudopolynomial time.

The problem of computing a Nash equilibrium in polynomial time is quite possibly a difficult one, as it is a very large-scale neighborhood search problem and existing VLSN search techiques (see [3, 4] for a survey) do not seem to apply. Panagopoulou and Spirakis [69] provide experimental evidence suggesting that local search techniques converge to an equilibrium in polynomial time in many cases; however, a polynomial algorithm remains unstraightforward to obtain.

Perhaps our most striking results are those for the price of anarchy. These results suggest that the price of anarchy for k-splittable flow is much closer to that of *unsplittable* flow than that of (infinitely) splittable flow, even as k grows very large. It would be interesting to see whether these results could be extended to values of k that grow with the size of the network, as the lower bounds presented in this section hold only for fixed values of k.

It would also be worthwhile to try and strengthen the remaining gaps in the bounds on the price of anarchy, both for directed and undirected games. It could be that a strengthening of the lower bounds is possible, by coming up with a novel construction. Existing techniques for proving the upper bounds seem unlikely to produce any stronger results, though it is possible that a new technique could improve upon these as well. Finally, it would be interesting to see whether we can obtain tighter bounds on the price of anarchy in undirected *weighted* network congestion games, as this issue was not addressed.

Bibliography

[1] D. Acklie. Statement by the Chairman of the American Trucking Associations. Prepared for the US Senate Hearing to Examine Bus and Truck Security and Hazardous Materials Licensing, 2001.

[2] A. Agarwal, T. Agarwal, S. Chopra, A. Feldmann, N. Kammenhuber, P. Krysta, and B. Vöcking. An experimental study of k-splittable scheduling for DNS-based traffic allocation. In *Proceedings of the Ninth International European Conference on Parallel Processing*, number 2790 in Lecture Notes in Computer Science, pages 230–235, Klagenfurt, Austria, 2003. Springer-Verlag.

[3] R. Ahuja, Ö. Ergun, J. Orlin, and A. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123:75–102, 2002.

[4] R. Ahuja, Ö. Ergun, J. Orlin, and A. Punnen. Very large-scale neighborhood search: theory, algorithms, and applications. Working Paper, Operations Research Center, MIT, Cambridge, MA, 2006.

[5] R. Ahuja, J. Orlin, and T. Magnanti. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, NJ, 1993.

[6] R. Ahuja, J. Orlin, G. Sechi, and P. Zuddas. Algorithms for the simple equal flow problem. *Management Science*, 45:1440–1455, 1999.

[7] A. Ali, J. Kennington, and B. Shetty. The equal flow problem. *European Journal of Operational Research*, 36:107–115, 1988.

[8] E. Anshelevich, A. Dasgupta, J. Kleinberg, É. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. In *Proceedings of the Forty-Fifth Symposium on Foundations of Computer Science*, pages 295–304, Rome, Italy, 2004. IEEE Computer Society.

[9] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties*. Springer-Verlag, Berlin, Germany, 1999.

[10] B. Awerbuch, Y. Azar, and A. Epstein. The price of routing unsplittable flow. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, pages 57–66, Baltimore, MD, 2005. ACM Press.

[11] G. Baier, E. Köhler, and M. Skutella. On the k-splittable flow problem. In *Proceedings of the Tenth European Symposium on Algorithms*, number 2461 in Lecture Notes in Computer Science, pages 101–113, Rome, Italy, 2002. Springer-Verlag.

[12] C. Barnhart and D. Kim. Routing models and solution procedures for regional less-than-truckload operations. *Annals of Operations Research*, 61:67–90, 1995.

[13] C. Barnhart and H. Ratliff. Modeling intermodal routing. *Journal of Business Logistics*, 14:205–233, 1993.

[14] R. Beier, A. Czumaj, P. Krysta, and B. Vöcking. Computing equilibria for congestion games with (im)perfect information. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 739–748, New Orleans, LA, 2004. Society for Industrial and Applied Mathematics.

[15] A. Bockmayr, N. Pisaruk, and A. Aggoun. Network flow problems in constraint programming. In *Proceedings of the Seventh International Conference on Principles and Practice of Constraint Programming*, number 2239 in Lecture Notes in Computer Science, pages 196–210, Paphos, Cyprus, 2001. Springer-Verlag.

[16] J. Bossert. *Modeling and Solving Variations of the Network Loading Problem*. PhD dissertation, Massachusetts Institute of Technology, September 2002.

[17] H. Calvete. Network simplex algorithm for the general equal flow problem. *European Journal of Operational Research*, 150:585–600, 2003.

[18] P. Carraresi and G. Gallo. Network models for vehicle and crew scheduling. *European Journal of Operational Research*, 16:139–151, 1984.

[19] D. Chakrabarty, A. Mehta, V. Nagarajan, and V. Vazirani. Fairness and optimality in congestion games. In *Proceedings of the Sixth ACM Conference on Electronic Commerce*, pages 52–57, Vancouver, BC, 2005. ACM Press.

[20] C. Chau and K. Sim. The price of anarchy for non-atomic congestion games with symmetric cost maps and elastic demands. *Operations Research Letters*, 31:327–334, 2003.

[21] G. Christodoulou and E. Koutsoupias. The price of anarchy in finite congestion games. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, pages 67–73, Baltimore, MD, 2005. ACM Press.

[22] E. Coffman, M. Elphick, and A. Shoshani. System deadlocks. *ACM Computing Surveys*, 3:67–78, 1971.

[23] R. Cominetti, J. Correa, and N. Stier-Moses. Network games with atomic players. Working Paper, Columbia University, New York, NY, 2005.

[24] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, OH, 1971. ACM Press.

[25] J. Correa, A. Schulz, and N. Stier-Moses. Selfish routing in capacitated networks. *Mathematics of Operations Research*, 29:971–976, 2004.

[26] J. Correa, A. Schulz, and N. Stier-Moses. On the inefficiency of equilibria in congestion games. In *Proceedings of the Eleventh International Integer Programming and Combinatorial Optimization Conference*, number 3509 in Lecture Notes in Computer Science, pages 167–181, Berlin, Germany, 2005. Society for Industrial and Applied Mathematics.

[27] A. Czumaj. Selfish routing on the Internet. In *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, chapter 42. Chapman & Hall/CRC, Boca Raton, FL, 2003.

[28] A. Czumaj, P. Krysta, and B. Vöcking. Selfish traffic allocation for server farms. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 287–296, Montréal, Canada, 2002. Society for Industrial and Applied Mathematics.

[29] A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 413–420, San Francisco, CA, 2002. Society for Industrial and Applied Mathematics.

[30] A. Elmagarmid. A survey of distributed deadlock detection algorithms. *ACM SIGMOD Record*, 15:37–45, 1986.

[31] R. Erickson, C. Monma, and A. Veinott Jr. Send-and-split method for minimum-concave-cost network flows. *Mathematics of Operations Research*, 12:634–664, 1987.

[32] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5:691–703, 1976.

[33] A. Fabrikant, C. Papadimitriou, and K. Talwar. The complexity of pure Nash equilibria. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, pages 604–612, Chicago, IL, 2004. ACM Press.

[34] G. Facchini, F. van Megen, P. Borm, and S. Tijs. Congestion models and weighted Bayesian potential games. *Theory and Decision*, 42:193–206, 1997.

[35] J. Feldman and D. Karger. Decoding turbo-like codes via linear programming. In *Proceedings of the Forty-Third Symposium on Foundations of Computer Science*, pages 251–260, Vancouver, Canada, 2002. IEEE Computer Society.

[36] R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Nashification and the coordination ratio for a selfish routing game. In *Proceedings of the Thirtieth International Colloquium on Automata, Languages and Programming*, number 2719 in Lecture Notes in Computer Science, pages 514–526, Eindhoven, The Netherlands, 2003. Springer-Verlag.

[37] S. Fischer and B. Vöcking. On the structure and complexity of worst-case Equilibria. In *Proceedings of the First International Workshop on Internet and Network Economics*, number 3828 in Lecture Notes in Computer Science, pages 151–160, Hong Kong, China, 2005. Springer-Verlag.

[38] S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10:111–121, 1980.

[39] D. Fotakis, S. Kontogiannis, and P. Spirakis. Selfish unsplittable flows. *Theoretical Computer Science*, 348:226–239, 2005.

[40] D. Fotakis, S. Kontogiannis, and P. Spirakis. Symmetry in network congestion games: Pure equilibria and anarchy cost. In *Proceedings of the Third Workshop on Approximation and Online Algorithms*, number 3879 in Lecture Notes in Computer Science, pages 161–175, Mallorca, Spain, 2005. Springer-Verlag.

[41] C. Fremuth-Paeger and D. Jungnickel. Balanced network flows 1: A unifying framework for design and analysis of matching algorithms. *Networks*, 33:1–28, 1999.

[42] M. Gairing, T. Lücking, M. Mavronicolas, B. Monien, and P. Spirakis. Extreme Nash equilibria. In *Proceedings of the Eighth Italian Conference on Theoretical Computer Science*, number 2841 in Lecture Notes in Computer Science, pages 1–20, Bertinoro, Italy, 2003. Springer-Verlag.

[43] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, NY, 1979.

[44] G. Glocker and G. Nemhauser. A dynamic network flow problem with uncertain arc capacities: formulation and problem structure. *Operations Research*, 48:233–242, 2000.

[45] M. Goemans, V. Mirrokni, and A. Vetta. Sink equilibria and convergence. In *Proceedings of the Forty-Sixth Symposium on Foundations of Computer Science*, pages 142–154, Pittsburgh, PA, 2005. IEEE Computer Society.

[46] D. Goldberg, J. Feldman, and C. Stein. The integral unit-capacity maximum flow problem with homologous arcs. Working Paper, Columbia University, New York, NY, 2006.

[47] D. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, Boston, MA, 1997.

[48] R. Holzman and N. Law-Yone. Strong equilibrium in congestion games. *Games and Economic Behavior*, 21:85–101, 1997.

[49] R. Johari and J. Tsitsiklis. Efficiency loss in a network resource allocation game. *Mathematics of Operations Research*, 29:407–435, 2004.

[50] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[51] W. Kocay and D. Stone. Balanced network flows. *Bulletin for the Institute of Combinatorics and its Applications*, 7:17–32, 1993.

[52] R. Koch, I. Spenke, and M. Skutella. Approximation and complexity of k-splittable flows. In *Proceedings of the Third Workshop on Approximation and Online Algorithms*, number 3879 in Lecture Notes in Computer Science, pages 244–257, Mallorca, Spain, 2005. Springer-Verlag.

[53] H. Konishi, M. Le Breton, and S. Weber. Equilibria in a model with partial rivalry. *Journal of Economic Theory*, 72:225–237, 1997.

[54] S. Kontogiannis and P. Spirakis. Atomic selfish routing in networks: a survey. In *Handbook of Parallel Computing: Models, Algorithms, and Applications*. Chapman & Hall/CRC, Boca Raton, FL, 2007. To appear.

[55] A. Kothari, S. Suri, C. Tóth, and Y. Zhou. Congestion games, load balancing, and price of anarchy. In *Proceedings of the First Workshop on Combinatorial and Algorithmic Aspects of Networking*, number 3405 in Lecture Notes in Computer Science, pages 13–27, Banff, Canada, 2004. Springer-Verlag.

[56] E. Koutsoupias, M. Mavronicolas, and P. Spirakis. Approximate equilibria and ball fusion. *ACM Transactions on Computer Systems*, 36:683–693, 2003.

[57] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 404–413, Trier, Germany, 1999. Springer-Verlag.

[58] P. Krysta, P. Sanders, and B. Vöcking. Scheduling and traffic allocation for tasks with bounded splittability. In *Proceedings of the Twenty-Eighth International Symposium on Mathematical Foundations of Computer Science*, number 2747 in Lecture Notes in Computer Science, pages 500–510, Bratislava, Slovak Republic, 2003. Springer-Verlag.

[59] T. Larsson and Z. Liu. An efficient Lagrangean relaxation scheme for linear and integer equal flow problems. *Optimization*, 40:247–284, 1997.

[60] H. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[61] C. Li, S. McCormick, and D. Simchi-Levi. The point-to-point delivery and connection problems: complexity and algorithms. *Discrete Applied Mathematics*, 36:267–292, 1992.

[62] T. Lücking, M. Mavronicolas, B. Monien, M. Rode, P. Spirakis, and I. Vrto. Which is the worst-case Nash equilibrium? In *Proceedings of the Twenty-Eighth International Symposium on Mathematical Foundations of Computer Science*, number 2747 in Lecture Notes in Computer Science, pages 551–561, Bratislava, Slovak Republic, 2003. Springer-Verlag.

[63] M. Mavronicolas and P. Spirakis. The price of selfish routing. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 510–519, Crete, Greece, 2001. ACM Press.

[64] I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13:111–124, 1996.

[65] I. Milchtaich. Generic uniqueness of equilibrium in large crowding games. *Mathematics of Operations Research*, 25:349–364, 2000.

[66] I. Milchtaich. Social optimality and cooperation in nonatomic congestion games. *Journal of Economic Theory*, 114:56–87, 2004.

[67] D. Monderer and L. Shapley. Potential games. *Games and Economic Behavior*, 14:124–143, 1996.

[68] J. Nash. Non-cooperative games. *Annals of Mathematics*, 54:286–295, 1951.

[69] P. Panagopoulou and P. Spirakis. Efficient convergence to pure Nash equilibria in weighted network congestion games. In *Proceedings of the Fourth International Workshop on Efficient and Experimental Algorithms*, number 3503 in Lecture Notes in Computer Science, pages 203–215, Santorini, Greece, 2005. Springer-Verlag.

[70] C. Papadimitriou. Computing correlated equilibria in multi-player games. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, pages 49–56, Baltimore, MD, 2005. ACM Press.

[71] W. Powell. *Supply Chain Management: Design, Coordination, and Operation*, volume 11 of *Handbooks in Operations Research and Management Science*, chapter 13. Elsevier, Amsterdam, the Netherlands, 2003.

[72] J. Rosen. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica*, 33:520–534, 1965.

[73] R. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[74] T. Roughgarden. The price of anarchy is independent of the network topology. *Journal of Computer and System Sciences*, 67:341–364, 2003.

[75] T. Roughgarden. *Selfish Routing and the Price of Anarchy*. MIT Press, Cambridge, MA, 2005.

[76] T. Roughgarden and É. Tardos. How bad is selfish routing? *Journal of the ACM*, 49:236–259, 2002.

[77] T. Roughgarden and É. Tardos. Bounding the inefficiency of equilibria in nonatomic congestion games. *Games and Economic Behavior*, 47:389–403, 2004.

[78] S. Sahni. Computationally related problems. *SIAM Journal on Computing*, 3:262–279, 1974.

[79] W. Sandholm. Potential games with continuous player sets. *Journal of Economic Theory*, 97:81–108, 2001.

[80] H. Shachnai and T. Tamir. Multiprocessor scheduling with machine allotment and parallelism constraints. *Algorithmica*, 32:651–678, 2002.

[81] F. Shepardson and R. Marsten. A Lagrangean relaxation algorithm for the two duty period scheduling problem. *Management Science*, 26:274–281, 1980.

[82] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, Boston, MA, 1997.

[83] K. Srinathan, P. Goundan, M. Ashwin Kumar, R. Nandakumar, and C. Pandu Rangan. Theory of equal flows in networks. In *Proceedings of the Eighth International Computing and Combinatorics Conference*, number 2387 in Lecture Notes in Computer Science, pages 514–524, Singapore, 2002. Springer-Verlag.

[84] S. Suri, C. Tóth, and Y. Zhou. Selfish load balancing and atomic congestion games. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 188–195, Barcelona, Spain, 2004. ACM Press.

[85] T. Ui. A Shapley value representation of potential games. *Games and Economic Behavior*, 31:121–135, 2000.

[86] V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin, Germany, 2001.

[87] B. Verweij, K. Aardal, and G. Kant. On an integer multicommodity flow problem from the airplane industry. Technical Report UU-CS-1997-38, Utrecht University, Department of Computer Science, 1997.

[88] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, New Jersey, 1947.

[89] M. Voorneveld. Equilibria and approximate equilibria in infinite potential games. *Economic Letters*, 56:163–169, 1997.

[90] M. Voorneveld, P. Borm, F. van Megen, S. Tijs, and G. Facchini. Congestion games and potentials revisited. *International Game Theory Review*, 1:283–299, 1999.